



Inteligencia Artificial

VIDA EN VERDE

Modelo de detección de estado de semáforos
para personas Invíidentes.



Integrantes



Cesar Enrique
Rojas Hernández
2191952



Santiago Andres
Delgado Quiceno
2211799



Mariana
Robayo Nieto
2195092

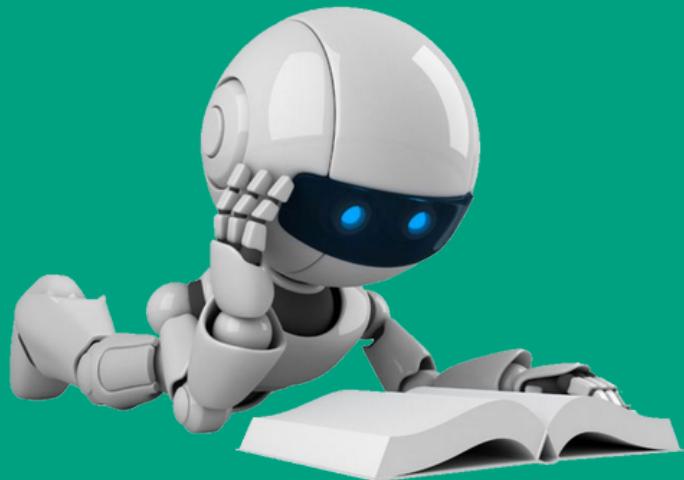


Planteamiento del Problema

El problema de autonomía para las personas invidentes al desplazarse por las calles radica en la falta de conciencia de los conductores y la dificultad para conocer el estado de los semáforos. Esto pone en riesgo su seguridad y limita su movilidad. Para abordar esta situación, se propone desarrollar un modelo de inteligencia artificial que aprenda a detectar el estado de los semáforos y proporcione indicaciones a las personas invidentes, permitiéndoles cruzar de manera más segura y autónoma.



Objetivos



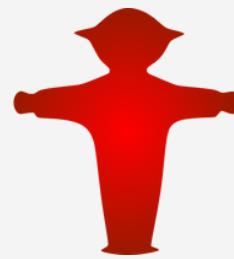
GENERAL

- Desarrollar un sistema de IA basado en modelos de aprendizaje que detecten el estado de los semáforos y proporcione indicaciones precisas.

ESPECIFICOS

- Recopilar y etiquetar un conjunto de datos diverso y representativo de imágenes de semáforos peatonales en diferentes condiciones de iluminación y ángulos de visión.
- Entrenar y comparar varios modelos supervisados y no supervisados, para determinar cual es el más efectivo en el reconocimiento del estado de los semáforos.
- Mejorar la autonomía y seguridad de las personas invidentes al cruzar las calles.

DataSet Utilizado



Semaforos Rojos =1040



Semaforos Verdes=1027

https://drive.google.com/drive/folders/1p6NGn4Ivwlsmy9xdo0nHv1ppGrep1cc?usp=drive_link



Tratamiento de imágenes

- Para el tratamiento de imágenes, lo que inicialmente se hizo fue buscar datasets de semáforos las cuales en el proceso se redimensionaban, se aplanaban y se normalizaban, posteriormente cuando se encontró un nuevo dataset con imágenes de resolución muy grande lo que se hizo fue recortar manualmente las imágenes en la parte que nos interesaba para luego hacer lo que se nombró anteriormente y así entrenar el modelo



Estimadores Utilizados



- GaussianNB
- DecisionTreeClassifier
- RandomForestClassifier
- Support Vector Classifier
- Redes Neuronales

GaussianNB



```
Precisión del clasificador 'GaussianNB':  
precision    recall   f1-score   support  
  
      0.0       0.48      0.57      0.52      126  
      1.0       0.67      0.58      0.62      187  
  
accuracy          0.58      313  
macro avg       0.57      0.58      0.57      313  
weighted avg     0.59      0.58      0.58      313
```

Tiempo de ejecución: 1 segundo.

Predicción: Rojo



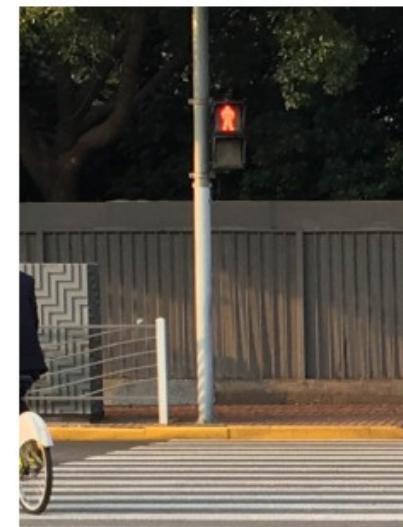
Predicción: Rojo



Predicción: Rojo



Predicción: Verde



Predicción: Rojo



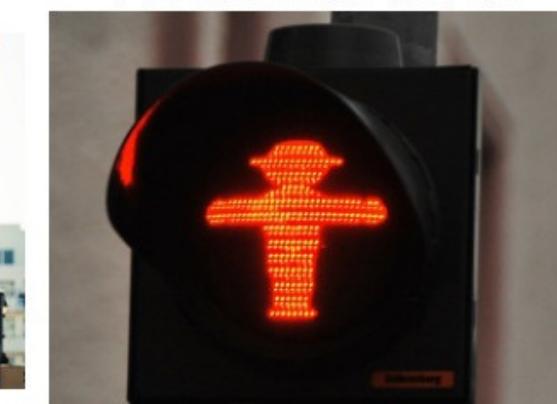
Predicción: Verde



Predicción: Rojo



Predicción: Verde



Predicción: Rojo



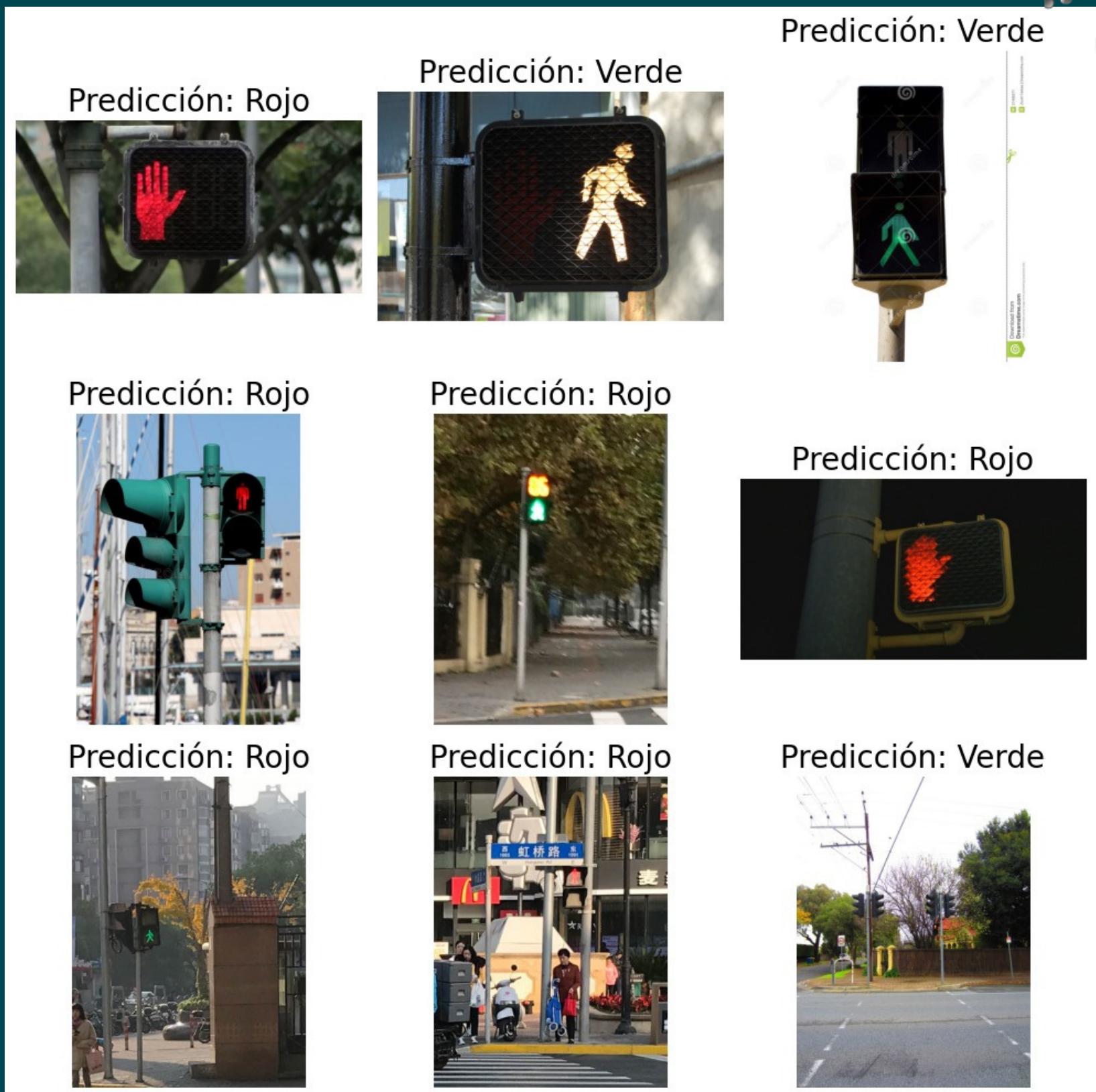
```
1 s = cross_val_score(model, X, y, cv=KFold(20, shuffle=True), scoring="accuracy")  
2 print ("accuracy %.3f (+/- %.5f)"%(np.mean(s), np.std(s)))  
  
accuracy 0.588 (+/- 0.13398)
```

DecisionTreeClassifier

```
Precisión del clasificador 'DecisionTreeClassifier':  
precision    recall   f1-score   support  
  
      0.0       0.63     0.59     0.61      160  
      1.0       0.60     0.64     0.62      153  
  
accuracy          0.62      0.62     0.62      313  
macro avg       0.62     0.62     0.62      313  
weighted avg     0.62     0.62     0.62      313
```

Tiempo de ejecución: 23 segundos

```
1 s = cross_val_score(model, X, y, cv=KFold(20, shuffle=True), scoring="accuracy")  
2 print ("accuracy %.3f (+/- %.5f)"%(np.mean(s), np.std(s)))  
  
accuracy 0.623 (+/- 0.06398)
```



RandomForest Classifier



```
Precisión del clasificador 'RandomForestClassifier':  
precision    recall   f1-score   support  
  
      0.0       0.71      0.63      0.67      170  
      1.0       0.61      0.70      0.65      143  
  
accuracy          0.66      0.66      0.66      313  
macro avg       0.66      0.66      0.66      313  
weighted avg     0.67      0.66      0.66      313
```

Tiempo de ejecución: 21 segundos

```
1 s = cross_val_score(model, X, y, cv=KFold(20, shuffle=True), scoring="accuracy")  
2 print ("accuracy %.3f (+/- %.5f)"%(np.mean(s), np.std(s)))  
  
accuracy 0.658 (+/- 0.05398)
```

Predicción: Verde



Predicción: Rojo



Predicción: Rojo



Predicción: Verde



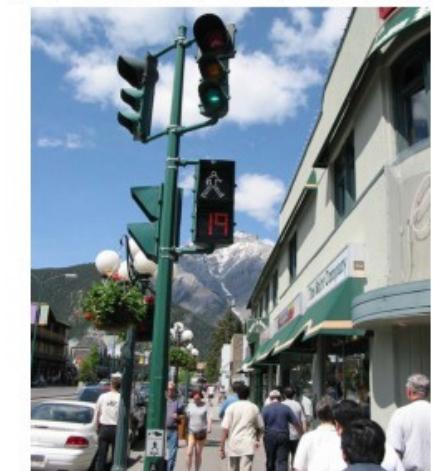
Predicción: Rojo



Predicción: Rojo



Predicción: Verde



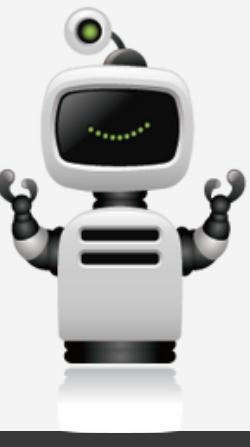
Predicción: Rojo



Predicción: Verde



Support Vector Classifier



| Precisión del clasificador 'SVC': | | | | |
|-----------------------------------|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0.0 | 0.71 | 0.66 | 0.68 | 161 |
| 1.0 | 0.66 | 0.71 | 0.69 | 152 |
| accuracy | | | 0.68 | 313 |
| macro avg | 0.68 | 0.68 | 0.68 | 313 |
| weighted avg | 0.69 | 0.68 | 0.68 | 313 |

Tiempo de ejecución: 1 minuto y 17 segundos

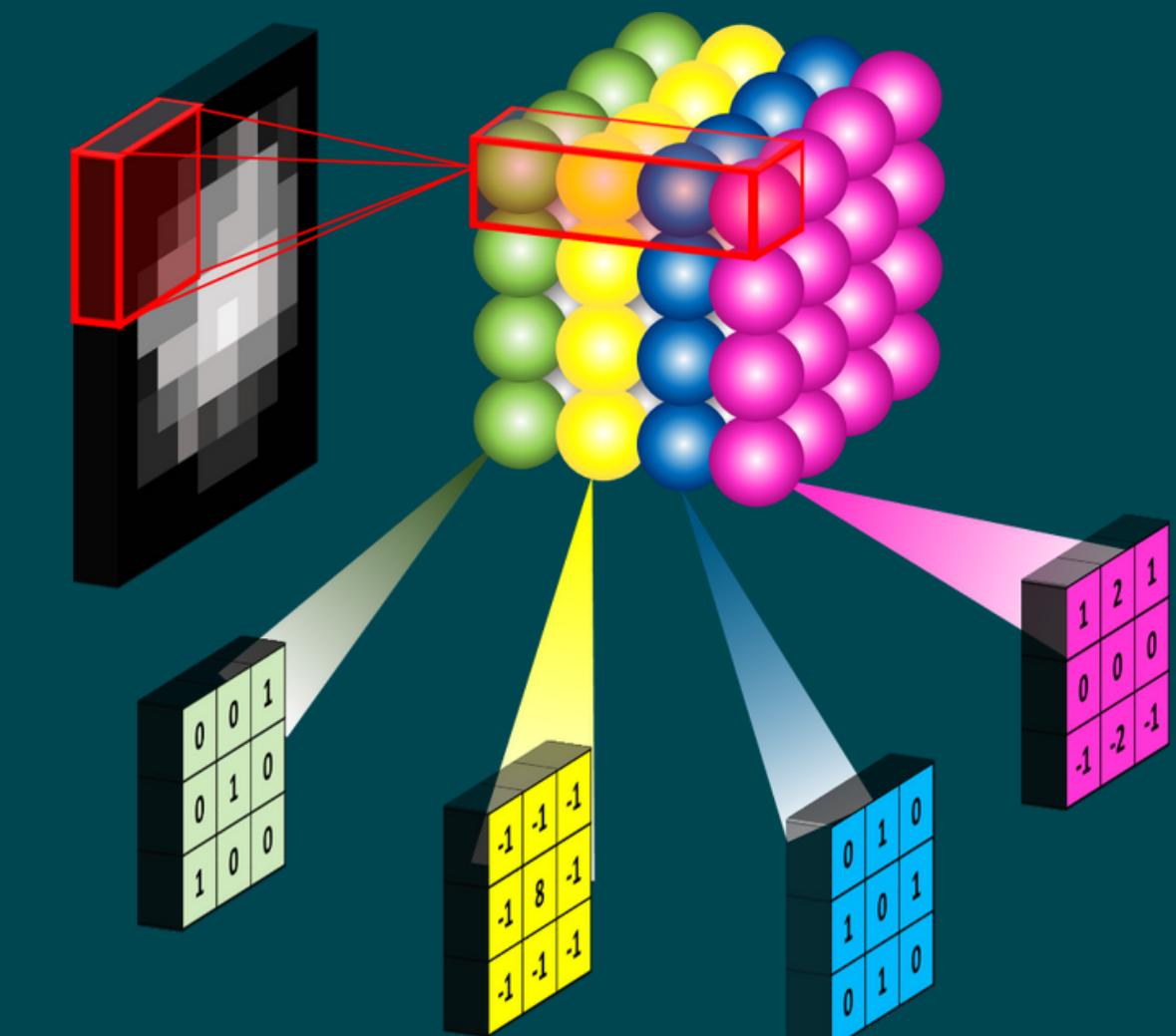


Redes Neuronales

```
model = keras.Sequential([
    keras.layers.Rescaling(1./255, input_shape=X_train[0].shape),
    keras.layers.Conv2D(128, (4, 4), activation='relu'),
    keras.layers.MaxPooling2D((4, 4)),
    keras.layers.Conv2D(64, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D((3, 3)),
    keras.layers.Conv2D(32, (2, 2), activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Flatten(),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy',
              metrics=['accuracy'])

model.fit(X_train, y_train, epochs=15, batch_size=64, validation_data=(X_test, y_test))
```



Resultados

```
1 test_loss, test_acc = model.evaluate(X_test, y_test)  
2 print('Test accuracy:', test_acc)
```

```
7/7 [=====] - 0s 11ms/step - loss: 0.1283 - accuracy: 0.9855  
Test accuracy: 0.9855072498321533
```



Predicción: Verde



Predicción: Rojo



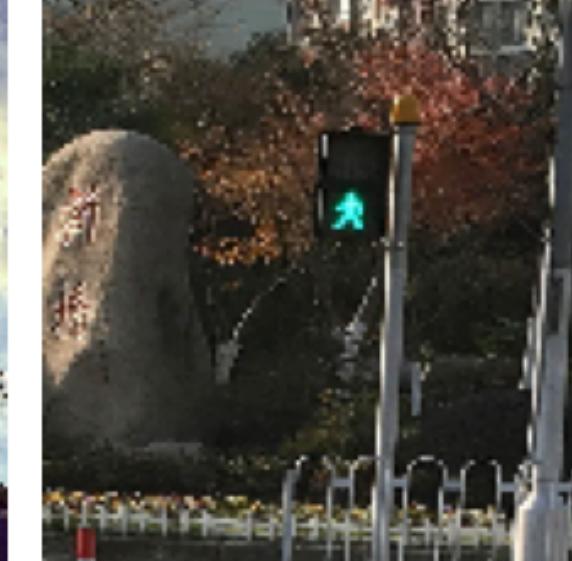
Predicción: Verde



Predicción: Rojo



Predicción: Verde



Predicción: Verde



Conclusiones

- A partir de la construcción de varios modelos con diferentes métodos se logra obtener un modelo funcional que ofrece entre un 95% a un 98% de accuracy el cual se puede considerar una exactitud bastante buena teniendo en cuenta que este es un modelo enfocado en ayudar a personas y el fallar puede implicar jugar con la vida de quienes lo utilicen.
- Durante el desarrollo se evidencio como en un principio con pocos datos los modelos como el GNB,DTC RFC, SVC obtuvieron resultados de mas del 70% de exactitud pero al aumentar los datos estos empezaron a fallar mas y el modelo que mejoró significativamente fueron las redes neuronales, por lo que fue el modelo que se escogió para terminar de mejorar y obtener los resultados de exactitud mas altos.





Gracias

