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

Tarea Grande 2

Profesor Denis Parra

Anunciada: 24 de abril de 2019

Indicaciones

- Fecha de Entrega: 12 de mayo 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.
- Usar el paquete `scikit-learn` de Python.
- 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 astronómicos que se encuentra en el repositorio del curso, en el archivo `astro_data.csv`. Este fichero contiene información sobre observaciones de objetos astronómicos, que pueden ser estrellas, galaxias o quasars.

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. Los nombres de los equipos deben considerar que las tareas son instancias serias y habrá descuentos por nombres poco adecuados.

Las cuatro 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.

En las primeras tres partes tendrás que programar y usar las librerías que te ayuden con esa tarea. La última sección corresponde a un informe en que debes responder preguntas sobre lo que realizaste en las primeras partes y es la parte que tiene mayor puntaje en la tarea. Es recomendable que leas las preguntas antes de empezar a trabajar, para así saber en qué estar más enfocado.

Set de datos

En el archivo `astro_data.csv`, cada fila corresponde a un objeto astronómico y posee los siguientes atributos por columna:

- `ID`: Identificador del objeto atronómico
- `JulianDate`: Fecha de la observación, en calendario Juliano y corregido
- `Ascención recta`: Coordenada celestial del objeto, el análogo a la longitud terrestre
- `Declinación`: Coordenada celestial del objeto, el análogo a la latitud terrestre
- `Redshift`: Desplazamiento del espectro lumínico del objeto hacia el rojo
- `FiltroU`: Medición entregada por el telescopio al aplicar el filtro U

- `FiltroG`: Medición entregada por el telescopio al aplicar el filtro G
- `FiltroR`: Medición entregada por el telescopio al aplicar el filtro R
- `FiltroI`: Medición entregada por el telescopio al aplicar el filtro I
- `FiltroZ`: Medición entregada por el telescopio al aplicar el filtro Z
- `FiltroX`: Medición entregada por el telescopio al aplicar el filtro X, presente solo en algunos telescopios
- `Camcol`: Columna de cámara como parte de una observación
- `Field`: Ventana de filtros en el cielo
- `Plate`: Placa usada en el telescopio que identifica los objetos
- `Clase`: Clase o categoría del objeto astronómico (estrella, galaxia o quasar)

Parte 1: Procesamiento de los datos (1 pto)

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.

Análisis de las características (0.3): En la sección anterior se detalló las características que tiene su dataset, pero puede ser que algunas de esas características no tengan ninguna influencia en la clase. Por lo tanto, después de cargar los datos debes eliminar las columnas que consideras que no aportan información para determinar la clase. Luego en el informe se te pedirá justificar.

Limpiar valores nulos (0.3): El siguiente paso es buscar si hay valores nulos. Usando Pandas, los valores nulos se ven como `NaN` (*not a number*) dentro del DataFrame. Si los hubiera debes tomar una decisión de qué hacer con ellos y justificarla en el informe.

Separar la clase y normalizar (0.4): Por último, antes de empezar a trabajar con modelos de clasificación, debes separar tu dataset en una matriz de características (*features*) y un vector de clases. Además debes normalizar las características.

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 (1,5 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 (en esta instancia no se le entrega la clase para que el modelo haga la predicción). 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 KNN. Este modelo se basa en almacenar puntos que representan los datos de entrenamiento y luego clasifica las instancias según la clase más popular entre sus vecinos más cercanos.

Para trabajar con técnicas de clasificación te recomendamos usar 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 (0.3): al final de la primera parte separaste la matriz de características del vector de clases. Lo que debes hacer ahora es separar estos datos en set de entrenamiento y set de pruebas. Busca en Internet las proporciones que se usan generalmente para separar los datos y hazlo acorde a eso, se te preguntará sobre tu decisión en el informe.

Instanciar y entrenar el clasificador (0.3): ahora que tienes tu set de datos separado, debes instanciar tu modelo KNN y entrenarlo usando el set de entrenamiento.

Calcular el rendimiento del clasificador (0.3): 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* o *score*. Debes mostrar qué score obtuviste en el set de entrenamiento y en el set de pruebas.

Ahora que sabes instanciar, entrenar y evaluar el clasificador, debes repetir este proceso probando distintos parámetros del modelo. Los parámetros puedes encontrarlos en la documentación de KNN y se configuran al crear la instancia. Finalmente debes escoger la combinación de parámetros que mejor rendimiento te entregue, y solo esa debe quedar en tu código. En el informe deberás explicar tus pruebas y decisiones.

Visualizar la matriz de confusión (0.6): ahora que ya escogiste tu modelo, con sus respectivos parámetros, evaluarás su desempeño usando una matriz de confusión. Esta matriz da información muy

parecida al *score* pero con mayor detalle (según cada clase). Se espera que obtengas esta matriz para el set de pruebas y luego la grafiques usando Altair.

Parte 3: Reducción dimensionalidad y clustering (1 pto)

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 lineal que nos entregue un número menor de características que representa 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.

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

Reducción de dimensionalidad (0.2): 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 en dos características (columnas).

Visualizar datos reducidos (0.3): 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 respectiva (según el color o la forma del punto en el gráfico).

Predecir las clases usando KMeans (0.2): 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 requiere de un parámetro obligatorio, tendrás definir ese parámetro y justificarlo

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

en el informe.

Visualizar los clusters de KMeans (0.3): 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).

Parte 4: Informe (2,5 ptos)

En base a todo lo realizado hasta ahora deberás responder una serie de preguntas, para evaluar qué tan bien entendiste los conceptos aplicados. Esta es la parte más importante de tu tarea.

Estas preguntas las debes responder también en tu archivo `.ipynb`, usando una celda en formato *markdown*. Las respuestas tienen una extensión máxima. Para las preguntas 1, 2, 3 y 4 tus respuestas pueden tener una extensión máxima de 5 líneas. Para las preguntas 5, 6, 7, 8 y 9 tus respuestas pueden tener una extensión máxima de 2 líneas. Se debe usar la letra por defecto en el formato *markdown* y no cumplir con estas restricciones tendrá 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 parámetros modificaste para probar tu clasificador? ¿Cuál te dio mejor resultado y por qué crees que es así? (0.3)
5. Explica la diferencia entre el score obtenido en el set de entrenamiento y el set de pruebas. Justifica. (0.3)
6. Usando la matriz de confusión que obtuviste en la segunda parte. Identifica la clase que mejor predice el modelo. Además identifica el par de clases que suele confundirse más. (0.3)
7. Al usar PCA reducimos la dimensionalidad de los datos. Nombra dos razones para hacer eso. (0.2)

8. ¿Qué representa el valor K en el algoritmo KMeans? ¿Qué valor usaste para K y por qué? (0.2)

9. En la tercera parte, compara el gráfico obtenido usando la información de las clases con el gráfico que no utiliza la información de las clases (KMeans). ¿Por qué crees que KMeans falla en identificar perfectamente los clusters reales? (0.3)

Bonus (0,5 ptos)

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

Para obtener el bonus tendrás que usar otro clasificador conocido como Árboles de Decisión (*Decision Trees*), entrenarlo con los datos (usando set de entrenamiento y de pruebas igual que en la segunda parte) y ver su rendimiento (*accuracy*) en el set de pruebas. Deberás comparar el rendimiento obtenido en KNN y en tu árbol de decisión y escribir un breve comentario explicando esa diferencia.

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 y con las preguntas del informe respondidas al final de este. 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.