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

Tarea Grande 2

Profesor Vicente Domínguez, Luis Ramírez

Anunciada: 30 de Septiembre de 2019

Indicaciones

- Fecha de Entrega: 14 de Octubre de 2019
- Debes entregar tu tarea en tu repositorio GitHub privado asignado para esta evaluación.
- Cada hora o fracción de atraso descuenta 0,5 puntos de la nota que obtengas.
- La tarea es *en parejas*. La copia será evaluada con nota 1 en el la tarea, además de las sanciones disciplinarias correspondientes.

Objetivo

El objetivo de esta tarea es que aprendas a:

- Analizar un set de datos para elegir las mejores características para solucionar un problema.
- Utlizar Pandas de Python para preprocesar un set de datos.
- Utilizar `scikit-learn` de Python para machine learning.
- Usar modelos de clasificación (aprendizaje supervisado) y medir su rendimiento.
- Usar algoritmos de clustering (aprendizaje no supervisado) y observar sus limitaciones.
- Visualizar resultados mediante `Altair` de Python

Descripción de la tarea

Esta tarea busca que explores dos áreas de *machine learning*: aprendizaje supervisado y no supervisado. Adicionalmente, aprenderás a hacer análisis exploratorio de datos. Para ello usarás un set de datos que se recolectó mediante encuestas a personas que trabajan. El dataset se encuentra en el repositorio del curso, en el archivo `adult_dataset.csv`. Este archivo contiene información detallada sobre personas: edad, profesión, estado civil, educación, horas trabajadas a la semana, entre otros. Además, para cada persona indica si gana más o menos de 50 mil dólares anuales.

Deberás trabajar en grupos de a dos y para ello recibirás un enlace de Github Classroom. El primer miembro debe registrar el equipo y el segundo debe incorporarse a ese equipo. Se debe trabajar usando Github de manera correcta: ambos miembros deben hacer commits al repositorio y estos tienen que ser atómicos y descriptivos (pueden aplicar descuentos). Este repositorio viene con un archivo README.md el cual deberán leer y luego modificar con sus datos además de un template de jupyter notebook para comenzar la tarea.

Las tres partes de la tarea se responden en un único Jupyter notebook (`.ipynb`), que deben entregar en el repositorio asignado a cada pareja. Para trabajar con Jupyter notebook es recomendable que usen Google Colab, así evitan tener que instalar Jupyter localmente y evitan la instalación de todas las demás librerías.

Para cada sección, deberán escribir código que haga lo pedido, además de responder las preguntas planteadas en estilo **markdown**, debajo del código correspondiente. Debe considerar que, para poder optar a el puntaje completo en una sección, tanto el código entregado como la respuesta a la pregunta deben estar correctos. De lo contrario, el puntaje obtenido será parcial. **Se recomienda fuertemente que discutan las respuestas a la preguntas con su compañero o compañera de grupo.**

Set de datos

En el archivo `adult_dataset.csv`, cada fila corresponde a una persona y posee los siguientes atributos por columna:

- `ID:` Identificador de la persona, es un **número entero** y este no se repite.
- `Age:` La edad de la persona. Es un **número entero**.
- `Workclass:` El rubro en el que trabaja la persona. Es un **string**.

- `FNLWGT:` Es el *final weight*, que corresponde al número de personas de la población que se ven representadas por esa respuesta. Es un **número**.
- `Education:` El nivel educativo de la persona. Es un **string**.
- `Education_Num:` Es el mismo nivel educativo de la columna anterior, pero esta vez representado por un **número entero** que es mayor mientras más estudios tiene la persona.
- `Marital_Status:` Estado civil de la persona. Es un **string**.
- `Ocupation:` El rubro en el que trabaja la persona. Es un **string**.
- `Relationship:` Indica si la persona es soltera, tiene pareja o similar. Es un **string**.
- `Sex:` Género de la persona. Es un **string**.
- `Capital_Gain:` Es la ganancia de capital de la persona, es decir, ganancias de objetos como acciones o bienes raíces. Es un **número**.
- `Capital_Loss:` Similar al término anterior, pero se refiere a las pérdidas. Es un **número**.
- `Hours_per_Week:` Horas semanales que trabaja la persona. Es un **número entero**.
- `Country:` País de donde proviene la persona. Es un **string**.
- `Target:` Valor que queremos predecir, corresponde a si la persona gana menos o más de 50K USD anuales. Es un `string`.

Parte 1: Procesamiento de los datos (2 ptos)

En esta primera parte deberás cargar, analizar y limpiar los datos. Para ello es obligatorio el uso de Pandas y está prohibido utilizar ciclos, pues se espera que el código sea eficiente.

Cargar dataset: Lo primero que debes hacer es cargar tu dataset desde el archivo `adult_dataset.csv` en tu programa usando Pandas.

Análisis de las características (0.4): Ya se detallaron las columnas (también llamadas características o *features*) que tiene el dataset. Sin embargo, hay algunas de ellas que contienen información redundante, o bien no aportan información para determinar la clase. Se espera que después de cargar los datos, elimines estas columnas.

Pregunta: ¿Qué columnas eliminaste y por qué?

Limpiar valores nulos (0.4): Si observas bien los datos, te darás cuenta que en algunas filas faltan datos. En Pandas, los valores nulos se ven como `NaN` (*not a number*) dentro del DataFrame. En general para trabajar en machine learning, se necesita que los datos estén completos, por lo tanto se espera que busques una solución al problemas de los datos faltantes.
Pregunta: ¿Qué hiciste con los valores nulos? Justifica cada una de las acciones que hayas tomado.

Manejar atributos de tipo string (0.6): Gran parte de las columnas del dataset tienen valores de tipo string. Sin embargo, algunos clasificadores (como KNN) solo pueden trabajar con valores numéricos. La librería que utilizaremos más adelante también requiere que los valores sean numéricos. Por esta razón, se espera que transformes las columnas de tipo string a una representación numérica.
Pregunta: ¿Cuál es la diferencia entre una variable categórica ordinal y una categórica nominal? De las columnas que son strings, ¿cuáles son de cada tipo?

Separar características de clase (0.2): Antes de empezar a trabajar con modelos de clasificación, debes separar tu dataset en una matriz de características (*features*) y un vector de clases.

Normalizar datos (0.4): En algunos algoritmos de clasificación es importante que los valores numéricos estén más o menos en la misma escala para que funcionen correctamente. A este proceso de llevar los valores a otra escala le llamamos normalización. Se espera que normalices las características de tu dataset.
Pregunta: ¿Qué normalizaste, filas o columnas? ¿Por qué? Explica por qué es tan importante normalizar en un algoritmo como KNN.

Todo lo que hagas de ahora en adelante será trabajando con estos datos (sin columnas innecesarias, sin valores nulos y con valores normalizados).

Parte 2: Clasificación automática (2.2 ptos)

En esta sección deberás entrenar un modelo de clasificación. El objetivo es entrenar el modelo usando una parte de los datos y luego evaluar su capacidad de clasificar usando otra parte de los datos. Por último obtendrás estadísticas para ver cómo lo hizo tu modelo.

Existen varios tipos de modelos de clasificación, pero en esta oportunidad usarás uno de los más sencillos llamado Decision Tree (DT). En esta sección tendrás que ayudarte de la librería `scikit-learn`, que cuenta con muchas funciones y herramientas para todo lo que se te pide hacer.

Separar datos de entrenamiento y pruebas (0.2): para usar un modelo de clasificación debes separar tu dataset en dos: una parte de los datos se usarán para entrenar el modelo de clasificación y la otra para probarlo. Se espera que obtengas el set de entrenamiento y el set de pruebas usando proporciones razonables.

Instanciar y entrenar el clasificador (0.4): ahora que tienes tu set de datos separado, debes instanciar tu modelo DT y entrenarlo usando el set de entrenamiento.

Calcular el rendimiento del clasificador (0.6): como tu modelo ya está entrenado, ahora debes probar su rendimiento. Para ello tienes que pedirle que prediga las clases de tu set de pruebas y ver qué tan bien lo hace (comparando su predicción con las clases reales). Esta métrica se conoce como *accuracy*. También debes buscar qué son las métricas *presicion*, *recall* y *F1-score*, y calcularlas.
Pregunta: ¿Cuánto accuracy obtuviste? ¿Se considera eso un buen resultado? Considerando que hay dos clases, ¿bajo qué valor de accuracy se puede decir que el modelo funciona mal?

Encontrar hiperparámetros (0.4): ahora que sabes instanciar, entrenar y evaluar el clasificador, debes repetir este proceso probando distintos hiperparámetros del modelo. Los hiperparámetros puedes encontrarlos en la documentación de DT y se configuran al crear la instancia. Finalmente debes escoger la combinación de hiperparámetros que mejor rendimiento te entregue.
Pregunta: ¿Qué hiperparámetros modificaste para probar tu clasificador y cuál combinación te dio los mejores resultados? Justifica por qué crees que esto sucede así.

Visualizar la matriz de confusión (0.6): ahora que tienes un modelo definitivo, con sus respectivos hiperparámetros, evalúa su desempeño usando una matriz de confusión. Esta matriz entrega información parecida al *accuracy score* pero con mayor detalle (según cada clase). Se espera que obtengas esta matriz y luego la grafiques usando Altair. Además, cada columna y fila debe contar con una leyenda claramente señalada, al igual que los ejes.
Pregunta: ¿De qué tamaño es la matriz de confusión? ¿Por qué? Explica qué representa el valor en cada una de las celdas de la matriz.

Parte 3: Reducción dimensionalidad y clustering (1.8 ptos)

Lo que hicimos en la sección anterior fue usar nuestros datos para crear un clasificador capaz de predecir las clases en nuevas muestras. Ahora vamos a hacer dos tareas distintas (que no tienen relación con lo anterior) pero son comunes también cuando uno trabaja con set de datos.

La primera tarea corresponde a hacer reducción de dimensionalidad. En palabras simples, la idea es tomar todas las características de nuestro set de datos y hacer una transformación que nos entregue un número menor de características que representen la misma información. Esto nos permite, por ejemplo, llevar los datos a dos dimensiones para visualizarlos.

La segunda etapa corresponde a usar un algoritmo de clustering. La idea de estos algoritmos es poder separar los datos cuando NO conocemos las clases. Esto es un ejemplo de aprendizaje no supervizado.

A diferencia de la primera parte, no utilizarás set de entrenamiento ni set de pruebas, sino que todos los datos juntos. Es decir, utilizarás los datos que obtuviste al final de la primera parte.

Reducción de dimensionalidad (0.4): para reducir la dimensionalidad de los datos debes tomar tu matriz de características y aplicar una técnica llamada *Principal Component Analysis* (PCA), de modo que los datos queden representados con solo dos características (columnas) derivadas de las características originales.
Pregunta: Al usar PCA reducimos la dimensionalidad de los datos. Nombra dos razones para querer hacer eso.

Visualizar datos reducidos (0.4): ahora que has reducido la matriz de características a dos columnas, vas a graficar los datos en dos dimensiones usando Altair [1]. Se espera que cada eje de tu gráfico sea una de las componentes de PCA y cada punto pueda ser identificado con su clase real respectiva (según el color o la forma del punto en el gráfico).

Predecir las clases usando KMeans (0.4): olvidándonos de las clases reales, y solo usando las dos características de PCA, deberás usar el algoritmo KMeans para dividir los datos en clusters. En concreto, usando KMeans debes obtener una predicción de clase para todos los datos reducidos. Notarás que el algoritmo KMeans requiere de un hiperparámetro obligatorio, averigua qué es y usa un valor

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

que tenga sentido para el set de datos.
Pregunta: ¿Qué representa el valor K en el algoritmo KMeans? ¿Qué valor usaste para K y por qué?

Visualizar los clusters de KMeans (0.6): al igual que antes tienes que graficar los datos en dos dimensiones usando Altair. Se espera que cada eje de tu gráfico sea una de las componentes de PCA y cada punto pueda ser identificado con la clase que predijo KMeans (según el color o la forma del punto en el gráfico).
Pregunta: Compara los dos gráficos obtenidos en esta sección. ¿A qué se debe la diferencia entre ellos?

Formato de entrega

La entrega de esta tarea se hará por medio del repositorio GitHub privado de tu grupo, donde deberás entregar un único archivo `.ipynb` con todo el desarrollo de la tarea. Cada subitem de la tarea debe ir en una celda de código, y si el subitem tiene una pregunta a responder, entonces se debe agregar otra celda de texto donde se debe responder a la pregunta. Si trabajaste con Google Collab, entonces desde ahí mismo puedes 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 aplicará descuento.