RECONOCIMIENTO FACIAL DE CAPTURAS A PARTIR DE VIDEO-ENTREVISTAS

COORDINADORES DEL PROYECTO:

Juan David Porras Gómez

Jesús Daniel Lizcano Castro

INTRODUCCION

Hoy día solo hace falta pensar en un problema y recurrir a la ciencias para buscar soluciones que lo satisfagan, tal es así que resuelto podemos ahorrar tiempo e optimizar procesos, la IA es uno de los nuevos pilares que aportan en esta área y en este proyecto se usaran recursos de reconocimiento facial para identificar gestos durante videos entrevistas, este método que tenia ya algo de historia se viralizo durante la presente pandemia que inicio a partir del segundo semestre del año 2019.

“Nuestra inteligencia es lo que nos hace humanos, y la IA es una extensión de esa cualidad”. **Yann LeCun**

PROBLEMATICA

Miles de candidatos se presentan a pruebas de admisión cada día, algunas de estas serán en línea y otras meros cuestionarios, el aspecto que tienen en común es que capturan nuestras reacciones, esta claro que para ellos el papel psicológico y la reacción ante posiciones de estrés son indispensables a la hora de elegir un candidato nuevo en su compañía, para eso nuestro proyecto planea catalogar

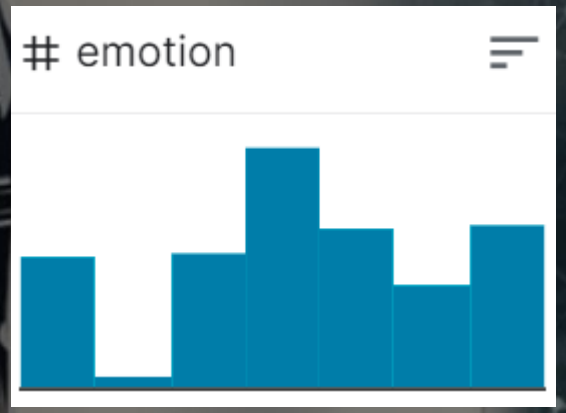
las reacciones que el entrevistado haya tenido durante la prueba y de este modo ayudar en el área de recursos humanos a decidir y calificar de una manera mas ágil a los candidatos sabiendo que serán cientos de persona que se postularan a cargos disponibles

OBJETIVOS

- Categorizar gestos durante una entrevista laboral facilitando la evaluación y selección de candidatos.

DATASET

Distribución de las clases



Vista general de la representación de los datos dentro del Dataset

Emotion	Pixels
0	70 80 82 34 125 27 240 120 76 …
3	75 1 33 24 56 100 205 24 156 …
1	12 34 64 156 1 180 47 58 29 28 …

Descripción de clases

categorie	emotion
0	Angry
1	Disgust
2	Fear
3	Happy
4	Sad
5	Surprise
6	Neutral

Tamaño del Dataset:

27473
unique values

Fuente de Información del Dataset

https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data?select=train.csv

PROCESAMIENTO DE DATOS

- Conversión de los datos a NumpyArray

```
datanp = data.to_numpy()
print(datanp.dtype)
```

- Separación de los las cadenas de texto y posterior conversión a enteros

```
for i in range(len(datanp)):
  datanp[i][1] = np.array(datanp[i][1].split(),dtype=int).reshape((48,48)).tolist()
```

- Exploración de números de datos por cada clase

```
unique, counts = np.unique(datanp[:,0], return_counts=True,)
for i in range(len(unique)):
  print(unique[i],": ",counts[i])
plt.bar(unique,counts)
```

```
0 :  3995
1 :  436
2 :  4097
3 :  7215
4 :  4830
5 :  3171
6 :  4965
<BarContainer object of 7 artists>
```

- Se crea un generador nuevas imágenes como herramienta para balancear

```
datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rotation_range=40,
    horizontal_flip=True,
    fill_mode='reflect'
)
```

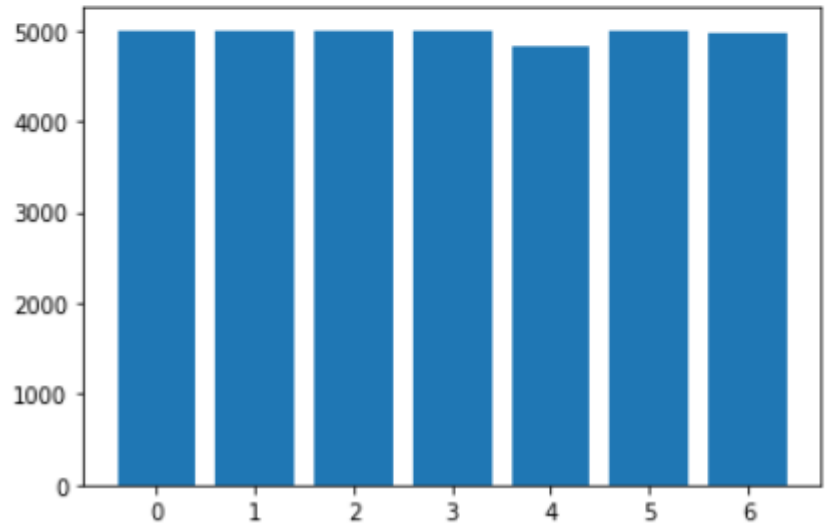
- Balanceo de todas las clases

```
def balancear(n):
  cant = datanp[datanp[:,0]==n].shape[0]
  print(f"Cantidad de datos para el grupo de {n}: ",cant)
  imagenes_generadas = []
  dt = datanp[datanp[:,0]==n]
  for i in range(5000 - cant):
    n = np.random.randint(cant)
    imgGenerada = datagen.flow(np.array(dt[n,1]).reshape(1,48,48,1), batch_size=1,).next().reshape(48,48)
    imagenes_generadas.append([dt[n,0],imgGenerada.astype(int).tolist()])

  imagenes_generadas = np.array(imagenes_generadas)

  class_names = ['Angry','Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']
  plt.figure(figsize=(10,5))
  for i in range(10):
      plt.subplot(2,5,i+1)
      plt.xticks([])
      plt.yticks([])
      plt.grid(False)
      plt.imshow(imagenes_generadas[i,1], cmap=plt.cm.binary)
      plt.xlabel(class_names[imagenes_generadas[i,0]])

  return imagenes_generadas
```

❖ Visualización del Dataset balanceado

```
unique, counts = np.unique(datanp[:,0], return_counts=True,)
for i in range(len(unique)):
  print(unique[i],": ",counts[i])

plt.bar(unique,counts)
```

```
0 :  5000
1 :  5000
2 :  5000
3 :  5000
4 :  4830
5 :  5000
6 :  4965
<BarContainer object of 7 artists>
```



PREDICCIONES E IMPLEMENTACIONES

- CONVOLUTIONAL NEURONAL NETWORK (CNN)

```
model = tf.keras.Sequential([
  tf.keras.layers.Conv2D(32, (3,3), input_shape=(48,48,1), activation='relu'), #1 - blanco y negro
  tf.keras.layers.MaxPooling2D(2,2),
  tf.keras.layers.Conv2D(64, (3,3), activation='relu'), #1 - blanco y negro
  tf.keras.layers.MaxPooling2D(2,2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(256, activation=tf.nn.relu),
  tf.keras.layers.Dense(100, activation=tf.nn.relu),
  tf.keras.layers.Dropout(.2, input_shape=(2,)),
  tf.keras.layers.Dense(60, activation=tf.nn.relu),
  tf.keras.layers.Dense(7, activation=tf.nn.softmax) #Para redes de clasificacion
])
model.summary()
```

```
Epoch 1/10
2436/2436 [==============================] - 17s 7ms/step - loss: 1.7330 - accuracy: 0.3107
Epoch 2/10
2436/2436 [==============================] - 16s 7ms/step - loss: 1.4124 - accuracy: 0.4633
Epoch 3/10
2436/2436 [==============================] - 16s 6ms/step - loss: 1.2338 - accuracy: 0.5370
Epoch 4/10
2436/2436 [==============================] - 16s 6ms/step - loss: 1.0873 - accuracy: 0.5911
Epoch 5/10
2436/2436 [==============================] - 16s 6ms/step - loss: 0.9417 - accuracy: 0.6462
Epoch 6/10
2436/2436 [==============================] - 16s 6ms/step - loss: 0.7980 - accuracy: 0.7002
Epoch 7/10
2436/2436 [==============================] - 16s 6ms/step - loss: 0.6749 - accuracy: 0.7513
Epoch 8/10
2436/2436 [==============================] - 16s 6ms/step - loss: 0.5608 - accuracy: 0.7973
Epoch 9/10
2436/2436 [==============================] - 16s 6ms/step - loss: 0.4640 - accuracy: 0.8333
Epoch 10/10
2436/2436 [==============================] - 16s 6ms/step - loss: 0.3940 - accuracy: 0.8572
<keras.callbacks.History at 0x7f02fc82d8d0>
```

PREDICCIONES E IMPLEMENTACIONES

- Gaussian Naive Bayes

```
accuracy:  0.229
tpr:  0.16633922724296005
tnr:  0.05968169761273209
```

- Decision Tree Classifier

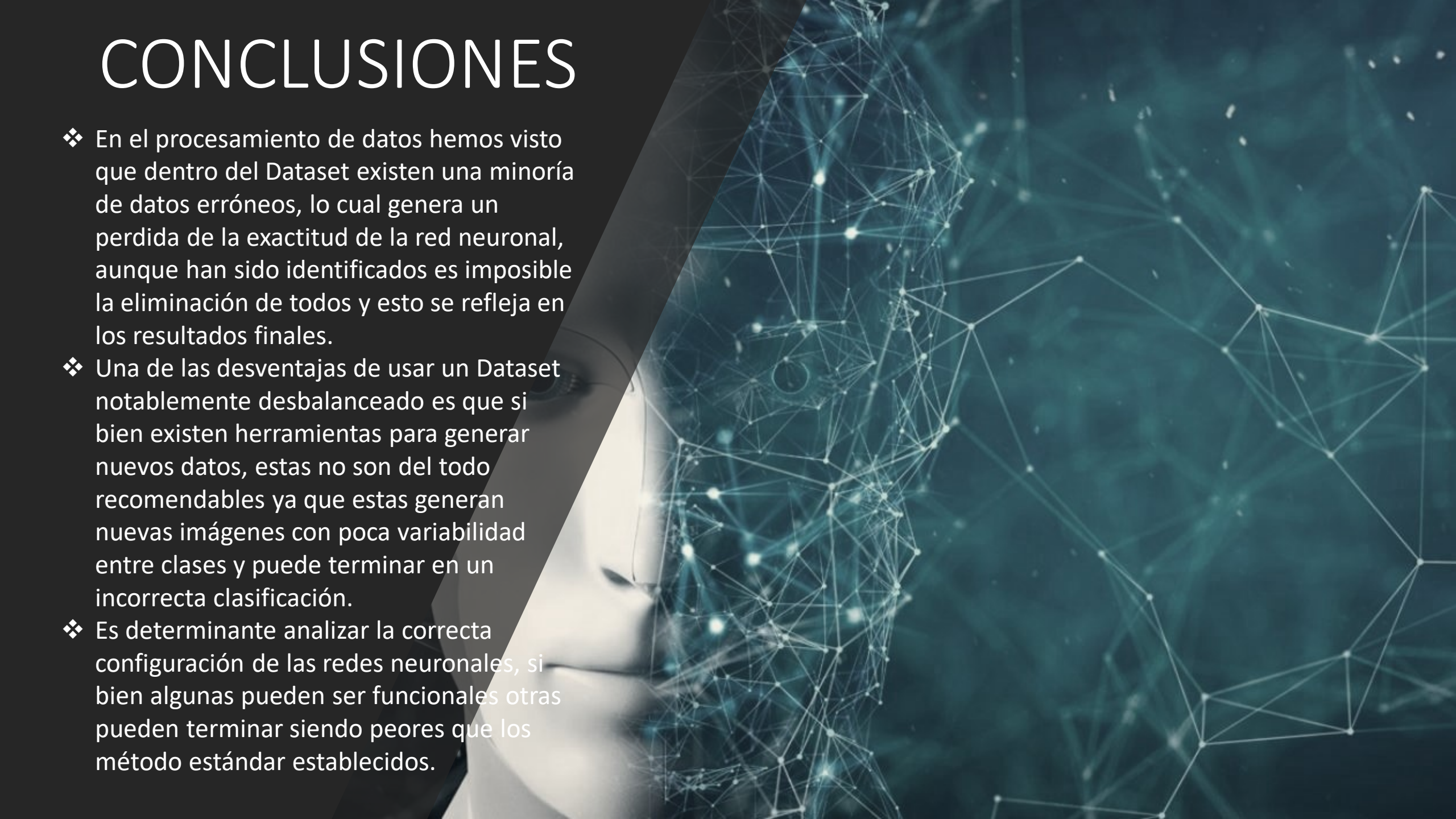
```
accuracy:  0.202
tpr:  0.31761624099541585
tnr:  0.0
```

- Random Forest Classifier (RFC)

```
accuracy:  0.455
tpr:  0.8055009823182712
tnr:  0.34018567639257297
```

CONCLUSIONES

- En el procesamiento de datos hemos visto que dentro del Dataset existen una minoría de datos erróneos, lo cual genera un perdida de la exactitud de la red neuronal, aunque han sido identificados es imposible la eliminación de todos y esto se refleja en los resultados finales.
- Una de las desventajas de usar un Dataset notablemente desbalanceado es que si bien existen herramientas para generar nuevos datos, estas no son del todo recomendables ya que estas generan nuevas imágenes con poca variabilidad entre clases y puede terminar en un incorrecta clasificación.
- Es determinante analizar la correcta configuración de las redes neuronales, si bien algunas pueden ser funcionales otras pueden terminar siendo peores que los método estándar establecidos.



BIBLIOGRAFIA Y REFERENCIAS

- Copyright 2008-2021, La comunidad NumPy. Inicio rápido de NumPy — Manual de NumPy v1.21 https://numpy.org/doc/stable/user/quickstart.html/
- Copyright 2002 - 2012 John Hunter, Darren Dale, Eric Firing, Michael Droettboom y el equipo de desarrollo de Matplotlib; 2012 – 2021, https://matplotlib.org/
- Copyright 2014-2020, imageio contributors Revision 23cdcf5e, https://imageio.readthedocs.io/en/stable/
- Copyright 2008-2021, el equipo de desarrollo de pandas https://pandas.pydata.org
- Tensorflow.org https://www.tensorflow.org/federated
- Redes Neuronales Convolucionales Convolutional Neural Networks: La Teoría explicada en Español | Aprende Machine Learning