



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

Facultad de Ingeniería

IIC2613 Inteligencia Artificial

2025-1

Tarea 4

Aprendizaje de Máquina

Fecha de entrega: Jueves 5 de Junio a las 23:59 hrs

Aspectos generales

Formato y plazo de entrega

El formato de entrega para la primera sección de la tarea (DCCopia) es **únicamente** un archivo con extensión .ipynb para tanto las respuestas de código como las teóricas. Para la segunda sección (Anti-DCCopia Emocional) debes entregar un archivo con extensión .ipynb para tanto las respuestas de código como las teóricas junto con un archivo con extensión .csv para el vector de predicciones. El lugar de entrega es en el repositorio de la tarea, en la branch por defecto, hasta el jueves 5 de junio a las 23:59 hrs. 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 **reales** (hábiles o no) 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, entendemos y motivamos el uso de material hecho por otras personas y por asistentes basados en IA, por lo que es importante reconocerlo de la forma apropiada. Todos los recursos externos que obtengas de internet debes citarlos 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. Asimismo, si utilizas información generada por inteligencia artificial, debes documentarlo según se detalla en el programa del curso.

Comentarios adicionales

El objetivo de esta tarea es que puedan utilizar distintas técnicas de aprendizaje de máquina, aplicándolas en problemas donde pueden ser de gran utilidad. Es fundamental que pongan énfasis en las justificaciones de sus respuestas, cuidando la redacción, ortografía; manteniendo el código ordenado y comentado. Aquellas respuestas que solo presenten resultados o código (sin contexto ni comentarios) no serán consideradas, mientras que tareas desordenadas pueden ser objeto de descuentos.

Recursos para realizar la tarea

Pandas

Esta librería 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
```

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**. El primero devuelve características (número de columnas, cuáles son y número de elementos no nulos), y 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 **isna**. Los elementos nulos aparecen con el denominativo **NaN**, y el método **isna** 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()
```

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**.

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"]]
```

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()
```

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, para la manipulación de archivos, existen dos tipos de métodos principales. Si lo que se desea es guardar un DataFrame procesado en un archivo **.csv**, se puede usar el método **to_csv**:

```
1 df.to_csv("archivo_de_destino.csv")
```

Por otro lado, si se quiere cargar un DataFrame desde un archivo **.csv**, se utiliza el método **read_csv**:

```
1 df.read_csv("archivo_por_cargar.csv")
```

Notar que existen métodos similares para otras extensiones de archivo.

Scikit-learn

Adicionalmente, usaremos la librera scikit-learn, la cual provee una gran gama de técnicas de aprendizaje de máquina, tal como clasificadores del tipo **SVM**, árboles de decisión, entre otros.

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
```

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

- Todos los clasificadores pueden ajustar sus hiperparámetros 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**.

Face Recognition

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

```
1 pip3 install face_recognition
```

Al importar la librería simplemente se debe hacer:

```
1 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 obtener un objeto imagen desde un archivo jpg.
- **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.

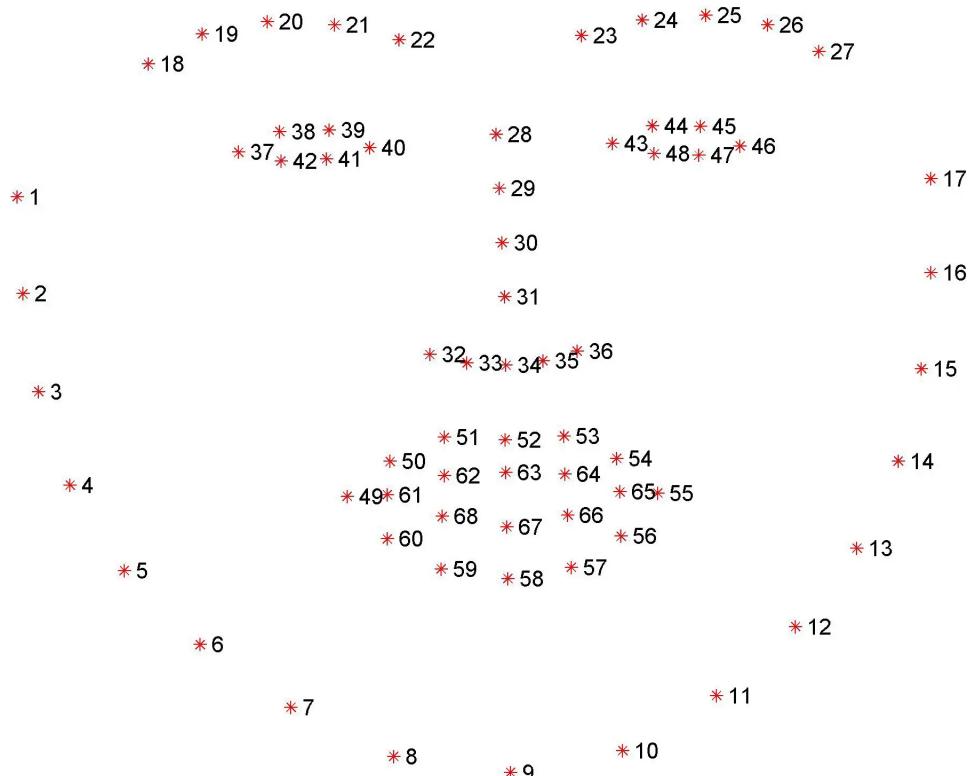


Figura 1: Modelo de puntos de fr.face_landmarks

A continuación un ejemplo de uso de esta librería:

```

1 import face_recognition as fr
2
3 ruta_imagen = "ruta_a_imagen.jpg"
4
5 # Cargamos el archivo usando la libreria face_recognition
6 image = fr.load_image_file(ruta_imagen)
7
8 # Le calculamos los landmarks a la imagen
9 face_landmarks_list = fr.face_landmarks(image)
10
11 print(f"Landmarks de la primera imagen: {face_landmarks_list}")

```

Obteniendo:

```
1 Landmarks de la primera imagen: [{'chin': [(8, 31), (9, 43), (11, 55), (13, 66),
2 (16, 77), (21, 87), (29, 96), (38, 101), (49, 102), (61, 100), (71, 94),
3 (79, 86), (84, 75), (87, 65), (89, 54), (91, 43), (93, 32)],  

4 'left_eyebrow
5 'right_eyebrow
6 'nose_bridgenose_tipleft_eyeright_eyetop_lipbottom_lip
```

Google Colab

Si queremos usar estos algoritmos de manera más rápida, podemos usar las famosas GPUs (tarjetas gráficas). Estas tarjetas 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 corresponde a Google Colab, por lo que recomendamos fuertemente desarrollar la tarea en esta plataforma, que además permite exportar el archivo como un archivo **.ipynb**, aunque no es requisito.

IMPORTANTE: Para poder activar el uso de GPU en Google Colab, deben ir a *Editar → Configuración del notebook → Acelerador de hardware → GPU*.

1. DCCopia (3 pts.)

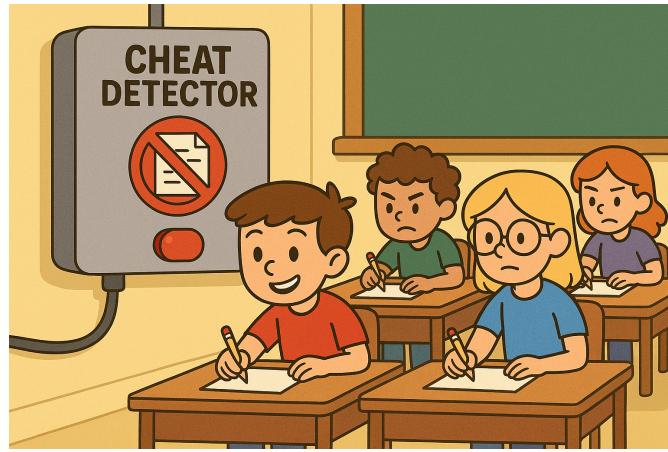


Figura 2: Cheat Detector activado en una sala de clases del DCC

Introducción

El profesor Hans Löbel, siempre atento a las estadísticas de su curso, notó algo peculiar al analizar los resultados de la última interrogación del curso *Inteligencia Artificial* del semestre pasado. Las notas no seguían la esperada y hermosa forma de campana de Gauss que tanto nos gusta a los ingenieros. Esta desviación del comportamiento normal levantó sospechas.

Intrigado por esta situación, el profesor encomendó a sus queridos y confiables alumnos una misión: **detectar posibles casos de copia mediante el análisis de datos anómalos.**

Objetivo

Deben construir un **sistema de Aprendizaje de Máquina, capaz de identificar qué estudiantes presentan un comportamiento anómalo** en sus datos, lo cual podría ser indicio de copia. Para esto, se entrega un dataset llamado `students.csv`, el cual deberán analizar y trabajar.

Dataset

El dataset entregado se llama `students.csv`, de tamaño 650x33, donde cada fila corresponde a las características de un alumno. Las columnas serán descritas a continuación:

school Escuela del estudiante.

sex Sexo del estudiante.

age Edad del estudiante.

address Tipo de dirección del hogar.

famsize Tamaño de la familia.

Pstatus Estado de convivencia de los padres.

Medu Nivel educativo de la madre.

Fedu Nivel educativo del padre.

Mjob Ocupación de la madre.

Fjob Ocupación del padre.

reason Razón para elegir la escuela.

guardian Tutor del estudiante.

traveltime Tiempo de viaje de casa a la escuela.

studytime Tiempo de estudio semanal.

failures Número de reprobaciones previas.

schoolsup Apoyo educativo extra en la escuela.

famsup Apoyo educativo familiar.

paid Clases particulares pagadas de la asignatura.

activities Actividades extracurriculares.

nursery Asistencia a jardín infantil.

higher Deseo de cursar educación superior.

internet Acceso a internet en el hogar.

romantic Relación amorosa actual.

famrel Calidad de las relaciones familiares.

freetime Tiempo libre después de clases.

goout Frecuencia de salidas con amigos.

Dalc Consumo de alcohol en días laborales.

Walc Consumo de alcohol los fines de semana.

health Estado de salud actual.

absences Número de ausencias escolares.

G1 Nota del primer periodo.

G2 Nota del segundo periodo.

G3 Nota final.

Cabe recalcar que este dataset **no contiene etiquetas identificando a los datos anómalos**, el objetivo es que puedan entrenar un modelo para que identifique las anomalías sin etiquetas.

Actividades

1.1. Pre-procesamiento datos (1 punto)

Para poder trabajar correctamente con los datos, debemos asegurarnos de que estén en el formato que se necesita. Para esto, lo primero que debemos hacer es una limpieza y transformación de algunos de ellos para que nuestros modelos puedan entenderlos. En esta sección debes hacer lo siguiente:

- a) Lee la información contenida en el archivo `students.csv`. Una vez que la información esté cargada, comenta sobre los elementos que contiene y su naturaleza. Para esto, responde al menos a estas preguntas (puedes plantear otras preguntas que guíen y expliquen el desarrollo de tu solución): ¿Cuántos datos hay? ¿Cuántos atributos hay y cuáles son? ¿A qué tipo de dato corresponde cada atributo?
Para entenderlos de mejor manera, es muy útil visualizar los datos con gráficos. Para esto te pediremos que hagas al menos un gráfico para la distribución de algún atributo que consideres relevante para el problema. Además, es necesario incluir un comentario que interprete los resultados obtenidos a partir de la o las visualizaciones realizadas.
- b) Luego debes identificar datos faltantes y decidir qué hacer con ellos. Asimismo, responde a la pregunta: ¿Son todos los atributos útiles para esta tarea? Justifica, y decide qué hacer con ellos. Recuerda que puedes normalizar o estandarizar las características según sea necesario.
Además, contesta la pregunta ¿Qué columnas o atributos son los más relevantes para la detección de copia? Justificar utilizando métricas a tu elección.
- c) Para terminar el preprocesamiento debes convertir variables categóricas a numéricas. Primero responde a la siguiente pregunta: ¿Por qué debemos hacer esta conversión? ¿Cuáles son las columnas que debemos convertir? Luego decide qué método utilizarás para cada una y justifica.

1.2. Entrenamiento y evaluación del modelo (1 punto)

A continuación, deberás investigar sobre cómo realizar la identificación de datos anómalos sin tener etiquetas que los identifiquen como tales, y realizar las siguientes actividades:

- a) Investiga dos modelos diferentes que sean adecuados para este problema, dentro de los que se hayan visto en el curso. Describe cómo funcionan, justifica su uso para esta tarea, y describe los hiperparámetros principales.
- b) Indica cómo medirás el rendimiento de los modelos. Para esto, investiga sobre métricas de rendimiento adecuadas al problema y cómo deben subdividirse los datos en este caso. Explica cuáles usarás y por qué.
- c) Entrena ambos modelos investigados. Prueba diferentes combinaciones de hiperparámetros para cada uno, justificando las elecciones. Considera la naturaleza de los datos y los resultados obtenidos.
- d) Muestra algunos datos que tu modelo identificó como anómalos y comenta cuál podría ser la anomalía en cada caso. ¿Consideras estos datos como anomalías? ¿Por qué? Realiza esto para ambos modelos.
- e) Muestra la proporción de datos identificados como anómalos y no anómalos de cada modelo y comente.

IMPORTANTE - RESTRICCIONES: No está permitido el uso de redes neuronales ni modelos complejos tipo black-box. Tampoco se pueden utilizar *embeddings* para la extracción de características del texto.

1.3. Análisis de Resultados (1 punto)

Una vez entrenados tus modelos e identificados los datos anómalos o copias, responde las siguientes preguntas:

- a) Habiendo analizado los datos identificados como anómalos, ¿que características fueron las más relevantes según cada modelo para identificar las anomalías? ¿Qué caracteriza a los datos anómalos según cada modelo? ¿Son estas diferentes a las pensadas inicialmente? ¿Por qué? Responder para cada modelo.
- b) ¿Qué modelo es mejor para identificar los datos anómalos? Justifica utilizando conclusiones y resultados obtenidos.
- c) ¿Qué tipo de error es menos dañino en este caso, ¿acusar a un inocente o no acusar a un culpable? Tomando esto en consideración, ¿Son estos modelos una buena opción para identificar las anomalías dadas por las copias? ¿Por qué?

Evaluación

Lo más importante en esta tarea es como enfrentan el problema y plantea su solución. Debido a eso, es necesario que comenten tanto el código utilizado como las decisiones y conclusiones a las que lleguen, ya que esto será lo más importante para la evaluación. Cualquier código y/o decisión no justificada no tendrá puntaje.

Entregables

Para esta actividad se deberá entregar un archivo `dccopia.ipynb` en el repositorio **con todas las celdas ejecutadas**. Dicho notebook deberá poder funcionar al momento de la corrección para obtener el puntaje. Aquí debes incluir tanto el código como las respuestas teóricas. Recuerda que debes justificar tus decisiones y explicar tu procedimiento, para cada una de las actividades.

Importante: No debes subir los datos que utilices a tu repositorio de github, de lo contrario habrá un penalización importante en tu nota.

2. DCCaras (3 ptos.)



Figura 3: Cheat Detector avanzado, listo para detectar emociones

Introducción

No contento con su nueva detección de datos anómalos, el profesor Hans Löbel quiso ir más allá. Ahora, quiere ser capaz de identificar casos de trampa **durante** la realización de las pruebas, para asegurar honestidad entre quienes la rinden. Para ello, tomará en cuenta las expresiones faciales de los estudiantes y las emociones que expresan en el transcurso de la evaluación, pues teoriza que puede existir una relación con la ejecución de prácticas fraudulentas.

De esta forma, el profesor decide encargarte una nueva misión: **identificar las emociones de los alumnos, a través de un análisis facial de imágenes.**

Objetivo

Debes crear un **modelo capaz de clasificar la emoción presente en una imagen del rostro de una persona**. Se requiere que pueda clasificar entre 7 posibles categorías: enojo, desagrado, miedo, felicidad, neutral, tristeza y sorpresa.

Archivos entregados

Se te entrega el dataset `faces.zip`, el cual contiene dos directorios. El primero de ellos, llamado `Validacion`, contiene imágenes **etiquetadas** de las 7 emociones descritas anteriormente. Cada una de ellas sigue la forma: `{emoción}_{id}.jpg` El segundo llamado `Test` contiene imágenes **no etiquetadas** de las 7 emociones. Cada una de ellas sigue la forma: `img_{id}.jpg`.

Estos corresponden a sets de validación y testeo respectivamente. Ambos sets provienen de las mismas fuentes, por lo que el set de validación se te entrega para que tengas una noción del formato de los datos (puedes utilizar este set de datos como estimes conveniente). Posteriormente se explicará el uso del set de testeo.

Todas las imágenes están en formato RGB (tres canales) y tienen dimensiones de 96×96 píxeles. Esto **no** implica que tu modelo deba utilizar exactamente este formato; si tu pipeline requiere una forma distinta, deberás realizar las transformaciones necesarias para que el modelo pueda procesarlas correctamente.

Actividades

2.1. Recopilación de datos (0.75 puntos)

En primer lugar, debes conseguir suficientes datos (imágenes) con los cuales entrenarás y testearás tus modelos de *Machine Learning*. Para ello, existen varios recursos de donde las puedes obtener, pero recuerda el objetivo que buscas conseguir.

En esta parte, deberás:

- a) Recopilar recursos suficientes, justificando porqué elegiste esa cantidad e indicar de dónde se obtuvieron. También, remarcar cualquier consideración/transformación que realizaste sobre los datos originales, de haberlo hecho.
- b) Una vez cargados, deberás graficar la distribución de datos para cada una de las clases que queremos predecir y mostrar la cantidad de datos únicos que existen.
- c) Obtener las características (features) que usaras para resolver la tarea. El único requisito es que uses como mínimo *landmarks* faciales¹ de la librería `face_recognition` explicada anteriormente. Analiza como utilizarlas para clasificar correctamente las emociones. Por otro lado, puedes usar cualquier otra característica que estimes conveniente, mientras cumpla con la restricción detallada más abajo en la sección de restricciones.
- d) Dividir el dataset en un conjunto de entrenamiento y uno de prueba. Justifica tu elección de las proporciones de cada una.

2.2. Implementación de algoritmos de *Machine Learning* (0.75 puntos)

En segundo lugar, deberás elegir al menos dos modelos predictivos de aprendizaje de máquina vistos en clases y entrenarlos para resolver esta tarea. Recuerda que, según el que escojas, habrán distintos hiperparámetros que manejar, por lo que considera que posteriormente debes ajustar sus valores según el rendimiento que busques.

Estos modelos pueden recibir las imágenes como mejor estimes conveniente, ya sea en cuanto a formato, dimensiones, etc. Solo ten en cuenta ser consistente.

IMPORTANTE - RESTRICCIONES: No está permitido el uso de redes neuronales ni modelos tipo *black-box*. Tampoco se pueden utilizar *embeddings* para la extracción de características de las imágenes.

2.3. Evaluación del modelo (0.75 puntos)

En tercer lugar, debes utilizar las siguientes métricas (se recomienda utilizar las implementaciones disponibles en las librerías). Explica brevemente qué indica cada una de estas métricas, de forma general:

- a) *Accuracy*
- b) *Precision*

Luego deberás investigar dos más que consideres relevantes para medir el rendimiento de tu modelo. Asimismo, explica brevemente qué indica cada una de estas métricas.

Estas métricas de rendimiento te servirán para ir adaptando tu modelo y mejorándolo, por lo que a continuación deberás calcúlalas para ambos modelos en el set de datos que corresponda. Segundo estos resultados decide que cambios realizaras en tus modelos.

¹Más información: <https://pypi.org/project/face-recognition/>

2.4. Predicción final (0.75 puntos)

Por último, deberás utilizar el contenido del directorio `face.zip` (explicado anteriormente) para evaluar el desempeño de tu modelo entrenado.

Primero, deberás clasificar las imágenes del directorio `Validación` utilizando tu modelo previamente entrenado. Para que esta actividad se considere completada, tu modelo debe alcanzar al menos un 70% de *accuracy* en este.

Luego, deberás cargar todas las imágenes del set de `Test`, clasificarlas utilizando tu modelo previamente entrenado y generar un vector de predicciones para estas. Este debe seguir el siguiente formato:

Formato del vector de predicciones:

Un archivo `predicciones.csv` que contenga las predicciones generadas por tu modelo para cada una de las imágenes dentro de la carpeta `Test`. El archivo debe tener dos columnas: `id_imagen` y `prediccion`, donde:

- `id_imagen` corresponde **únicamente al id** del archivo de la imagen. Debe estar ordenado en orden **descendiente** según venían originalmente.
- `prediccion` corresponde al *label* asignado por tu modelo a la imagen respectiva.

El archivo debe seguir el siguiente formato, sin encabezado:

```
id0_imagen, prediccion  
id1_imagen, prediccion  
id2_imagen, prediccion
```

Es importante destacar que las imágenes de `Test` deben utilizarse exclusivamente para evaluación. Se revisará explícitamente que **no** hayan sido utilizadas durante el entrenamiento, ya que esto comprometería la validez de los resultados.

2.5. Bonus

Para esta parte, se realizará un podio con los modelos con mayor exactitud (*accuracy*) de la sección en el set de `Test`. Esto se realizará utilizando el vector de predicciones.

De esta forma, el puntaje de bonus se distribuirá de la siguiente forma, según el lugar del podio:

1. **1^{er} lugar:** 5 décimas
2. **2^{do} lugar:** 4 décimas
3. **3^{er} lugar:** 3 décimas
4. **4^{to} lugar:** 2 décimas
5. **5^{to} lugar:** 1 décima.

Evaluación

Lo más importante en esta tarea es como enfrentan el problema y plantea su solución. Debido a eso, es necesario que comenten tanto el código utilizado como las decisiones y conclusiones a las que lleguen, ya que esto será lo más importante para la evaluación. Cualquier código y/o decisión no justificada no tendrá puntaje.

Entregables

Para esta actividad se deberá entregar un archivo `dccaras.ipynb` en el repositorio con **todas las celdas ejecutadas**. Dicho notebook deberá poder funcionar al momento de la corrección para obtener el puntaje. Aquí debes incluir tanto el código como las respuestas teóricas. Recuerda que debes justificar tus decisiones y explicar tu procedimiento, para cada una de las actividades.

Asimismo, deberás entregar tu vector de predicciones siguiendo el formato explicado anteriormente en la **Actividad 4**.

Importante: No debes subir los datos que utilices a tu repositorio de github, de lo contrario habrá un penalización importante en tu nota.