PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
Facultad de Ingeniería
IIC2613 Inteligencia Artificial
2022-2

Tarea 3

Aspectos generales

Formato y plazo de entrega

El formato de entrega es un archivo con extensión .ipynb. Es importante que a la hora de entregar, tu notebook se encuentre **con las celdas ejecutadas**, de lo contrario no será corregido. Las respuestas a las preguntas teóricas deben incluirse en celdas de texto adyacentes al código que utilices, de manera tal que el texto y las celdas de código sigan una estructura lógica que evidencie tu trabajo. El lugar de entrega es en el repositorio de la tarea, en la *branch* por defecto, hasta el **16 de junio a las 23:59**. Para crear tu repositorio, debes entrar en el enlace del anuncio de la tarea en Canvas. Por último, recuerda que los cupones de atraso son días **no hábiles** extra.

Integridad Académica

Este curso se adhiere al Código de Honor establecido por la universidad, el cual tienes el deber de conocer como estudiante. Todo el trabajo hecho en esta tarea debe ser **totalmente individual**. La idea es que te des el tiempo de aprender estos conceptos fundamentales, tanto para el curso, como para tu formación profesional. Las dudas se deben hacer exclusivamente al cuerpo docente a través de las *issues* en GitHub.

Por otra parte, sabemos que estás utilizando material hecho por otras personas, por lo que es importante reconocerlo de la forma apropiada. Todo lo que obtengas de internet debes citarlo de forma correcta (ya sea en APA, ICONTEC o IEEE). Cualquier falta a la ética y/o a la integridad académica será sancionada con la reprobación del curso y los antecedentes serán entregados a la Dirección de Pregrado.

Comentarios adicionales

Para esta tarea deben usar un conjunto de datos de imágenes, con el que entrenarán diferentes modelos de aprendizaje de máquina. Con ellos, clasificaremos las imágenes según la persona que aparece en ella.
No debes subir el dataset a tu repositorio debido a su gran tamaño. Esto terminaría en un descuento significativo en tu nota.
Puedes encontrar el dataset en este link[1].

[1] https://drive.google.com/file/d/1ZV_nCUfuitYz0vpKFhg5wdwz_LlaVQVo/view?usp=share_link

1. Recursos para realizar la tarea

Usaremos principalmente 3 librerías para esta tarea, las cuales serán explicadas a continuación:

1.1. Pandas

Esta librería[2] es especialmente útil para manejar conjuntos de datos en forma de tablas. Es una extensión de **NumPy**, por lo que los cálculos matemáticos realizados usando las clases de esta librería son muy eficientes, aunque sus mayores atributos están en la forma en la que se indexan los datos. La clase **DataFrame** permite hacer un manejo muy eficiente de estos.

En general, se denominará **DataFrame** a la tabla de datos cuando estos ya estén cargados, y **Series** a las columnas. Para importar la librería por convención utilizamos el diminutivo **pd**:

```
1    import pandas as pd
2
```

Por convención, un **DataFrame** se denomina **df**. Si queremos tener una idea de lo que contiene, podemos usar el método **head**, que despliega los primeros cinco elementos (filas) del dataset, lo que es bastante rápido y ordenado.

Por otro lado, podemos conocer el número de filas y columnas de un **DataFrame** con el atributo **shape**. También pueden resultar útiles los métodos **info** y **describe**. Mientras el primero devuelve características (número de columnas, cuáles son y el número de elementos no nulos), el segundo proporciona información básica estadística del conjunto de datos.

Por otro lado, si queremos limpiar el **DataFrame** de datos indeseados, deberemos conocerlos primero. Para ello, se puede utilizar el método **isnull**. Los elementos nulos aparecen con el denominativo **NaN**, y el método **isnull** devolverá un booleano con un valor correspondiente a si se encuentra dicho denominativo o no. De la misma manera, el método **duplicated** nos ayudará a encontrar duplicados.

Navegar por los datos es relativamente intuitivo. Una columna o serie puede ser accedida como si fuera un atributo de valor **name**, por ejemplo, **df.name**, o como si fuera un diccionario con una llave: **df[“name”]**. Para acceder a dos series se utiliza la extensión **df[[“name1”, “name2”]]**. Una lista de todas las series disponibles se obtiene con **df.columns**. Finalmente, para acceder a un elemento dentro de una columna se debe utilizar su índice luego de haber llamado a la serie. Por ejemplo, al elemento **354** de **name** se accede haciendo **df[“name”][354]**.

Una vez hemos podido acceder a ciertas series, o a elementos de ellas, es posible trabajar de manera numérica. Algunas operaciones matemáticas son bastante simples y directas, mientras que otras pueden requerir un poco más de investigación en la documentación de la librería. Por ejemplo:

```
1    #esta es una operacion simple sobre dos celdas del DataFrame
2    q = df["name1"][354] + df["name2"][726]
3
4    #estas son operaciones sobre toda la serie
5    m = df["name3"].mean()
6    s = df["name4"].sum()
7
```

Una operación numérica interesante sobre la serie es la de **value_counts**, que retornará la cantidad de elementos de cada clase de una serie. En particular, las clases de esa serie se pueden recuperar con el método **unique**.

[2] https://pandas.pydata.org/

Existen accesos más avanzados que se pueden invocar con el método **loc** o el método **groupby**. Estos métodos permiten trabajar con grandes cantidades de datos, hacer consultas y agrupaciones. El método **loc** permite filtrar filas o columnas mediante una etiqueta o un booleano. Un ejemplo puede ser:

```
1     df.loc[df["name1"] > 6, ["name2"]]
2
```

En este ejemplo se filtrarán todas las filas cuya serie de nombre **name1** tenga un valor **mayor que 6**, pero se recuperarán los valores de la serie **name2** de esas filas.

Por otro lado, el método **groupby** retornará un objeto con varias propiedades interesantes. Típicamente se utiliza sobre valores no numéricos a los cuales se les pueden aplicar otras operaciones o métodos:

```
1     df.groupby(["name1"]).sum()
2
```

Si la serie **name1** tiene **5 instancias**, el método mostrará una tabla de **5 filas** para las cuales intentará sumar sus valores en las otras series.

Finalmente, si lo que se desea es guardar un DataFrame procesado, se puede usar el método **to_csv**:

```
1     df.to_csv("nombre del archivo csv")
2
```

1.2. Scikit-learn

Adicionalmente, usaremos la librera scikit-learn [3], la cual provee una gran gama de técnicas de aprendizaje de máquina, tal como clasificadores del tipo **SVM**.
Es una librería orientada a objetos, y también importaremos algunas funciones útiles. Primero que todo, para esta oportunidad nos interesan los clasificadores, que podemos importar haciendo:

```
1     #Clase que utilizaremos para un clasificador SVM
2     from sklearn.svm import SVC
3
4     #Clase que utilizaremos para un clasificador Bayesiano ingenuo
5     from sklearn.naive_bayes import GaussianNB
6
7     #Clase que utilizaremos para un clasificador de arbol de decision
8     from sklearn.tree import DecisionTreeClassifier
9
```

A la hora de trabajar con estas clases se debe tener en cuenta lo siguiente:

- Todos los clasificadores pueden ajustar sus hiperpárametros con el método **set_params**.
- El método **fit(X,y)** ejecutará el procedimiento de optimización de parámetros o entrenamiento del clasificador.
- El método **predict(X)** generará un vector **y** con la predicción del clasificador para la matriz **X**.

1.3. Face Recognition

La librería se encuentra en `https://pypi.org/project/face-recognition/#description`. Se puede instalar con el comando:

[3] https://scikit-learn.org/stable/

```
pip3 install face_recognition
```

Al importar la librería simplemente se debe hacer:

```
import face_recognition as fr
```

Algunas de las funciones **requieren del uso de una GPU**. Más abajo veremos cómo tener un acceso gratis y temporal a una. Por el momento estos son algunos comandos útiles:

- **image = fr.load_image_file("my_picture.jpg")**: Sirve para tener un objeto imagen
- **face_locations = fr.face_locations(image, model="cnn")**: Genera un arreglo de varios puntos que representan cajas contenedoras donde se han hallado rostros.
- **face_landmarks_list = fr.face_landmarks(image)**: Entrega un diccionario con varias listas de puntos que corresponden a marcas faciales. Un ejemplo se puede ver en la Figura 1.
- **my_face_encoding = fr.face_encodings(picture_of_me)[0]** : Entrega un vector con una codificación que representa un rostro.
- **results = fr.compare_faces([my_face_encoding], unknown_face_encoding)**: Devuelve un booleano si ha podido establecer una similitud suficiente entre **my_face_encoding** y **unknown_face_encoding** para determinar si un rostro se parece o a otro.

1.4. Google Colab

Arquitecturas como esta son posibles gracias a las GPUs. Estos son procesadores diseñados para realizar tareas gráficas, pero también han demostrado una gran utilidad en el campo del aprendizaje automático por su alto poder de procesamiento en paralelo. Google ha dispuesto una herramienta gratuita para poder aprovechar su uso. Esta es Google Colab [4], por lo que recomendamos fuertemente desarrollar la tarea en esta plataforma, que además permite exportar el archivo como un archivo **.ipynb**.

IMPORTANTE: Para poder activar el uso de GPU en Google Colab, deben ir a *Editar - Configuración del notebook - Acelerador de harware - GPU*. Si no activan la GPU no podrán importar la librería `face_recognition`.

2. Reconocimiento de rostros usando ML

Eres un agente que forma parte del famoso equipo de S.H.I.E.L.D. y te encuentras en Manhattan luchando contra Loki, en un intento por salvar el planeta Tierra! De pronto recuerdas que en la bodega de la base hay unos drones que sirven para reconocer caras, por lo que se te ocurre que los puedes usar para distinguir entre tus aliados los Vengadores, y tus enemigos los Chitauri. Lamentablemente te acuerdas de que no han sido actualizados en mucho tiempo, por lo que no serán capaces de reconocer las caras de los Vengadores actuales. Afortunadamente eso no es un impedimento para ti, ya que como experto en Machine Learning decides entrenar a estos drones para que sean capaces de reconocer las caras de tus aliados y así ganar la batalla contra Loki!

[4] https://colab. research.google.com/

Dataset

Los datos se encuentran disponibles acá. Estos corresponden a imágenes de las caras de los Vengadores. El dataset sigue la siguiente estructura de archivos:

1) La carpeta **Avengers** contiene las siguientes subcarpetas :

   1.1) **black_widow**

   1.2) **captain_america**

   1.3) **hawkeye**

   1.4) **hulk**

   1.5) **iron_man**

   1.6) **nick_fury**

   1.7) **thor**

   Cada una de estas carpetas contiene exclusivamente imágenes de las caras de su respectivo Vengador. Algo importante a tener en consideración, es que **la única forma de extraer la clase de la imagen es a través del nombre de la carpeta que la contiene**.

A continuación un ejemplo de como acceder a una imagen:

En una celda pimero corremos esto:

```
# el link donde se encuentra el dataset es:
# https://drive.google.com/file/d/1ZV_nCUfuitYz0vpKFhg5wdwz_LlaVQVo/view?usp=share_link
# por lo que usamos el id que corrsponde a
# 1ZV_nCUfuitYz0vpKFhg5wdwz_LlaVQVo para descargarlo

!gdown 1ZV_nCUfuitYz0vpKFhg5wdwz_LlaVQVo
!unzip 'Avengers 1.zip'

```

Luego, en otra celda esto:

```
from os import listdir
from os.path import join
import face_recognition as fr

root = "Avengers 1"
# Con este comando vemos todos los subdirectorios y archivos presentes en el directorio actual
lista_directorios = listdir(root)
# Con esto podemos obtener el nombre de la primera carpeta del directorio que corresponde a
# un nombre de un avenger, en este caso 'black_widow'
primera_carpeta = lista_directorios[0]

# Con esto tenemos la ruta donde se encuetran las fotos del primer Vengador
directorio_primer_avenger = join(root, primera_carpeta)

# Obtenemos una lista de los nombres de los archivos de las fotos presentes en la carpeta
lista_fotos_primer_avenger = listdir(directorio_primer_avenger)

primera_imagen = lista_fotos_primer_avenger[1]
ruta_imagen = join(directorio_primer_avenger, primera_imagen)

```

```
    # Cargamos el archivo usando la libreria face_recognition
    image = fr.load_image_file(ruta_imagen)

    # Le calculamos los landmarks a la imagen del Vengador
    face_landmarks_list = fr.face_landmarks(image)

    print(f"Clase del Vengador: {primera_carpeta} \nLandmarks del primer Vengador: {
    face_landmarks_list}")

```

Obteniendo:

```
    Clase del Vengador: captain_america
    Landmarks del primer Vengador: [{'chin': [(8, 31), (9, 43), (11, 55), (13, 66), (16, 77)
    , (21, 87), (29, 96), (38, 101), (49, 102), (61, 100), (71, 94), (79, 86), (84, 75),
    (87, 65), (89, 54), (91, 43), (93, 32)], 'left_eyebrow': [(17, 24), (23, 20), (30, 20),
    (38, 21), (46, 23)], 'right_eyebrow': [(57, 23), (63, 20), (71, 19), (78, 20), (84, 23)
    ], 'nose_bridge': [(51, 29), (51, 36), (50, 44), (51, 52)], 'nose_tip': [(42, 57), (46,
    59), (50, 60), (54, 59), (58, 57)], 'left_eye': [(25, 30), (30, 27), (35, 27), (40, 30),
     (35, 32), (29, 32)], 'right_eye': [(61, 30), (66, 27), (71, 27), (75, 29), (71, 31),
    (66, 32)], 'top_lip': [(36, 74), (41, 71), (46, 69), (51, 70), (55, 68), (60, 70), (65,
    74), (62, 74), (55, 73), (50, 73), (46, 73), (39, 74)], 'bottom_lip': [(65, 74), (60,
    78), (55, 80), (51, 80), (46, 80), (41, 78), (36, 74), (39, 74), (46, 74), (51, 74),
    (55, 73), (62, 74)]}]

```

Actividad 1: carga de datos y análisis (0,2 ptos.)

Lo primero que debes hacer es conocer los datos con los que se va a trabajar. Para eso carga correctamente los datos, y explica a qué tipo de datos corresponden. Luego realiza un análisis de la distribución de las clases en el dataset, y para cada clase muestra una imagen acompañada de su etiqueta.

Actividad 2: pre-procesamiento de los datos (0,3 ptos.)

Como estamos trabajando con caras, no queremos usar toda la imagen, sino que nos gustaría poder extraer características de cada una de ellas, para luego usar únicamente estas para entrenar nuestros modelos e identificar correctamente a cada uno de los Vengadores.

Para lograr lo anterior, te entregamos la librería **face_recognition**, la cual tiene una función llamada **fr.face_landmarks** (explicada anteriormente), que extrae landmarks de una cara, es decir, puntos de esta. En la figura 1 puede verse un ejemplo de su funcionamiento, donde si se reconoce una cara se extraen los siguientes landmarks.

En base a lo anterior, en esta sección de la tarea deberán obtener, para cada foto, sus respectivos landmarks. Muestra un ejemplo de la foto original y luego la foto con los landmarks incluidos. Repite esto con una imagen de cada clase. Por último, deberán manejar adecuadamente los casos en que no se obtienen landmarks, mostrando la cantidad de imágenes para las que estos no se obtuvieron. Adicionalmente, deberán decidir qué hacer con ellas de manera justificada, explicitando los motivos de cualquier modificación y/o eliminación. Finalmente, deberán mostrar la nueva distribución de las clases y analizar el balance de estas.

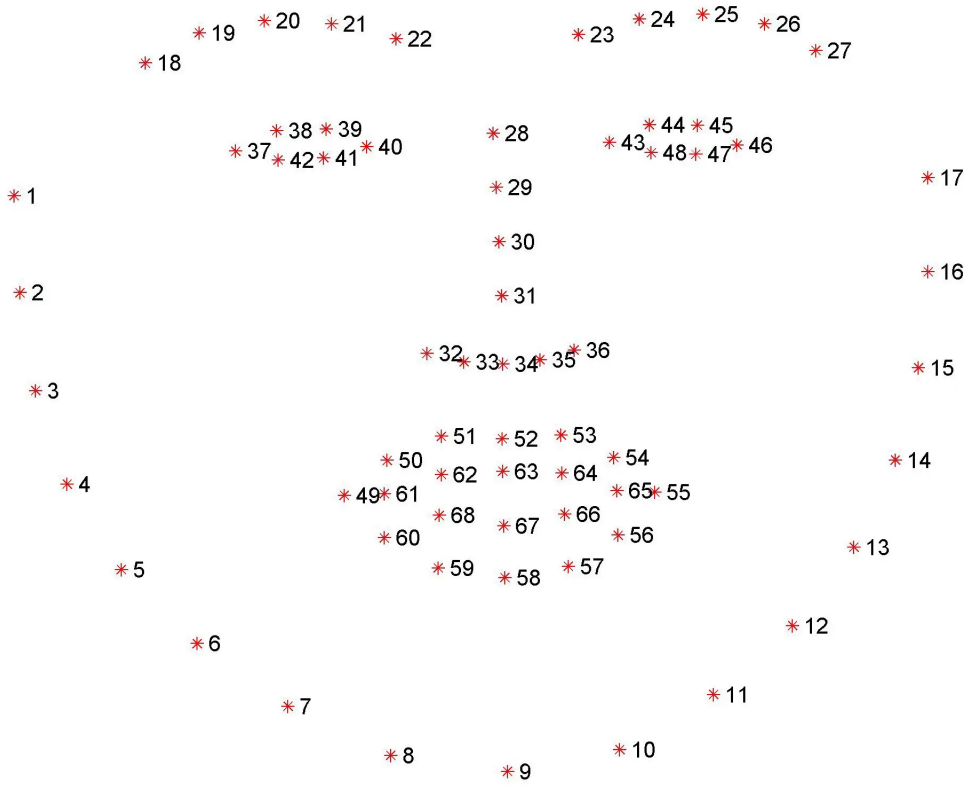

Figura 1: Modelo de puntos de fr.face_landmarks

Actividad 3: creando las features (1 pto.)

Ahora que ya tienes los landmarks de cada imagen, debes determinar como usarlos para poder clasificar correctamente cada cara. Para ello, debes crear las **features** (o características) con las que describirás cada imagen, a partir de estos landmarks. Recuerden que una buena elección de features es muy importante para poder entrenar correctamente a un modelo. Por lo tanto, deben diseñar **al menos 10 features** usando los landmarks, justificando correctamente cada una y su diseño, y explica que información de la imagen quieres codificar con ella.

Actividad 4: modelos de clasificación multiclase (1,5 ptos.)

Antes de entrenar los modelos, deberán dividir las imágenes en los conjuntos adecuados para la metodología de entrenamiento que utilicen, incluyendo una justificación para esto. Adicionalmente, deberán comentar sobre el balance de las clases y el efecto que este puede tener en el entrenamiento de los modelos.

Ahora que todo está listo, corresponde usar los modelos de clasificación para identificar a Los Vengadores. Para esto, utilizando `scikit learn`, deberán entrenar modelos de las siguientes familias:

1. Decision Tree
2. Random Forest
3. Gradient Boosting
4. KNN
5. SVM (con cualquier kernel)

Para cada una de estas familias, elijan aquel modelo que presente el mejor rendimiento de acuerdo a la metodología de entrenamiento y validación elegida. Para esto, no olviden variar los hiperparámetros más relevantes para cada familia. Para finalizar, reporte el rendimiento obtenido utilizando métricas como accuracy, recall, precision, f_1, etc, y discutan sobre cuales funcionan mejor o peor y su porqué.

IMPORTANTE: un mal uso de los sets de datos resultará en un descuento considerable ya que no obtendrán puntos por el rendimiento de tus modelos.

Actividad 5: matriz de confusión (1,5 ptos.)

Para cada uno de los modelos obtenidos, analice los resultados de su matriz de confusión asociada y responda preguntas como: ¿cuál es el error más común? ¿qué Vengador tiene el mejor rendimiento? ¿cuál es el más bajo?, ¿qué Vengadores son los que se confunden más comúnmente?, etc. Indiquen finalmente, para cada modelo, porqué creen que se dieron estos resultados, en base a las características particulares de estos.

Actividad 6: análisis de las features (1,5 ptos.)

Investigue sobre el método LIME para interpretar modelos de Machine Learning y utilícelo para analizar la importancia de las features para cada uno de los modelos elegidos. En particular, analice la existencia de features dominantes o si distintos modelos favorecen distintas features, si existen features adecuadas para clases o ejemplos específicos, etc. Acompañen su análisis con gráficos, tablas y/o visualizaciones que permitan facilitar la comprensión (p. ej., reducción de dimensionalidad).

Bonus: nuevas features (1 pto.)

A partir del análisis ya realizado, diseñen al menos 2 nuevas features (que no sean distancias) que puedan mejorar el rendimiento. Justifiquen la elección conceptual y empíricamente, esto último reentrenando al menos 3 de los modelos previamente encontrados y reportando las mejoras.

Comentarios al cierre

Como podrán notar, buena parte de esta tarea involucra respuestas de desarrollo escrito, donde deberán transparentar su razonamiento y explicar las decisiones tomadas. Por este motivo, para que una respuesta se considere correcta, deben tener cuidado de fundamentar apropiadamente lo que digas, aludiendo a referencias confiables y a la documentación de las librerías que utilicen. Por ejemplo, si les pedimos explicar un hiperparámetro en particular, o calcular una métrica de desempeño, se espera que demuestren un dominio general de lo que hace dicho hiperparámetro, o de la información que entrega la métrica solicitada. Para esto, es esperable que busquen y referencien los recursos (libros, artículos, documentación oficial, etc.) en los que se fundamentan las respuestas.

Al mismo tiempo, es posible que al intentar ejecutar operaciones costosas (como entrenar un modelo sobre un conjunto de datos), la ejecución sea más lenta de lo que esperan. Esto es un escenario cotidiano al trabajar con modelos de aprendizaje de máquina, de modo que se espera que sean capaz de lidiar con tales situaciones, y que transparenten y fundamenten las decisiones que tomadas.