CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

APRENDIZAJE INTELIGENTE

6° "A"

PROYECTO

Profesor: Francisco Javier Luna Rosas

Alumnos:

Espinoza Sánchez Joel Alejandro

Gómez Garza Dariana

González Arenas Fernando Francisco

Fecha de Entrega: Aguascalientes, Ags.**, 2** de junio de 2021

Proyecto

Introducción

Como ya sabemos inteligencia artificial es la combinación de algoritmos planteados con el propósito de crear máquinas que presenten las mismas capacidades que el ser humano. El machine learning es una rama de la computación de la IA que se basa en el entrenamiento de algoritmos de aprendizaje automático a partir de datos anteriores.

Esos tipos de aprendizajes se dividen en supervisados, no supervisados, semi-supervisado y por refuerzo; pero en esta ocasión nos centraremos en el aprendizaje supervisado.

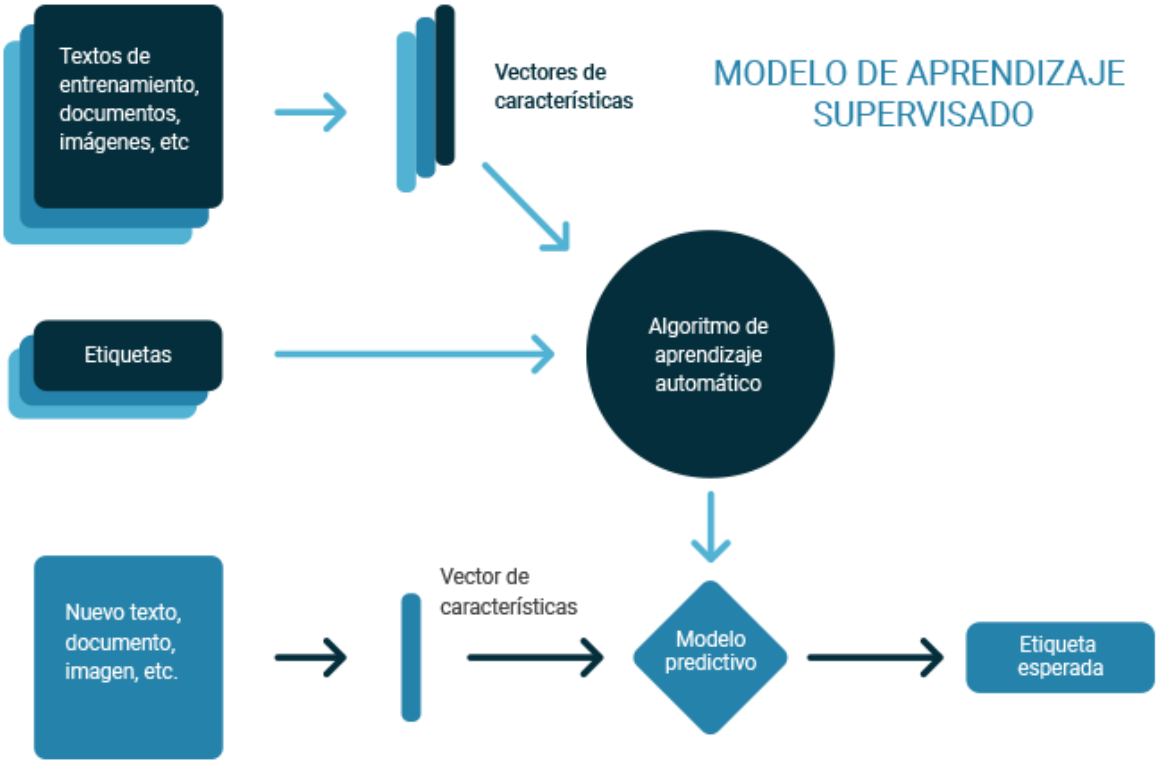
El aprendizaje supervisado necesita conjuntos de datos etiquetados, es decir, le decimos al modelo qué es lo que queremos que aprenda. Dependiendo del tipo de etiqueta, dentro del aprendizaje supervisado existen dos tipos de modelos:

- **Los modelos de clasificación**, que producen como salida una etiqueta discreta, es decir, una etiqueta dentro de un conjunto finito de etiquetas posibles. A su vez, los modelos de clasificación pueden ser binarios si tenemos que predecir entre dos clases o etiquetas (enfermedad o no enfermedad, clasificación de correos electrónicos como “spam” o no “spam”) o multiclase, cuando se tiene que clasificar más de dos clases (clasificación de imágenes de animales, análisis de sentimientos, etc.).

- **Los modelos de regresión** producen como salida un valor real.

En los algoritmos de aprendizaje supervisado se genera un modelo predictivo, basado en datos de entrada y salida. La palabra clave “supervisado” viene de la idea de tener un conjunto de datos previamente etiquetado y clasificado, es decir, tener un conjunto de muestra, el cual ya se sabe a qué grupo, valor o categoría pertenecen los ejemplos. Con este grupo de datos que llamamos datos de entrenamiento, se realiza el ajuste al modelo inicial planteado. Es de esta forma como el algoritmo va “aprendiendo” a clasificar las muestras de entrada comparando

el resultado del modelo, y la etiqueta real de la muestra, realizando las compensaciones respectivas al modelo de acuerdo con cada error en la estimación del resultado.



Existen muchos algoritmos de aprendizaje supervisado que nos ayudan a clasificar, por ejemplo:

- Árboles de decisión
- Clasificación de Naïve Bayes
- Regresión por mínimos cuadrados
- Regresión Logística
- Support Vector Machines (SVM)
- Métodos “Ensemble” (Conjuntos de clasificadores).

Hablando un poco del modelo Support Vector Machine, es o, mejor dicho, son un conjunto de algoritmos de aprendizaje supervisado desarrollados por Vladimir Vapnik y su equipo en los laboratorios AT&T. Que funciona a grandes rasgos de la siguiente manera; Dado un conjunto de puntos, subconjunto de un conjunto mayor

(espacio), en el que cada uno de ellos pertenece a una de dos posibles categorías, un algoritmo basado en SVM construye un modelo capaz de predecir si un punto nuevo (cuya categoría desconocemos) pertenece a una categoría o a la otra.

Como en la mayoría de los métodos de clasificación supervisada, los datos de entrada (los puntos) son vistos como un vector p-dimensional (una lista ordenada de p números).

La SVM busca un hiperplano que separe de forma óptima a los puntos de una clase de la de otra, que eventualmente han podido ser previamente proyectados a un espacio de dimensionalidad superior.

En ese concepto de "separación óptima" es donde reside la característica fundamental de las SVM: este tipo de algoritmos buscan el hiperplano que tenga la máxima distancia (margen) con los puntos que estén más cerca de él mismo. Por eso también a veces se les conoce a las SVM como clasificadores de margen máximo. De esta forma, los puntos del vector que son etiquetados con una categoría estarán a un lado del hiperplano y los casos que se encuentren en la otra categoría estarán al otro lado.

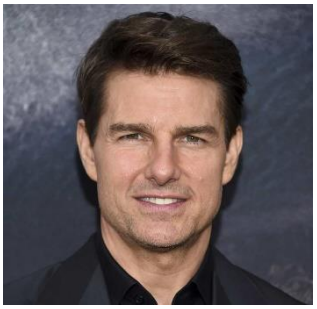
- **Los algoritmos SVM pertenecen a la familia de los clasificadores lineales.**

En nuestro proyecto utilizamos este algoritmo para el reconocimiento facial, específicamente para clasificar hombres con diferentes edades y agrupándolos en 5 conjuntos: bebés, niños, jóvenes, adultos y adultos mayores. Por ejemplo, en base a estas imágenes (15 en total en cada grupo) el algoritmo deberá etiquetar estas imágenes de la siguiente forma:

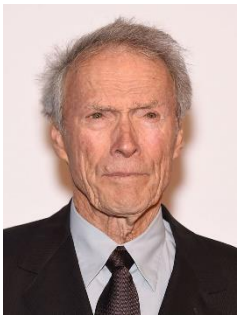
Bebé

Niño

Joven



Adulto



Anciano

Procedimiento

El primer paso dentro del desarrollo del clasificador que se desarrolló es importar las librerías necesarias para el desarrollo de este modelo:

```
#%% Importamos las librerías
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import pickle
import random
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
```

Posteriormente se declara el directorio en el que se encuentra el conjunto de imágenes y donde se escribirán algunos documentos de entrenamiento y prueba:

```
#%% Declaramos el directorio donde están las imágenes de prueba
dir = 'C:\\Users\\alexe\\Desktop\\Proyecto2\\images'
```

Luego se debe determinar en el programa las categorías de clasificación para el modelo:

```
#%% Declaramos las categorías
categories = ['bebe', 'nino', 'joven', 'adulto', 'anciano']
```

A continuación, se revisará por categoría cada imagen de la carpeta señalada y será extraída a un formato de tipo arreglo después de estandarizarlas a una dimensión

de 100 píxeles de altura y anchura en donde todo se guardará en una variable denominada como `data`:

```
#%% Se revisa por categoría cada imagen del directorio
for category in categories:
    path = os.path.join(dir, category)
    label = categories.index(category)

    for img in os.listdir(path):
        imgpath = os.path.join(path,img)
        face_img = cv2.imread(imgpath,0)
        try:
            face_img = cv2.resize(face_img, (50,50))
            image = np.array(face_img).flatten()

            data.append([image,label])
        except Exception as e: #Exepción de error en resolución de imagen
            pass
```

Ahora, para proceder al procesamiento se deberá guardar esta información en un archivo. En este caso se le llamó `data1.pickle`:

```
#%% Se crea el documento data1 en el mismo directorio de archivos con el análisis SVM
pick_in = open('data1.pickle', 'wb')
pickle.dump(data, pick_in)
pick_in.close()

#%% Se carga la información del archivo data1 en la variable data1
pick_in = open('data1.pickle', 'rb')
data1 = pickle.load(pick_in)
pick_in.close()
```

El siguiente paso mezclará los individuos de la variable data1 y de la cual se procederá a tomar las características de cada individuo para próximamente ponderarlas en el método que aplicará el algoritmo Support Vector Machine.

```
#%% Se elige un individuo al azar
random.shuffle(data1)
features1 = []
label1 = []

#%% Se llena el arreglo de características de cada individuo
for feature, labelint in data1:
    features1.append(feature)
    label1.append(labelint)
```

A continuación se dividirá la población en dos secciones, siendo una porción para el entrenamiento y otra para la prueba que se hará con la función `train_test_split` que dividirá a las variables `xtrain`, `xtest`, `ytrain` y `ytest` con un porcentaje de población para prueba determinado por la variable `test_size`, en este caso un 20% y el resto para realizar el entrenamiento y posteriormente se realiza el algoritmo Support Vector Machine bajo una combinación lineal de cada píxel de las imágenes reducidas a un tamaño estándar para todas las imágenes.

```
#%% Se realiza la clasificación a partir del entrenamiento usando 75% de la población
xtrain, xtest, ytrain, ytest = train_test_split(features1, label1, test_size = 0.20)

#%% Se aplica el algoritmo SVC de la librería sklearn con los datos previamente entren
model = SVC(C = 1, kernel = 'poly', gamma = 'auto')
model.fit(xtrain, ytrain)
```

El siguiente paso consiste en la creación de un documento denominado `model.sav` que guardará los datos de entrenamiento y posteriormente se carga en el procedimiento.

```
#%% Se crea el documento model con el cual se pasa a la predicción
pick = open('model.sav', 'wb')
pickle.dump(model, pick)
pick.close()

#%% Una vez creado, se carga el documento model a la variable model
pick = open('model.sav', 'rb')
model = pickle.load(pick)
pick.close()
```

Finalmente se realizará la predicción y se calculará la precisión de la predicción por medio de las variables de prueba para finalizar el procedimiento de Support Vector Machine:

```
#%% Se calcula la predicción y la precisión del model
prediction = model.predict(xtest)
accuracy = model.score(xtest, ytest)
```

Por último en el código se imprimen los resultados para tener el conocimiento de los productos realizados al finalizar el procedimiento.

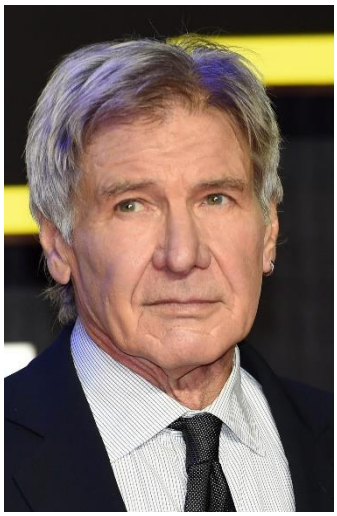
```
#%% Impresión de los resultados en texto
print(accuracy)
print(categories[prediction[0]])

#%% Impresión de la imagen usada y los píxeles tomados a consideración
my_boy = xtest[0].reshape(50,50)
plt.imshow(my_boy, cmap = 'gray')
plt.show()
```

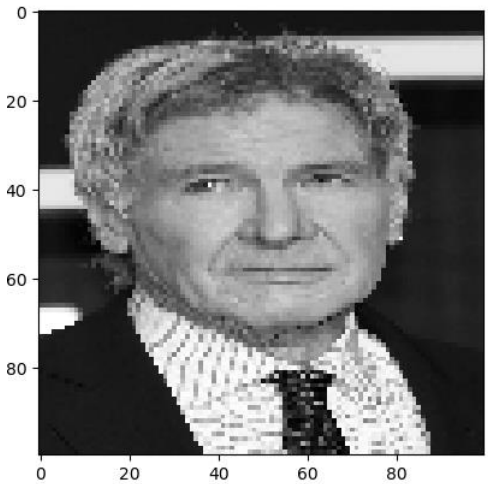
Obtención y Procesamiento de Datos

Pueden observarse los resultados del procedimiento a continuación.

Primeramente, las imágenes se estandarizan a un tamaño y dimensiones para que sean de un mismo análisis. Así puede observarse de ejemplo la siguiente imagen:

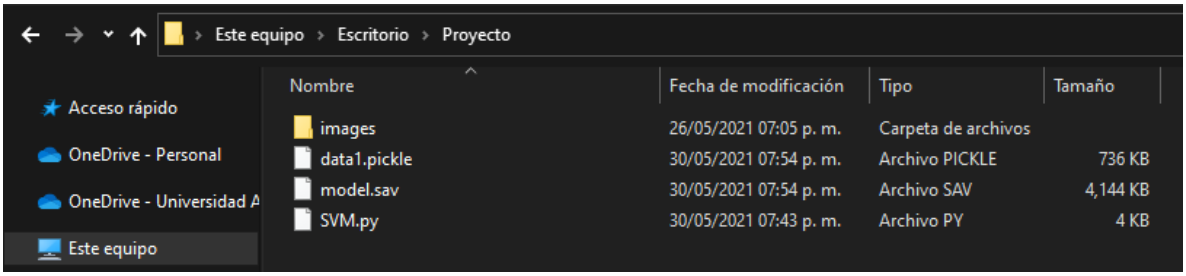


Que se estandariza de la manera que se presenta a continuación:



Esto ocurre con todas las imágenes, guardándose como un arreglo numérico.

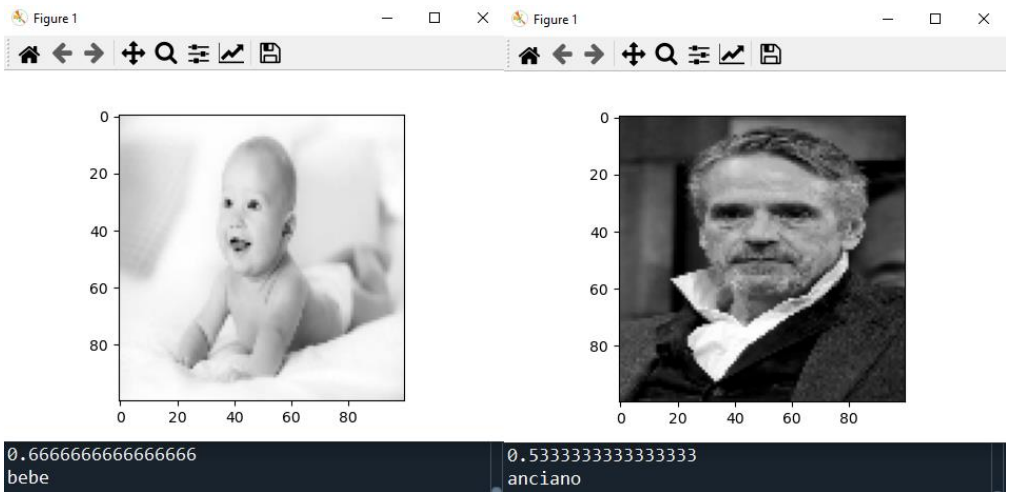
La población modificada a un arreglo se guarda en un documento llamado `data1.pickle`:



Este archivo se vuelve a cargar para llevarse a las variables de entrenamiento y prueba, que posteriormente serán las variables de entrada al método Support Vector Machine. El modelo realizado se cargará en el archivo `model.sav` que también se observa guardado en el directorio.

El algoritmo procede a realizar Support Vector Machine por cada píxel dado y separa la población en prueba y entrenamiento, de modo que al finalizar el entrenamiento (cuando se crean los dos archivos mostrados anteriormente).

Una vez que el programa elija aleatoriamente a un individuo de los escogidos para la prueba, lo analizará en función de los individuos separados para entrenamiento. El programa mostrará el individuo elegido junto con la predicción del individuo y su precisión correspondiente como pueden observarse algunos ejemplos a continuación:



Como última aclaración, el conjunto de imágenes de evaluación se cambió posteriormente a la redacción del presente informe para mejorar la calidad de la predicción de la máquina desarrollada.

Aclaración de Características

Se aclararán las siguientes características:

a) **Dataset.**

   El conjunto de datos utilizado es un grupo de imágenes presentado como evidencia en el entregable. Las variables predictoras consisten en cada píxel del rostro y con base en ellas, la única variable a predecir es el rango de edad del individuo de prueba al que pertenece, a saber, bebé, niño, joven, adulto o anciano.

b) **Tipo de algoritmos que utilizan.**

   1) **Support Vector Machine:**

      El Support Vector Machine trabaja por medio de combinaciones lineales de todas las coordenadas de cada píxel de las imágenes introducidas.

   2) **Modelo Naive Bayes:**

      El modelo Naive Bayes trabaja bajo una probabilidad condicional usando la regla de Bayes, pues en nuestro caso compara la probabilidad de que un píxel posea cierto valor dado que todos los píxeles del conjunto de una categoría poseen otro valor determinado. El análisis se realiza análogamente para cada píxel

   3) **Random Forest:**

      El Random Forest trabaja bajo la creación de múltiples árboles de decisión para el entrenamiento del modelo. En nuestro caso práctico, el modelo toma una muestra de un número de casos con reemplazo y será el conjunto de construcción del árbol. Hay diferentes variables de entrada y de éstas, se trata de realizar la mejor división de atributos para ramificar el árbol y se mantiene constante este parámetro durante la generación de todo el bosque.

c) Tipo de entrenamiento que se utiliza.

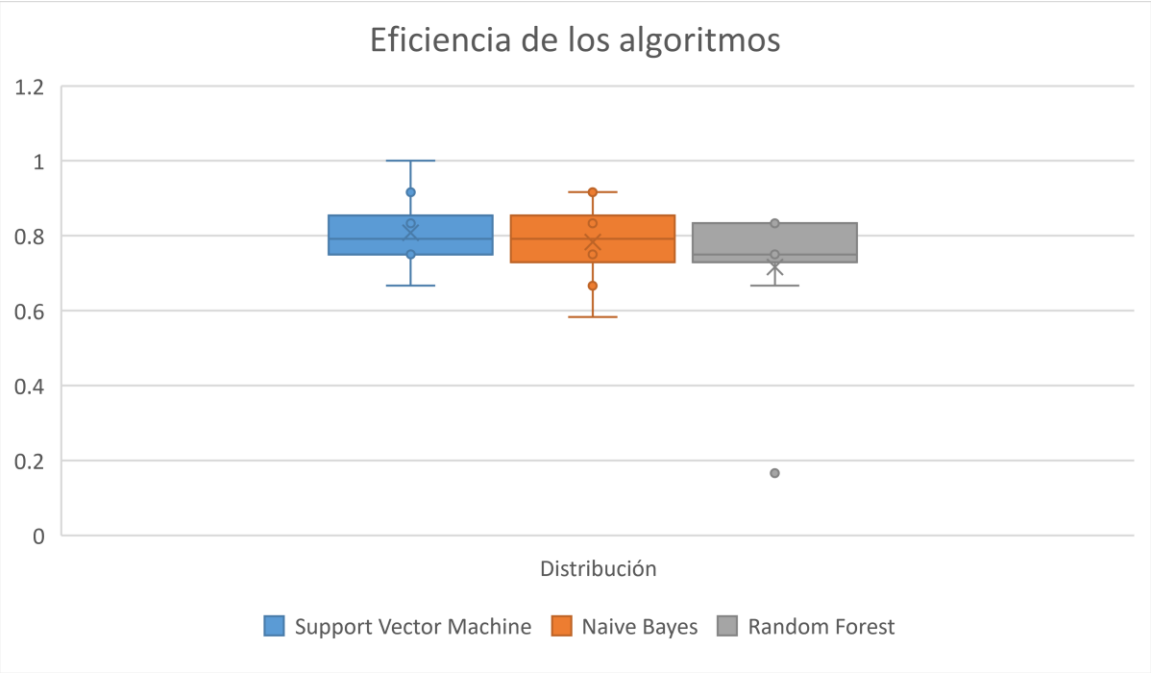
Este entrenamiento pertenece a la clasificación de aprendizaje supervisado. El entrenamiento utilizado es el Support Vector Machine por medio de combinaciones lineales de todas las coordenadas de cada píxel de las imágenes introducidas. Este entrenamiento pertenece a la clasificación de aprendizaje supervisado.

d) Análisis comparativo de máquinas de Machine Learning.

Se realizó una prueba de efectividad en la que se ejecutaban los algoritmos diez veces y se realizó un promedio de la precisión obtenida. Los resultados se muestran a continuación:

Modelo	Support Vector Machine	Naive Bayes	Random Forest
Ejecución 1	0.75	0.75	0.8333
Ejecución 2	0.9166	0.75	0.8333
Ejecución 3	0.75	0.8333	0.8333
Ejecución 4	0.75	0.8333	0.6666
Ejecución 5	0.8333	0.6666	0.75
Ejecución 6	0.6666	0.5833	0.75
Ejecución 7	0.8333	0.9166	0.8333
Ejecución 8	0.8333	0.8333	0.75
Ejecución 9	1	0.75	0.75
Ejecución 10	0.75	0.9166	0.1666
Promedio	0.8083	0.7833	0.7166

Observemos este grupo de la siguiente manera:



Conclusiones

Joel Alejandro Espinoza Sánchez: Gracias al proyecto, nos permitimos explorar el planteamiento de un problema de aprendizaje desde la selección del algoritmo, ya que no todos los algoritmos están diseñados para la solución de una misma tarea. Su diversidad permite la solución de problemas orientados a distintos tipos de datos, predicciones y objetivos.

Pude orientar personalmente un enfoque de investigación para proponer el mejor algoritmo con base en lo que deseábamos predecir y el conjunto de datos con lo que realizaríamos el procedimiento.

Dariana Gómez Garza: En este proyecto final pudimos agrandar el conocimiento que teníamos en aprendizaje supervisado, ya que nunca habíamos hecho una inteligencia artificial a base de imágenes en el curso.

Es interesante la amplia gama de opciones que hay para realizar este tipo de programas, por ejemplo, encontramos mucha información sobre Tensorflow y cómo era más sencillo clasificarlo así en lugar de realizarlo con el aprendizaje supervisado; pero el punto de este último parcial era tratar de realizarlo con este tema en lugar de redes neuronales.

Me gustó mucho como quedó nuestro proyecto final y también era muy interesante como el algoritmo detectaba las imágenes y las clasificaba.

Fernando Francisco González Arenas: El reconocimiento de rostros por medio de aprendizaje supervisado es una técnica muy útil que tiene múltiples usos en la vida diaria de las personas alrededor del mundo en la actualidad, se puede usar para fines de seguridad, clasificación de grupos de población, controles de autenticación, detectar emociones y sentimientos de las personas, etc.

Con la realización de esta práctica investigamos formas de detectar rostros en las imágenes y clasificar a las personas por edades, lo cual puede tener muchas aplicaciones prácticas en el futuro, incluso para futuros proyectos de los integrantes de este mismo equipo.

Referencias

- Igual, L. (2017). *Introduction to Data Science*. Barcelona: Springer.
- Rebala, G. (2019). *An Introduction to Machine Learning*. California: Springer.
- Sarkar, D. (2018). *Practical Machine Learning with Python*. Chicago: Apress.
- Unpingco, J. (2019). *Python for Probability, Statistics and Machine Learning*. California: Springer.

Anexos

Anexo 1: Código del clasificador SVM orientado a la predicción de edad en Python

```
#%% Presentación
'''
   Universidad Autónoma de Aguascalientes

          Centro de Ciencias Básicas
 Departamento de Ciencias de la Computación
           Aprendizaje Inteligente
                    6° "A"

                  Práctica 5

   Profesor: Francisco Javier Luna Rosas
  Alumnos:
      Espinoza Sánchez Joel Alejandro
      Gómez Garza Dariana
```

```
      González Arenas Fernando Francisco

  Fecha de Entrega: 7 de abril del 2021

Descripción: SVM aplicado
'''

#%% Importamos las librerías
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import pickle
import random
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

#%% Declaramos el directorio donde están las imágenes de prueba
dir = 'C:\\Users\\alexe\\Desktop\\Proyecto2\\images'

#%% Declaramos las categorías
categories = ['bebe', 'nino', 'joven', 'adulto', 'anciano']

#%% Declaramos la variable que mantendrá los datos a lo largo del
procedimiento SVM
data = []

#%% Se revisa por categoría cada imagen del directorio
for category in categories:
    path = os.path.join(dir, category)
    label = categories.index(category)

    for img in os.listdir(path):
        imgpath = os.path.join(path,img)
        face_img = cv2.imread(imgpath,0)
        try:
            face_img = cv2.resize(face_img, (50,50))
            image = np.array(face_img).flatten()

            data.append([image,label])
        except Exception as e: #Exepción de error en resolución de imagen
            pass

#%% Se crea el documento data1 en el mismo directorio de archivos con el
análisis SVM
pick_in = open('data1.pickle', 'wb')
pickle.dump(data, pick_in)
pick_in.close()

#%% Se carga la información del archivo data1 enla variable data1
```

```
pick_in = open('data1.pickle', 'rb')
data1 = pickle.load(pick_in)
pick_in.close()

#%% Se elige un individuo al azar
random.shuffle(data1)
features1 = []
label1 = []

#%% Se llena el arreglo de características de cada individuo
for feature, labelint in data1:
    features1.append(feature)
    label1.append(labelint)

#%% Se realiza la clasificación a partir del entrenamiento usando 75% de
la población
xtrain, xtest, ytrain, ytest = train_test_split(features1, label1,
test_size = 0.75)

#%% Se aplica el algoritmo SVC de la librería sklearn con los datos
previamente entrenados
model = SVC(C = 1, kernel = 'poly', gamma = 'auto')
model.fit(xtrain, ytrain)

#%% Se crea el documento model con el cual se pasa a la predicción
pick = open('model.sav', 'wb')
pickle.dump(model, pick)
pick.close()

#%% Una vez creado, se carga el documento model a la variable model
pick = open('model.sav', 'rb')
model = pickle.load(pick)
pick.close()

#%% Se calcula la predicción y la precisión del model
prediction = model.predict(xtest)
accuracy = model.score(xtest, ytest)

#%% Definición de las categorías (apartado repetido)
categories = ['bebe', 'nino', 'joven', 'adulto', 'anciano']

#%% Impresión de los resultados en texto
print(accuracy)
print(categories[prediction[0]])

#%% Impresión de la imagen usada y los píxeles tomados a consideración
my_boy = xtest[0].reshape(50,50)
plt.imshow(my_boy, cmap = 'gray')
plt.show()
```

Anexo 2: Código del clasificador Naive Bayes orientado a la predicción de edad en Python

```
#%% PresentaciÃ³n
'''
   Universidad AutÃ³noma de Aguascalientes

         Centro de Ciencias BÃ¡sicas
 Departamento de Ciencias de la ComputaciÃ³n
          Aprendizaje Inteligente
                   6Â° "A"

                PrÃ¡ctica 5

   Profesor: Francisco Javier Luna Rosas
  Alumnos:
      Espinoza SÃ¡nchez Joel Alejandro
      GÃ³mez Garza Dariana
      GonzÃ¡lez Arenas Fernando Francisco

  Fecha de Entrega: 7 de abril del 2021

DescripciÃ³n: SVM aplicado
'''


#%% Importamos las librerÃ-as
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import pickle
import random
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix

#%% Declaramos el directorio donde estÃ¡n las imÃ¡genes de prueba
dir = 'C:\\Users\\alexe\\Desktop\\Proyecto2\\images'

#%% Declaramos las categorÃ-as
categories = ['adulto', 'anciano', 'bebe', 'joven']
```

```
#%% Declaramos la variable que mantendrÃ¡ los datos a lo largo del
procedimiento SVM
data = []

#%% Se revisa por categorÃ-a cada imagen del directorio
for category in categories:
    path = os.path.join(dir, category)
    label = categories.index(category)

    for img in os.listdir(path):
        imgpath = os.path.join(path,img)
        face_img = cv2.imread(imgpath,0)
        try:
            face_img = cv2.resize(face_img, (100,100))
            image = np.array(face_img).flatten()

            data.append([image,label])
        except Exception as e: #ExepciÃ³n de error en resoluciÃ³n de
imagen
            pass

#%% Se crea el documento data1 en el mismo directorio de archivos
con el anÃ¡lisis SVM
pick_in = open('data1.pickle', 'wb')
pickle.dump(data, pick_in)
pick_in.close()

#%% Se carga la informaciÃ³n del archivo data1 en la variable data1
pick_in = open('data1.pickle', 'rb')
data1 = pickle.load(pick_in)
pick_in.close()

#%% Se elige un individuo al azar
random.shuffle(data1)
features1 = []
label1 = []

#%% Se llena el arreglo de caracterÃ-sticas de cada individuo
for feature, labelint in data1:
    features1.append(feature)
    label1.append(labelint)

#%% Se realiza la clasificaciÃ³n a partir del entrenamiento usando
75% de la población
xtrain, xtest, ytrain, ytest = train_test_split(features1, label1,
test_size = 0.20, random_state = 0)
```

```
sc = StandardScaler()

xtrain = sc.fit_transform(xtrain)
xtest = sc.transform(xtest)

classifier = GaussianNB()
classifier.fit(xtrain, ytrain)

ypred = classifier.predict(xtest)
cm = confusion_matrix(ytest, ypred)
print(cm)
```

Anexo 3: Código del clasificador Random Forest orientado a la predicción de edad en Python

```
#%% PresentaciÃ³n
'''
   Universidad AutÃ³noma de Aguascalientes

         Centro de Ciencias BÃ¡sicas
 Departamento de Ciencias de la ComputaciÃ³n
          Aprendizaje Inteligente
                   6Â° "A"

                PrÃ¡ctica 5

   Profesor: Francisco Javier Luna Rosas
  Alumnos:
      Espinoza SÃ¡nchez Joel Alejandro
      GÃ³mez Garza Dariana
      GonzÃ¡lez Arenas Fernando Francisco

  Fecha de Entrega: 7 de abril del 2021

DescripciÃ³n: SVM aplicado
'''

#%% Importamos las librerÃ-as
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import pickle
import random
from sklearn.model_selection import train_test_split
```

```
from sklearn.svm import SVC

from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from      sklearn.metrics      import      classification_report,
confusion_matrix, accuracy_score

#%% Declaramos el directorio donde estÃ¡n las imÃ¡genes de prueba
dir = 'C:\\Users\\alexe\\Desktop\\Proyecto2\\images'

#%% Declaramos las categorÃ-as
categories = ['adulto', 'anciano', 'bebe', 'joven']

#%% Declaramos la variable que mantendrÃ¡ los datos a lo largo del
procedimiento SVM
data = []

#%% Se revisa por categorÃ-a cada imagen del directorio
for category in categories:
    path = os.path.join(dir, category)
    label = categories.index(category)

    for img in os.listdir(path):
        imgpath = os.path.join(path,img)
        face_img = cv2.imread(imgpath,0)
        try:
            face_img = cv2.resize(face_img, (100,100))
            image = np.array(face_img).flatten()

            data.append([image,label])
        except Exception as e: #ExepciÃ³n de error en resoluciÃ³n de
imagen
            pass

#%% Se crea el documento data1 en el mismo directorio de archivos
con el anÃ¡lisis SVM
pick_in = open('data1.pickle', 'wb')
pickle.dump(data, pick_in)
pick_in.close()

#%% Se carga la informaciÃ³n del archivo data1 en la variable data1
pick_in = open('data1.pickle', 'rb')
data1 = pickle.load(pick_in)
pick_in.close()
```

```
#%% Se elige un individuo al azar
random.shuffle(data1)
features1 = []
label1 = []

#%% Se llena el arreglo de caracterÃ-sticas de cada individuo
for feature, labelint in data1:
    features1.append(feature)
    label1.append(labelint)

#%% Se realiza la clasificaciÃ³n a partir del entrenamiento usando
75% de la población
xtrain, xtest, ytrain, ytest = train_test_split(features1, label1,
test_size = 0.20, random_state = 0)

sc = StandardScaler()

xtrain = sc.fit_transform(xtrain)
xtest = sc.fit_transform(xtest)

classifier = RandomForestClassifier(n_estimators = 12, random_state
= 0)
classifier.fit(xtrain, ytrain)

ypred = classifier.predict(xtest)

#print("Matriz de confusión:")
#print(confusion_matrix(ytest, ypred))

#print("Reporte de clasificación:")
#print(classification_report(ytest, ypred))

print("Precisión:")
print(accuracy_score(ytest, ypred))
```