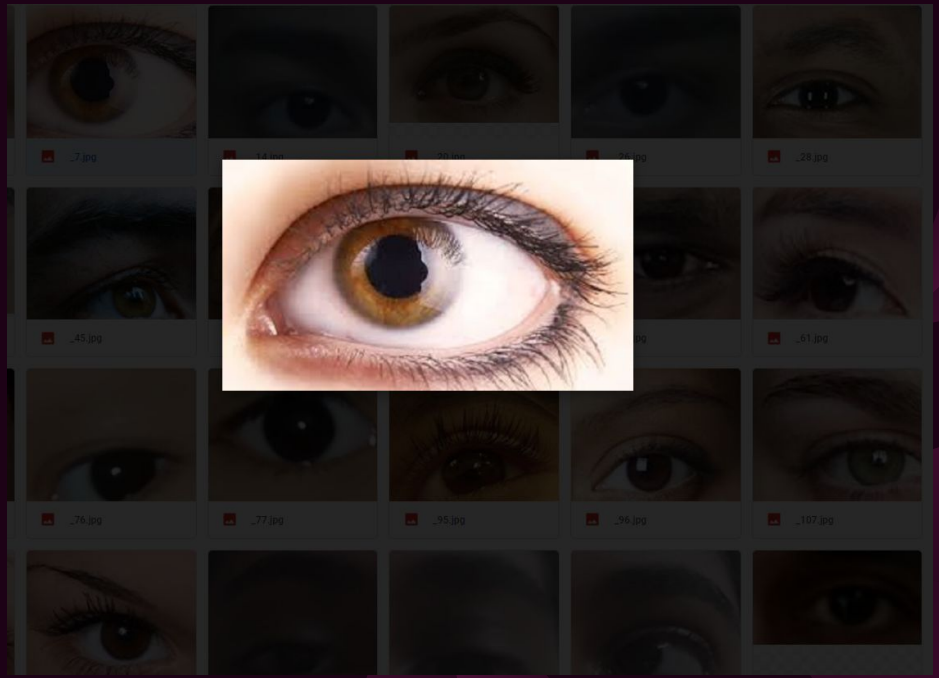
DETECCIÓN DE SOMNOLENCIA EN CONDUCTORES

Iveth Medina-2183199
Najhery Soler-2182696
John Uribe-2152080

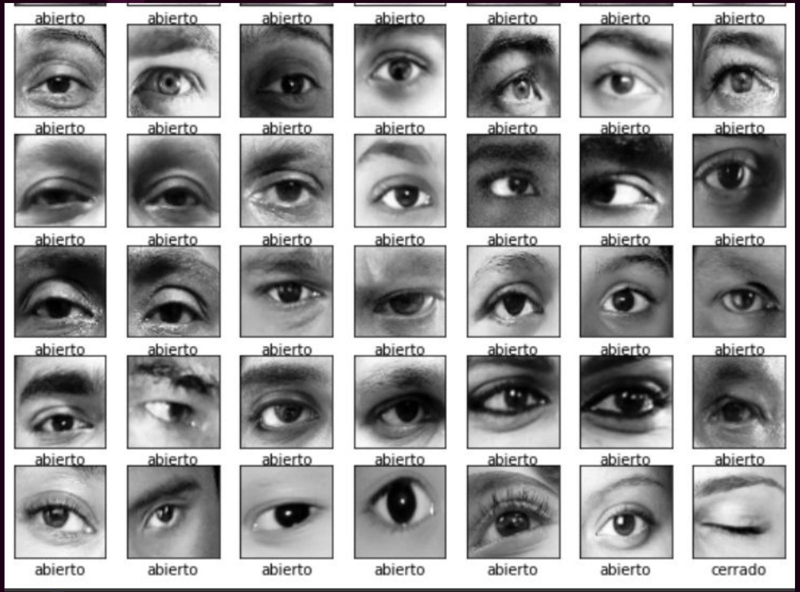
DATASET

```
cantidad de imágenes en train:
ojos abiertos : 617
ojos_cerrados : 617
bostezo : 617
no_bostezo : 616


cantidad de imágenes en test:
ojos abiertos: 109
ojos cerrados: 109
bostezando : 106
no bostezando : 109
```



TRATAMIENTO DE DATOS



pasamos las imágenes a escala de grises

IMPLEMENTACIÓN DE LOS MODELOS DE CLASIFICACIÓN

- GAUSSIAN NB A=0.751

```
matrix de confusión en test
 [[ 91  15   3   0]
 [ 16  88   4   1]
 [  1   0  43  62]
 [  0   0   6 103]]
```

- DECISION TREE A=0.866

```
matrix de confusión en test
 [[89 18  1  1]
 [17 92  0  0]
 [ 0  1 90 15]
 [ 2  1  7 99]]
```

IMPLEMENTACIÓN DE LOS MODELOS DE CLASIFICACIÓN

- SVC A=0.882

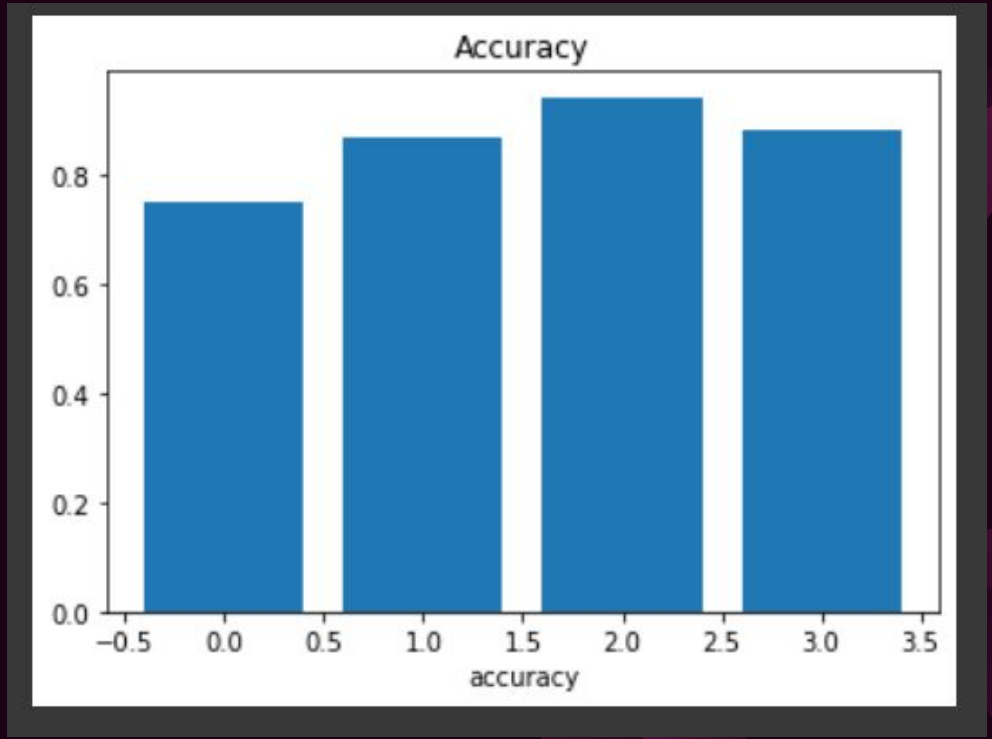
```
matrix de confusión en test
 [[101   8   0   0]
 [  9 100   0   0]
 [  1   0 100   5]
 [  0   0   1 108]]
```

A=0.942

```
matrix de confusión en test
 [[101   8   0   0]
 [  9 100   0   0]
 [  1   0 100   5]
 [  0   0   1 108]]
```

IMPLEMENTACIÓN DE LOS MODELOS DE CLASIFICACIÓN

1. RANDOM FOREST
2. DECISION TREE
3. SVC
4. GAUSSIAN-NB



REDES NEURONALES

```python
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(  input_shape=(64,64,1)),
    tf.keras.layers.Dense(128, activation=tf.nn.relu),
    tf.keras.layers.Dense(64, activation=tf.nn.relu),
    tf.keras.layers.Dense(32, activation=tf.nn.relu),
    tf.keras.layers.Dense(4, activation=tf.nn.softmax)
])
model.summary()
```

```
Epoch 19/24
78/78 [==============================] - 13s 163ms/step - loss: 0.2081 - accuracy: 0.9096
Epoch 20/24
78/78 [==============================] - 13s 162ms/step - loss: 0.2020 - accuracy: 0.9092
Epoch 21/24
78/78 [==============================] - 13s 165ms/step - loss: 0.2013 - accuracy: 0.9116
Epoch 22/24
78/78 [==============================] - 13s 164ms/step - loss: 0.1946 - accuracy: 0.9145
Epoch 23/24
78/78 [==============================] - 13s 165ms/step - loss: 0.1980 - accuracy: 0.9112
Epoch 24/24
78/78 [==============================] - 13s 165ms/step - loss: 0.1935 - accuracy: 0.9141
```

ACCURACY TEST= 0.953

RED CONVOLUCIONAL

```
#@title *code* CNN
model = tf.keras.Sequential([
  tf.keras.layers.Conv2D(32, (3,3), input_shape=(64,64,1), activation='relu'), #1
  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(4, activation=tf.nn.softmax) #Para redes de clasificacion
])
model.summary()
```

LOSS = 0.3143 ACCURACY = 0.9076

CONCLUSIONES

- Los métodos en los que obtuvimos un mayor accuracy fueron de random-forest y la Red-Neuronal.

- Podemos ver que la matriz de confusión del método Gaussian-NB tomo muchas imagenes de ojos cerrados como si fueran abiertos.

- Para aplicar los diferentes métodos de clasificación y que estos funcionen de una mejor manera tenemos que hacer un tratamiento de las imágenes.