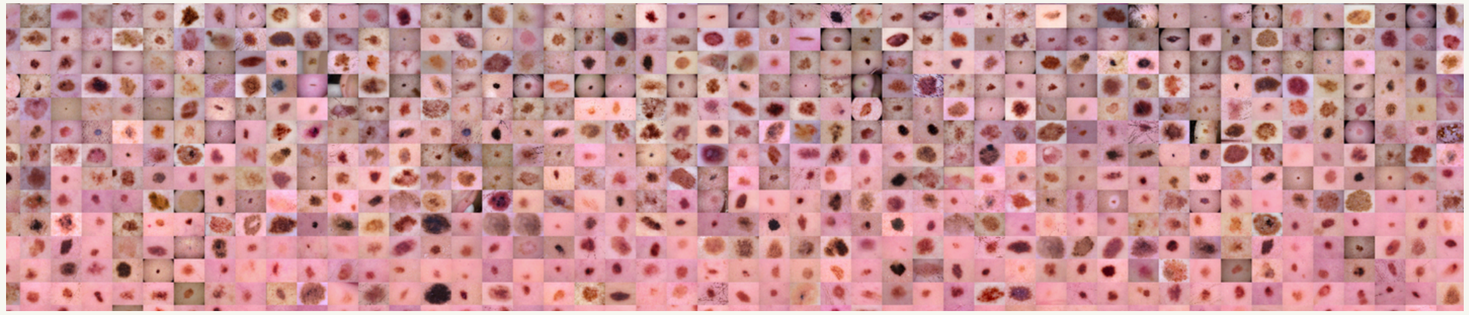
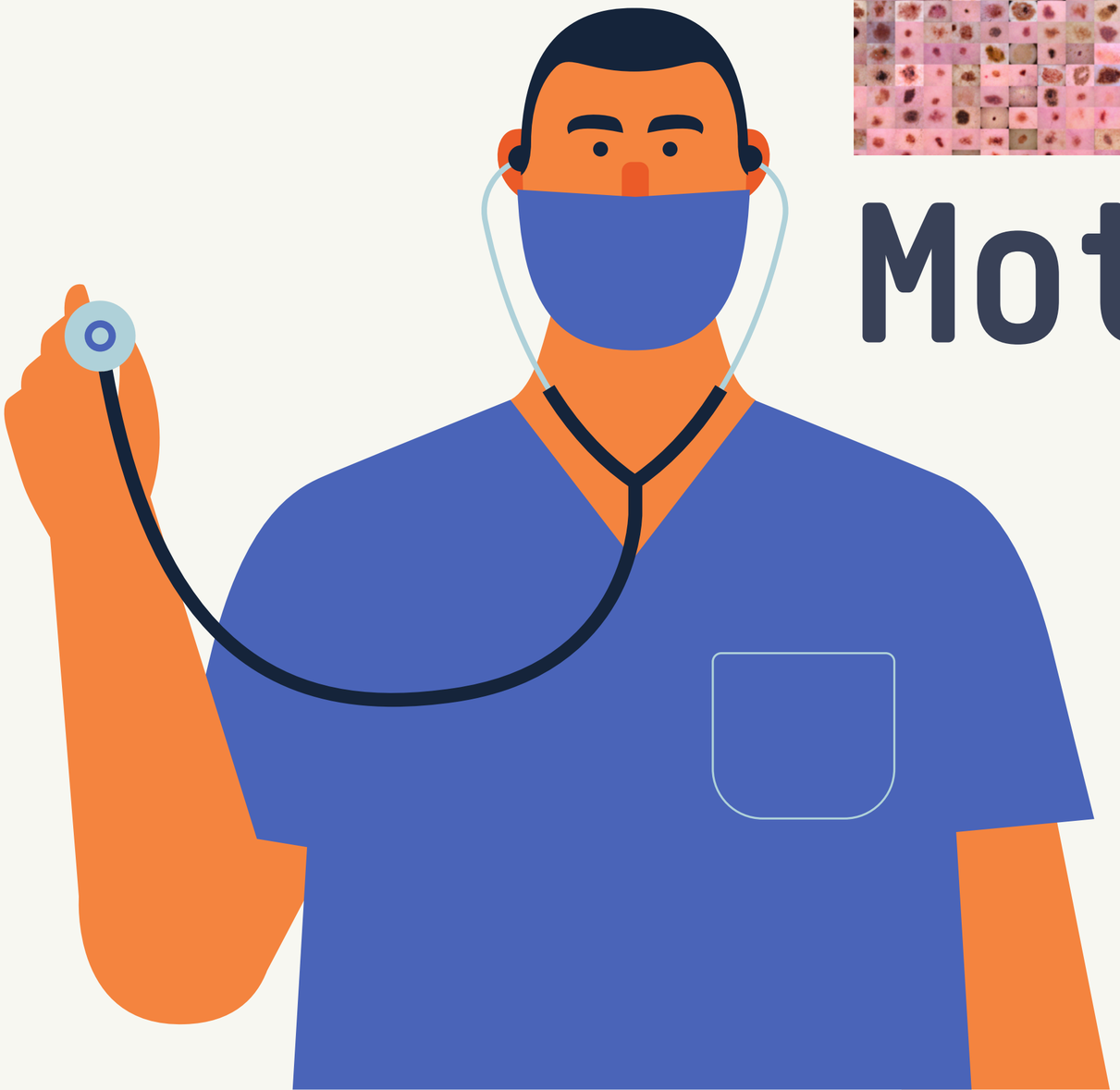
Skin Cancer

GABRIEL ARAUJO
STEVEN CABALLERO



Motivación



- Es uno de los tipos de cáncer más comunes en el mundo.
- Los algoritmos de IA pueden ayudar a los médicos a identificar lesiones malignas con mayor precisión.
- La automatización del diagnóstico permite analizar miles de imágenes en segundos.
- Contiene una gran cantidad de imágenes de lesiones en la piel etiquetadas.

¿Qué queremos lograr con este análisis?

APLICAR Y COMPARAR

Clasificar correctamente los datos, evaluando su rendimiento con métricas como accuracy, precision, recall y F1-score.

EXPLORAR ESTRUCTURAS

Mediante técnicas de clustering como KMeans y PC3 evaluando la calidad de los grupos con métricas internas.

VISUALIZACIÓN DE DATOS

Graficar y analizar visualmente los datos utilizando distintos tipos de representaciones con diagramas para facilitar su interpretación.

CLASES DE LESIONES

Abreviatura	Nombre Completo	Descripción
bkl	Benign Keratosis-like Lesions	Lesiones benignas parecidas a la queratosis, como la queratosis seborreica.
nv	Melanocytic Nevi	Nevos melanocíticos o lunares, generalmente benignos.
df	Dermatofibroma	Tumor benigno de la piel, generalmente inofensivo.
vasc	Vascular Lesions	Lesiones vasculares como hemangiomas o angioqueratomas.
mel	Melanoma	Tipo de cáncer de piel maligno y agresivo.
bcc	Basal Cell Carcinoma	Carcinoma basocelular, un tipo de cáncer de piel común pero de crecimiento lento.
akiec	Actinic keratoses and intraepithelial carcinoma	Una lesión precancerosa de la piel que puede convertirse en carcinoma de células escamosas

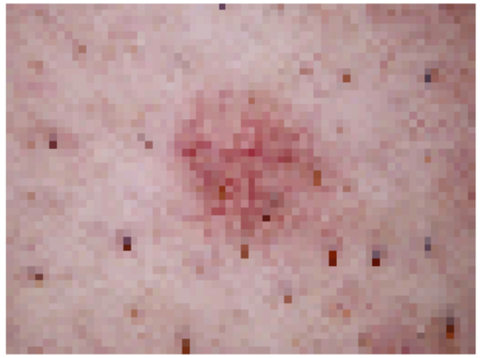
Información del Dataset

COLUMNAS CLAVE

- lesion_id
- image_id
- dx
- dx_type
- path
- **label** “b or m”
- **image** ←

CLASES DE LESIONES

Actinic keratoses (akiec)



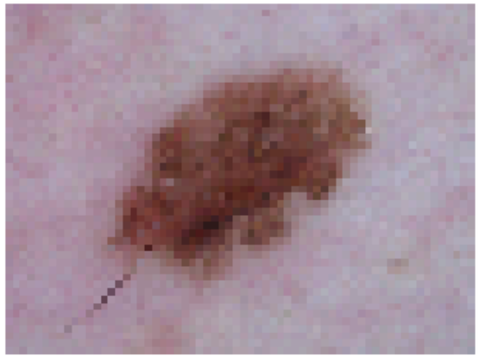
Basal cell carcinoma (bcc)

Benign keratosis-like lesions (bkl)

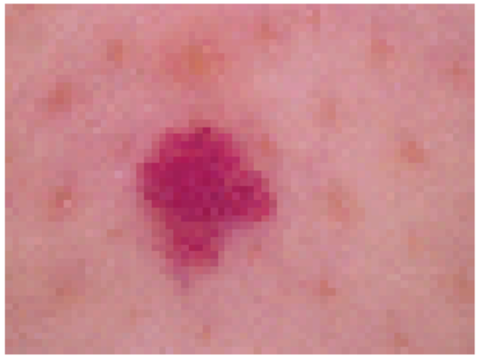
Dermatofibroma (df)

Melanoma (mel)

Melanocytic nevi (nv)



Vascular lesions (vasc)



libreria cv2

```python
def load_image_cv2(path):
    img = cv2.imread(path)
    img = cv2.resize(img, (64, 48))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    return img
```

```python
df_balanced['image'][0]
```

```
ndarray (48, 64, 3) show data
```

Preprocessing

```python
#separacion
malignos = ['mel', 'bcc', 'akiec']
df['label'] = df['dx'].apply(lambda x: 'maligno' if x in malignos else 'benigno')

from sklearn.utils import resample

benigno_df = df[df['label'] == 'benigno']
maligno_df = df[df['label'] == 'maligno']
min_size = min(len(benigno_df), len(maligno_df))

benigno_df = resample(benigno_df, replace=False, n_samples=min_size, random_state=42)
maligno_df = resample(maligno_df, replace=False, n_samples=min_size, random_state=42)
```

RANDOM PERMUTATION

```python
df_balanced = pd.concat([benigno_df, maligno_df]).reset_index(drop=True)

#guardado de imagenes
imageid_path = {os.path.splitext(os.path.basename(x))[0]: x
                    for x in glob(os.path.join(base_dir, '*', '*.jpg'))}

df_balanced['path'] = df_balanced['image_id'].map(imageid_path.get)
df_balanced['image'] = df_balanced['path'].map(load_image_cv2)

#caracteristicas
X = np.stack(df_balanced['image'].values) / 255.0 #normalizacion
y = df_balanced['label'].map({'benigno': 0, 'maligno': 1}).values
```

```python
df_balanced['label'].value_counts()
```

label	count
benigno	1952
maligno	1952

dtype: int64

Modeling

```
x_train, x_test, y_train_o, y_test_o = train_test_split(X, y, test_size=0.20,random_state=999, stratify=y)
#stratify=y ayuda a la partición a separar entre clases de forma controlada
```

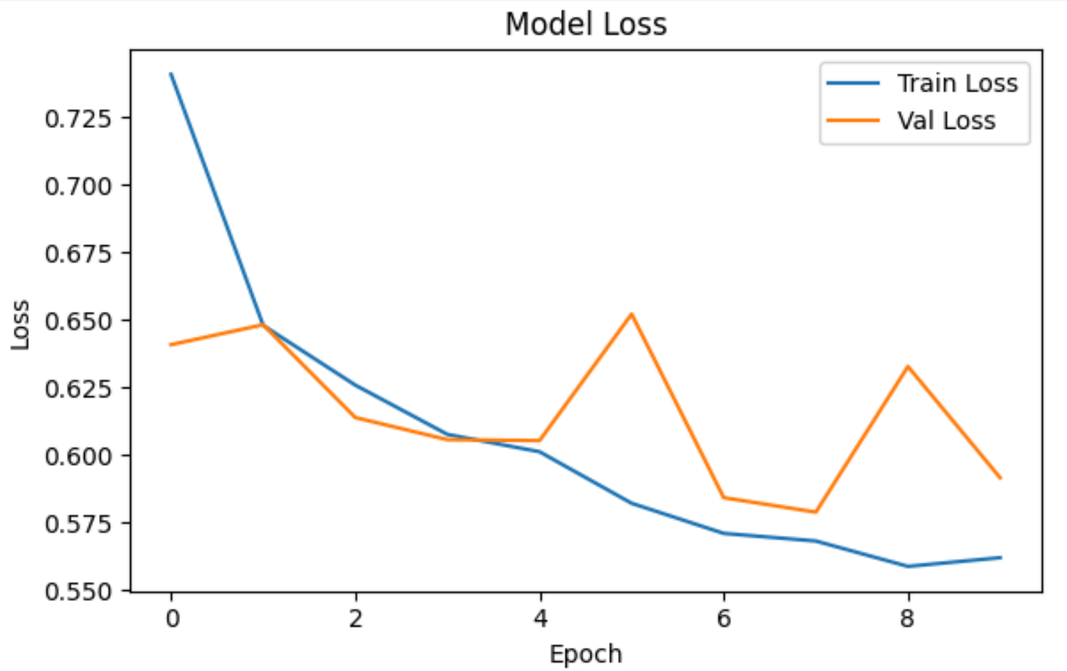
```
#one hot codification
y_train = to_categorical(y_train_o, num_classes = 2)
y_test = to_categorical(y_test_o, num_classes = 2)
```

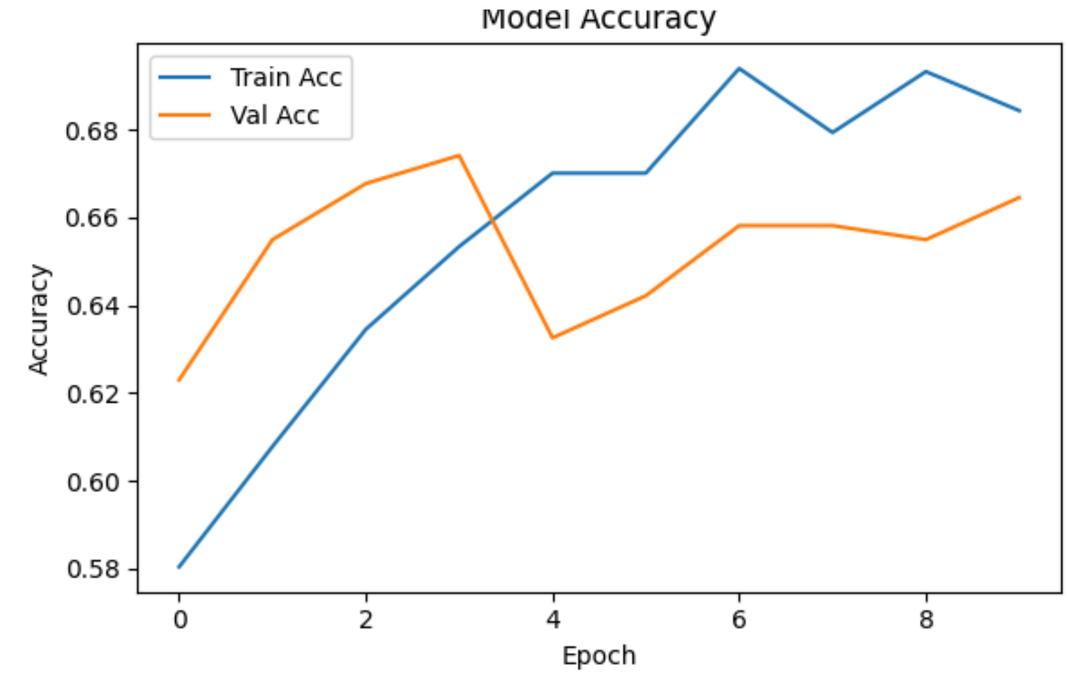
```
x_train, x_validate, y_train, y_validate = train_test_split(x_train, y_train, test_size = 0.1, random_state = 999, stratify=y_train)
x_train = x_train.reshape(x_train.shape[0], *(48, 64, 3))
x_test = x_test.reshape(x_test.shape[0], *(48, 64, 3))
x_validate = x_validate.reshape(x_validate.shape[0], *(48, 64, 3))
```

Modelos S

MODEL RELU

```
#@title model ANN relu
model_ann_relu = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(48, 64, 3)),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')
])

model_ann_relu.summary()

model_ann_relu.compile(
    optimizer=tf.keras.optimizers.SGD(),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

history=model_ann_relu.fit(
    x_train, y_train,
    validation_data=(x_validate, y_validate),
    epochs=10,
    batch_size=32
)

test_loss_ann_relu, test_acc_ann_relu = model_ann_relu.evaluate(x_test, y_test)
print('Test accuracy:', test_acc_ann_relu)

plot_model_history(history)
```

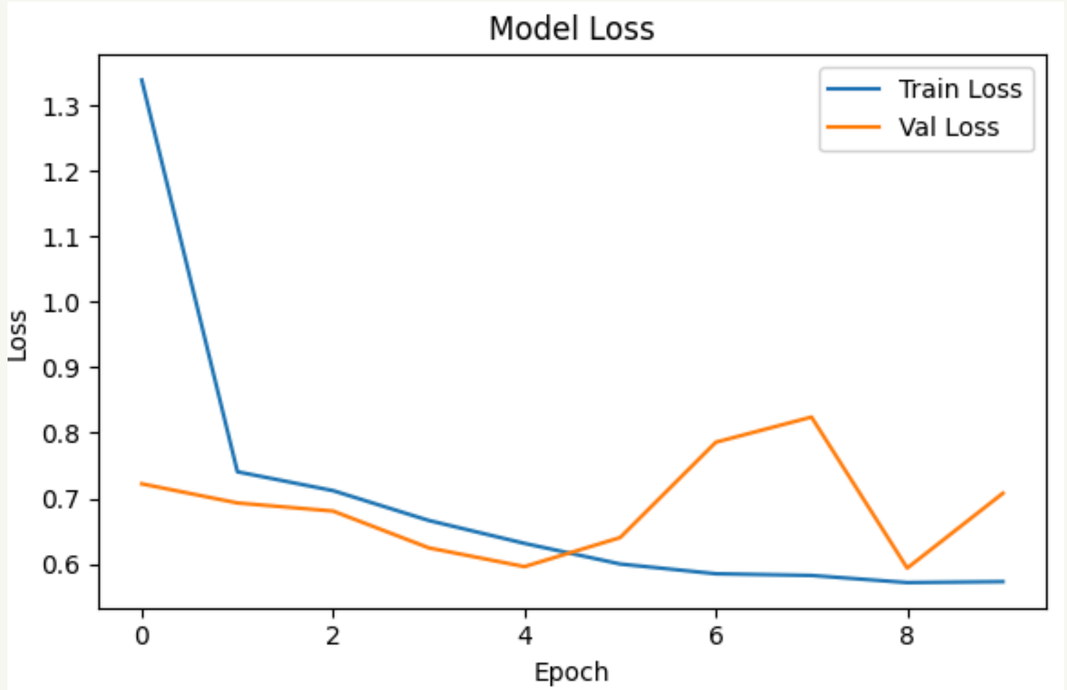


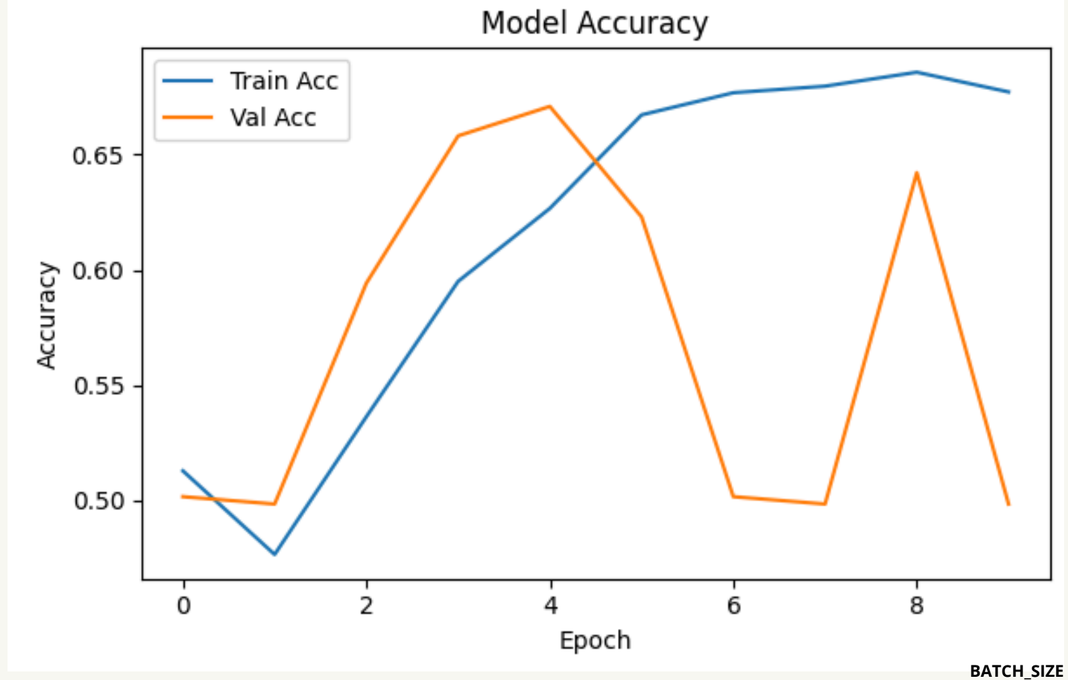


Modelos S

MODEL ANN DIFFERENT

```python
#@title model ANN different
model_ann_different = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(48, 64, 3)),
    tf.keras.layers.Dense(2048, activation='selu'),
    tf.keras.layers.AlphaDropout(0.3),
    tf.keras.layers.Dense(512, activation='selu'),
    tf.keras.layers.AlphaDropout(0.3),
    tf.keras.layers.Dense(128, activation='selu'),
    tf.keras.layers.Dense(64, activation='selu'),
    tf.keras.layers.Dense(2, activation='softmax')
])

model_ann_different.compile(
    optimizer=tf.keras.optimizers.Nadam(),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

history=model_ann_different.fit(
    x_train, y_train,
    validation_data=(x_validate, y_validate),
    epochs=10,
    batch_size=32
)

test_loss_ann_different, test_acc_ann_different = model_ann_different.evaluate(x_test, y_test)
print('Test accuracy (ANN diferente):', test_acc_ann_different)

plot_model_history(history)
```



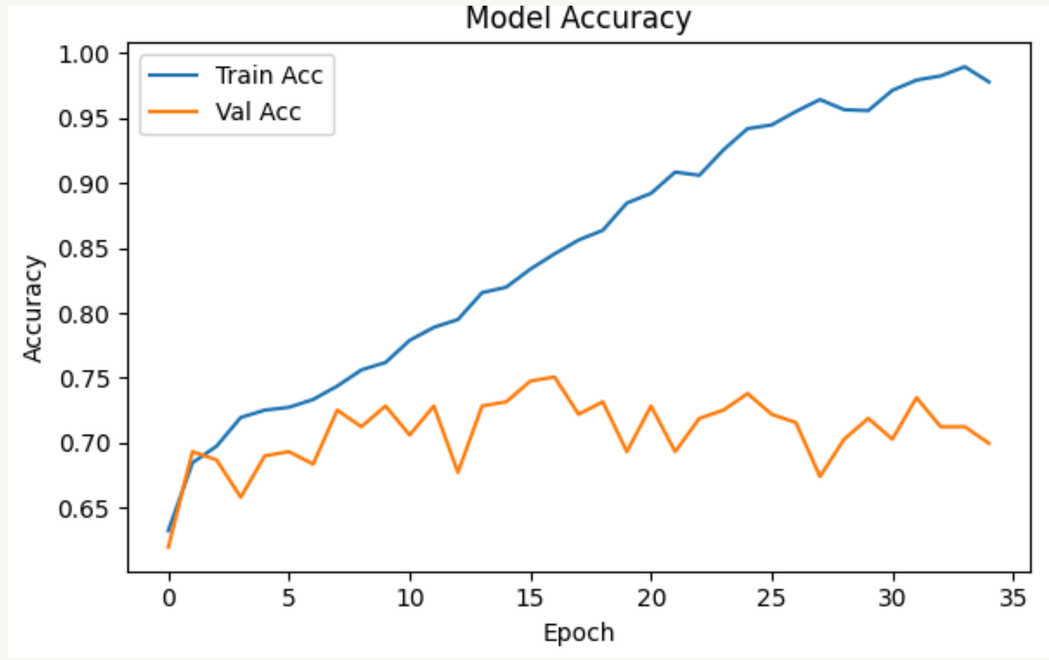


BATCH_SIZE

Modelos S

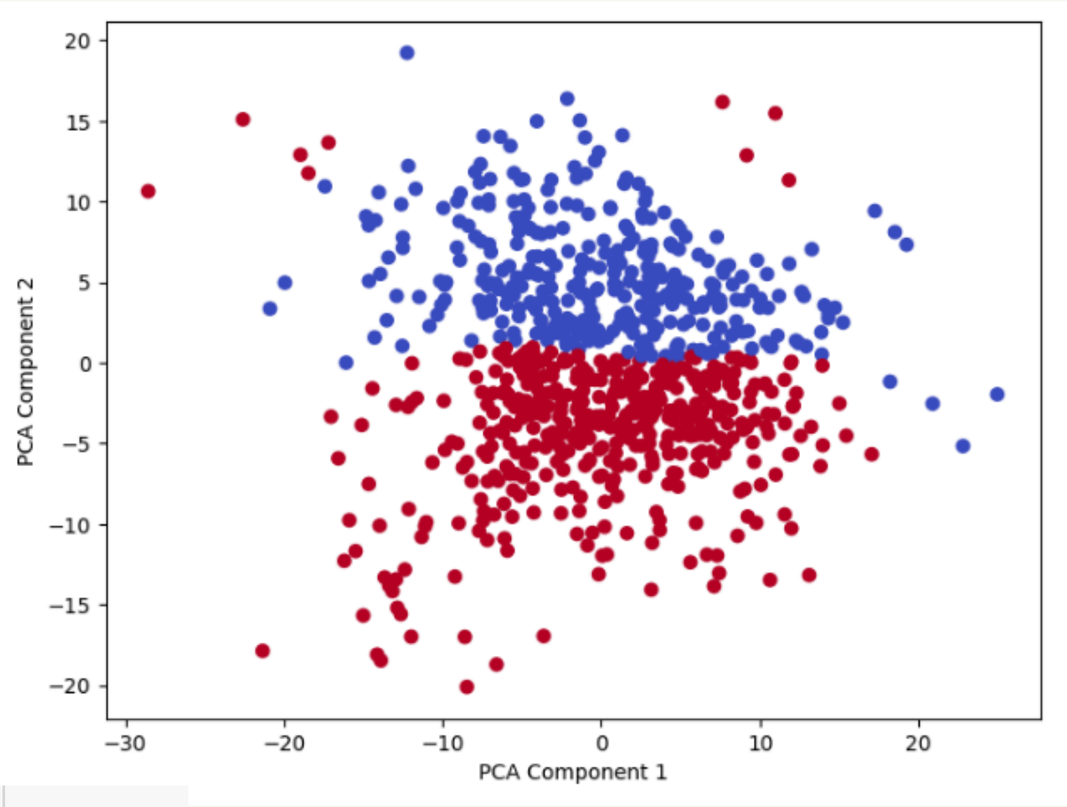
MODEL CNN

```
#@title model CONV2D
model_cnn = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(48, 64, 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),

    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),

    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')
])

model_cnn.compile(
    optimizer=tf.keras.optimizers.Adam(),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

history=model_cnn.fit(
    x_train, y_train,
    validation_data=(x_validate, y_validate),
    epochs=35,
    batch_size=32
)

test_loss_cnn, test_acc_cnn = model_cnn.evaluate(x_test, y_test)
print('Test accuracy (CNN):', test_acc_cnn)

plot_model_history(history)
```



Modelos NS

PCA




```python
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_flat)

x_train, x_test, y_train, y_test = train_test_split(X_pca, y, stratify=y, test_size=0.2)

svm_model = SVC(kernel='rbf', C=1, gamma='scale')
svm_model.fit(x_train, y_train)
y_pred_svm = svm_model.predict(x_test)

print(classification_report(y_test, y_pred_svm))
```

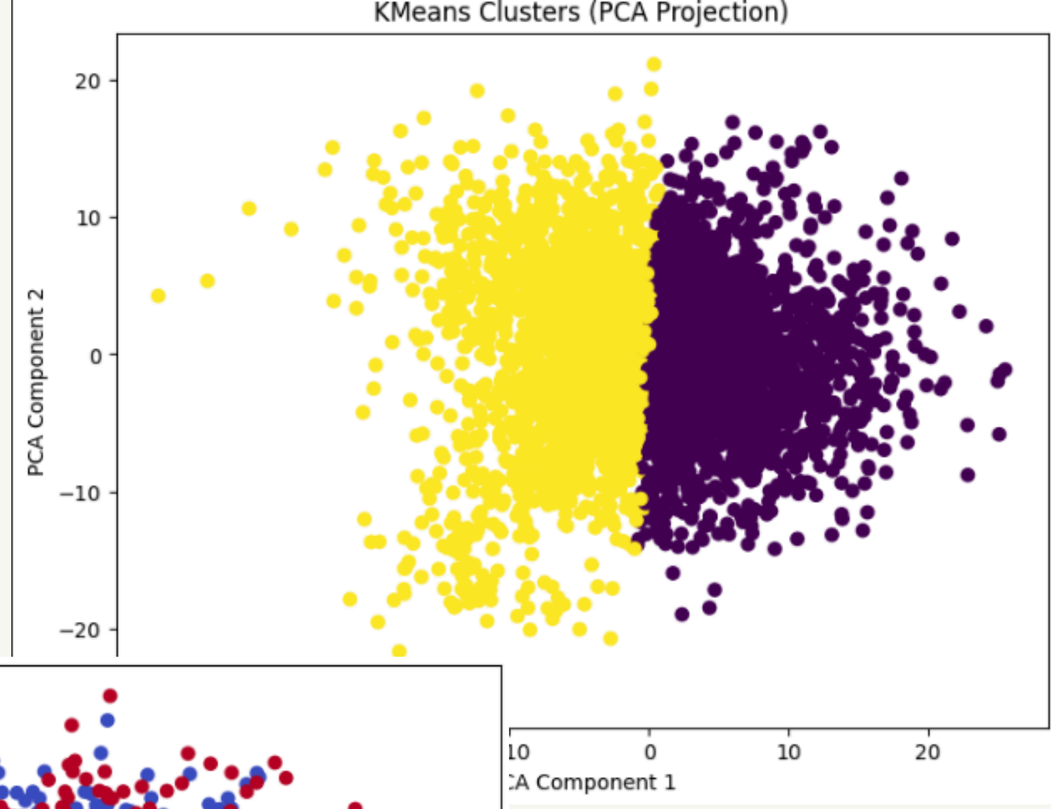
	precision	recall	f1-score	support
0	0.68	0.59	0.63	390
1	0.64	0.72	0.68	391
accuracy			0.66	781
macro avg	0.66	0.66	0.66	781
weighted avg	0.66	0.66	0.66	781

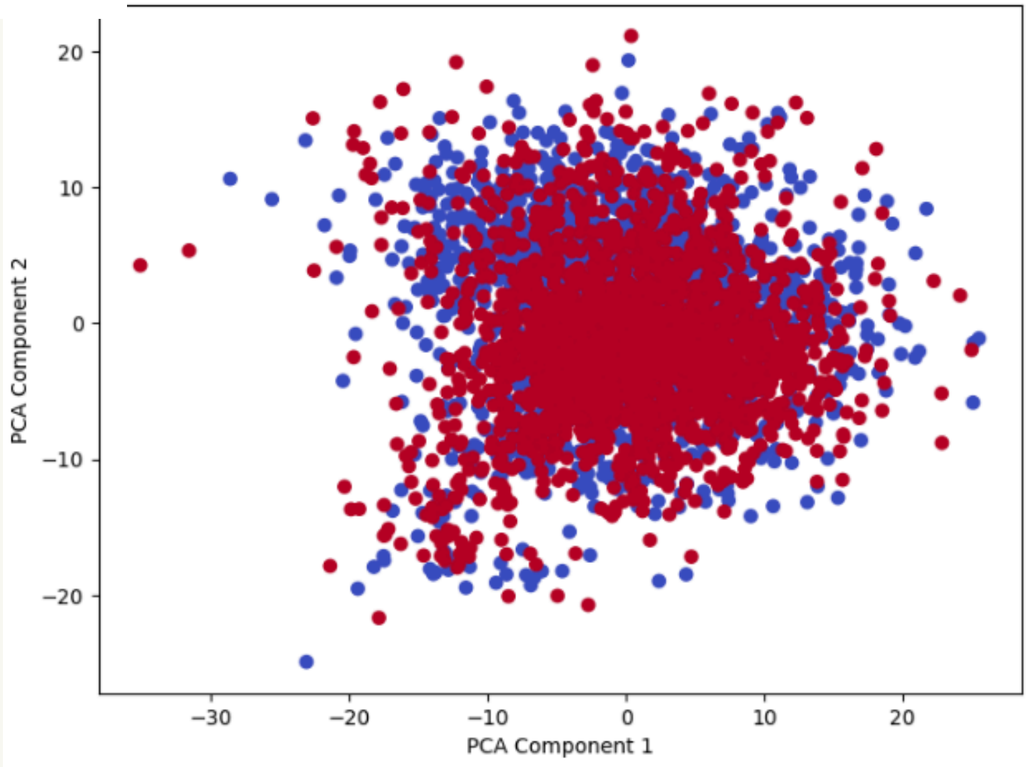
Modelos NS

KMEANS

```
kmeans = KMeans(n_clusters=2, random_state=42)
clusters_kmeans = kmeans.fit_predict(X_flat)

print(classification_report(y, clusters_kmeans))
```

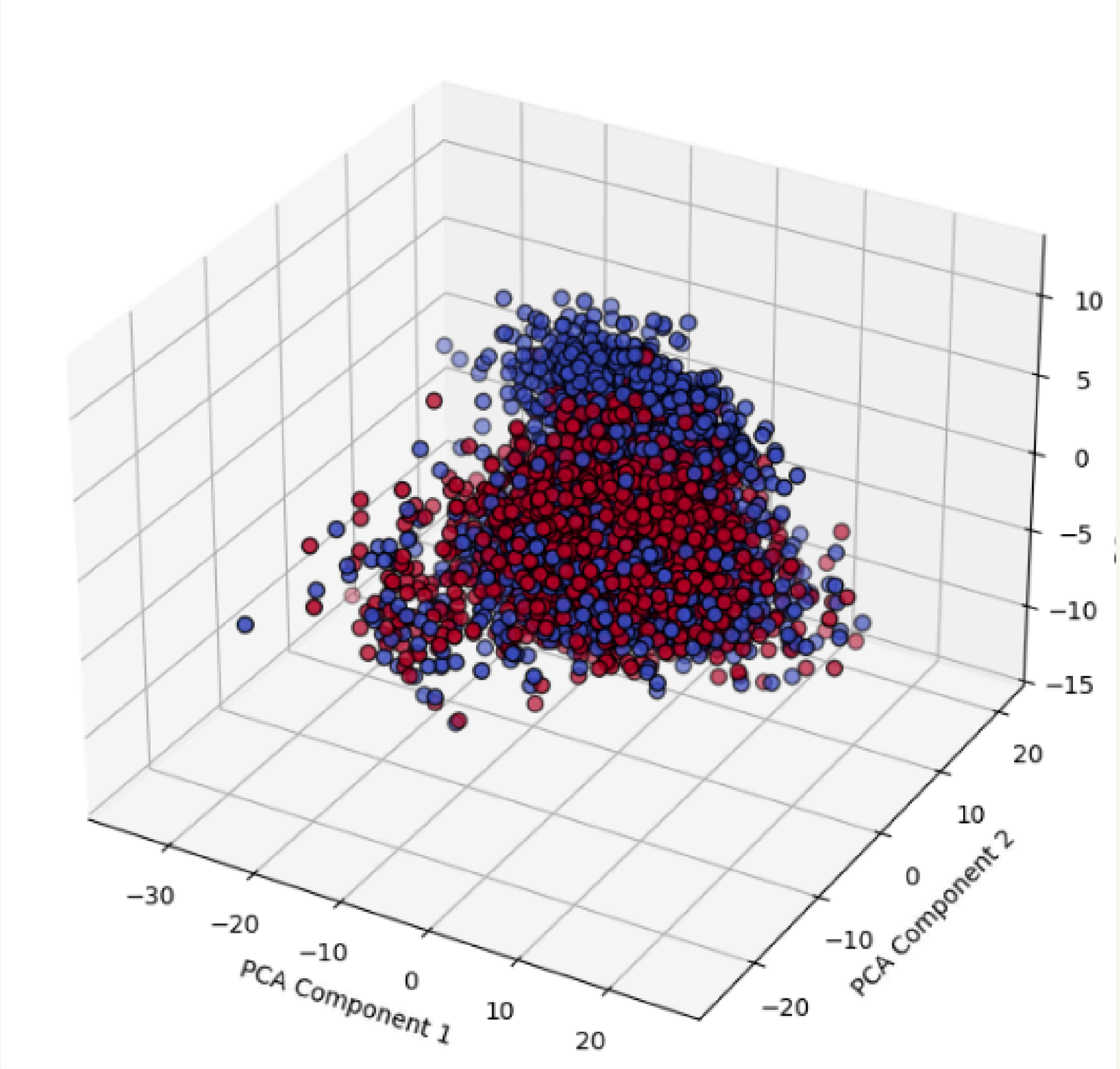
	precision	recall	f1-score	support
0	0.48	0.49	0.48	1952
1	0.48	0.48	0.48	1952
accuracy			0.48	3904
macro avg	0.48	0.48	0.48	3904
weighted avg	0.48	0.48	0.48	3904



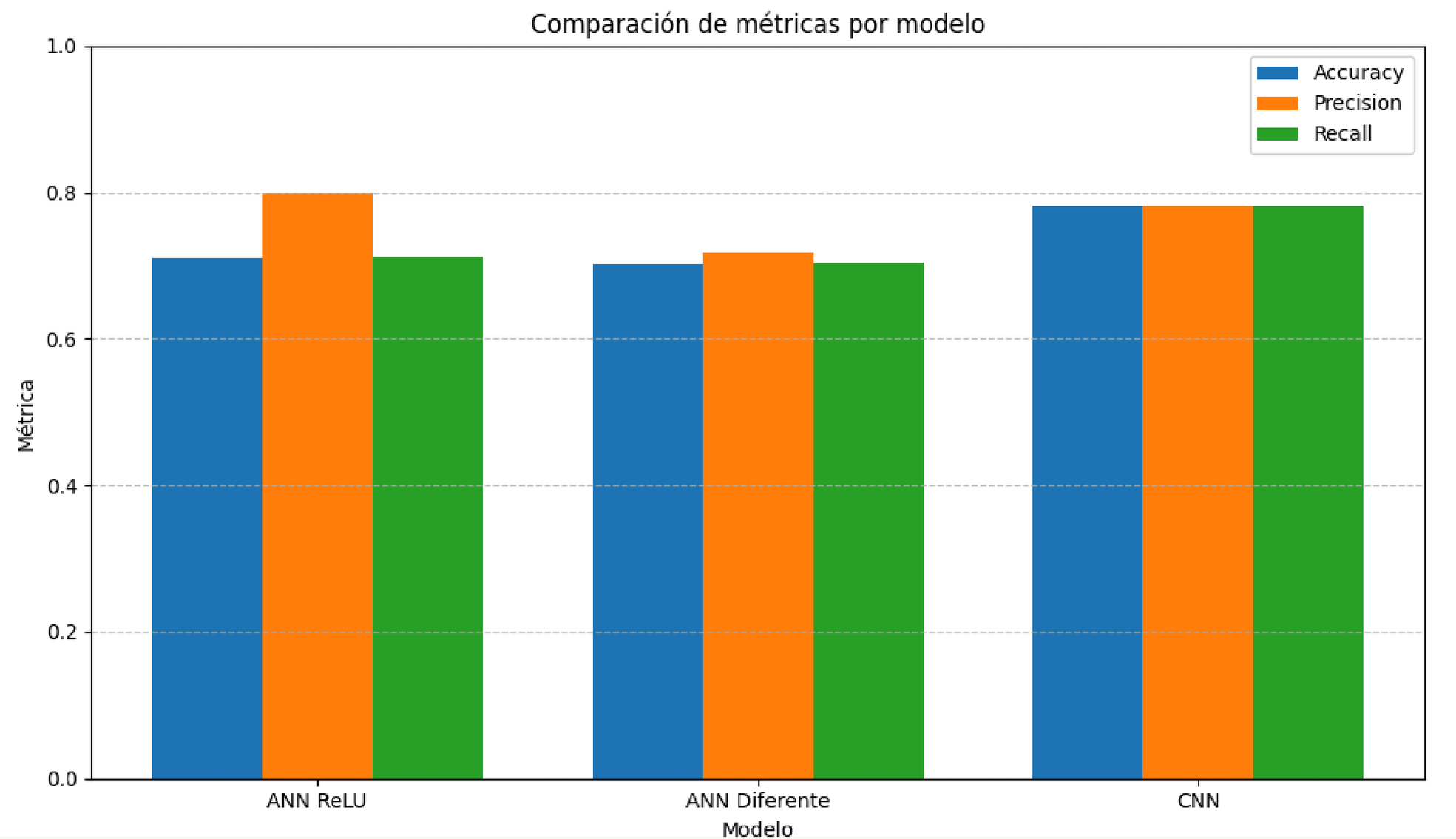


Modelos NS

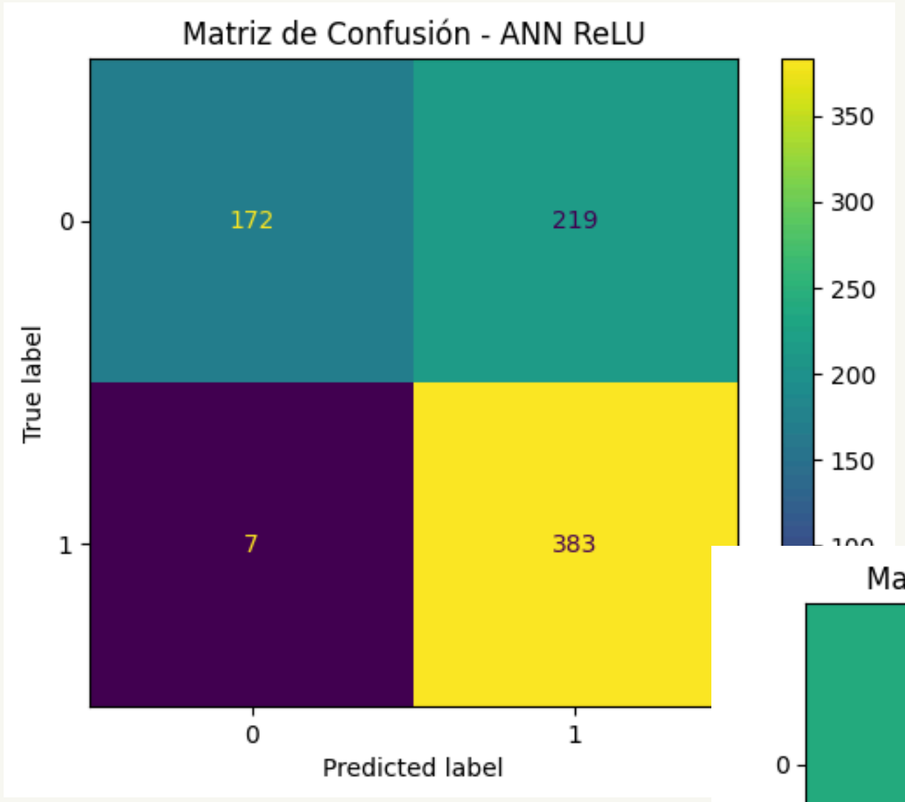
PCA 3D

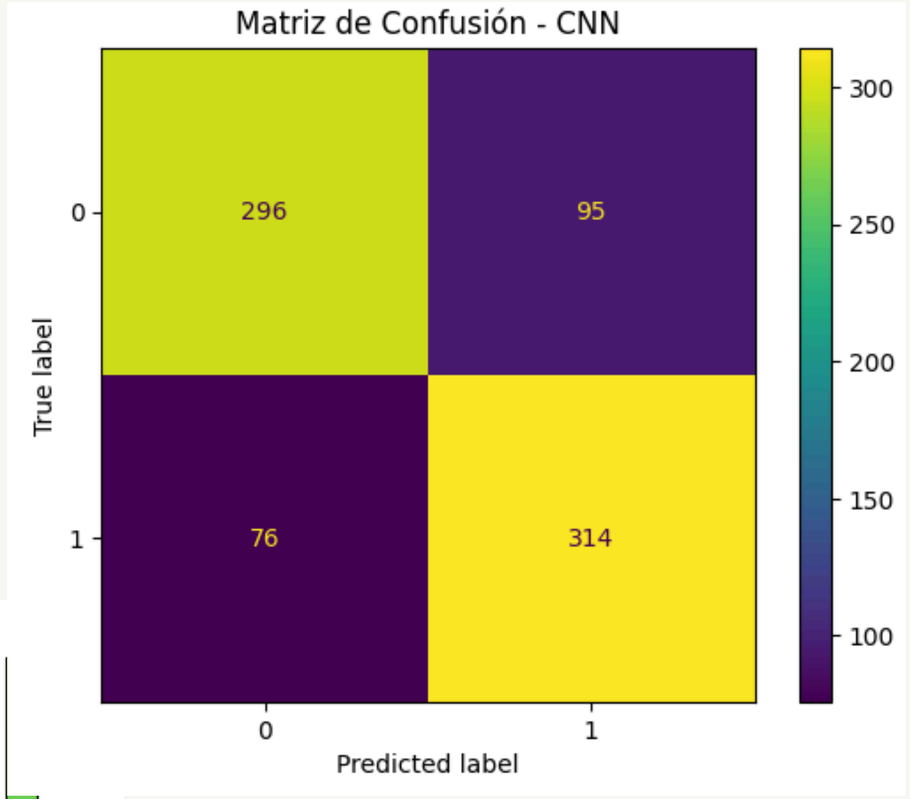


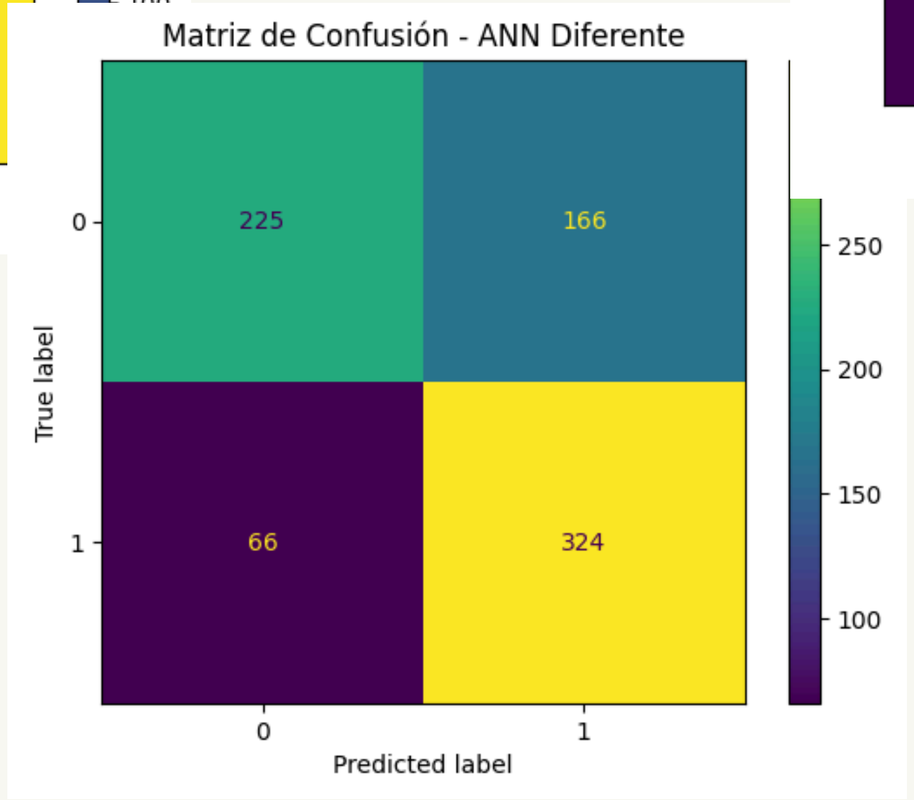
	Modelo	Accuracy	Precision	Recall
0	ANN ReLU	0.710627	0.798553	0.710974
1	ANN Diferente	0.702945	0.717210	0.703108
2	CNN	0.781050	0.781713	0.781081

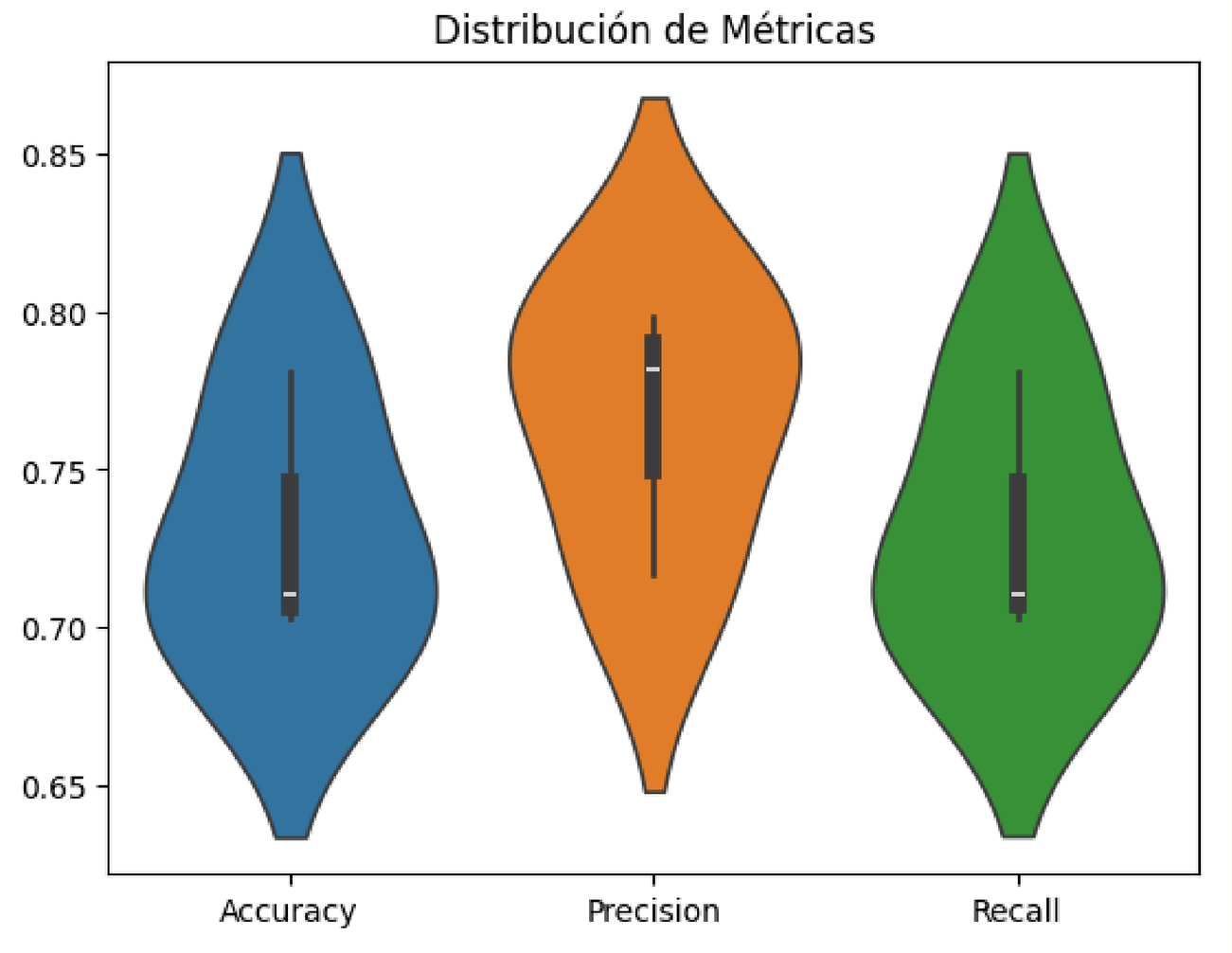


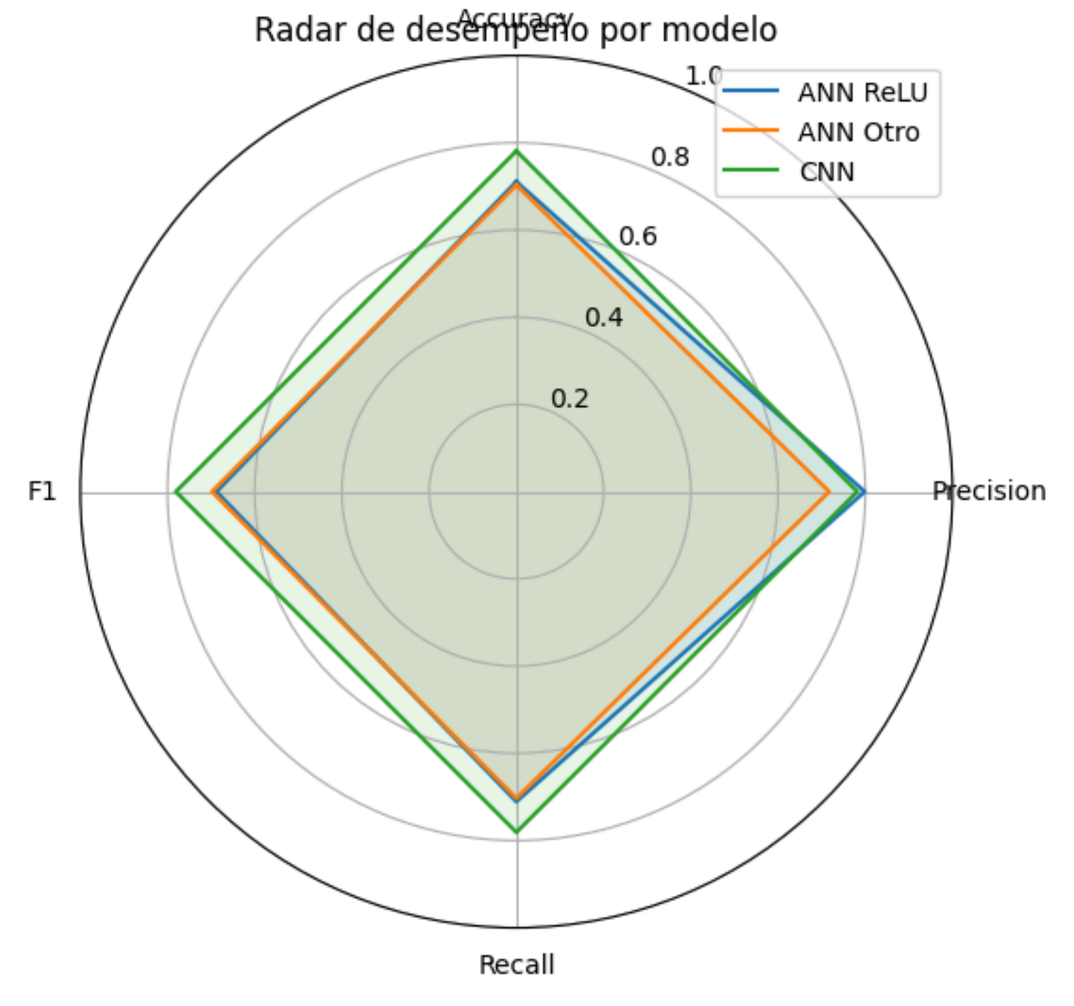
Matriz de Confusión



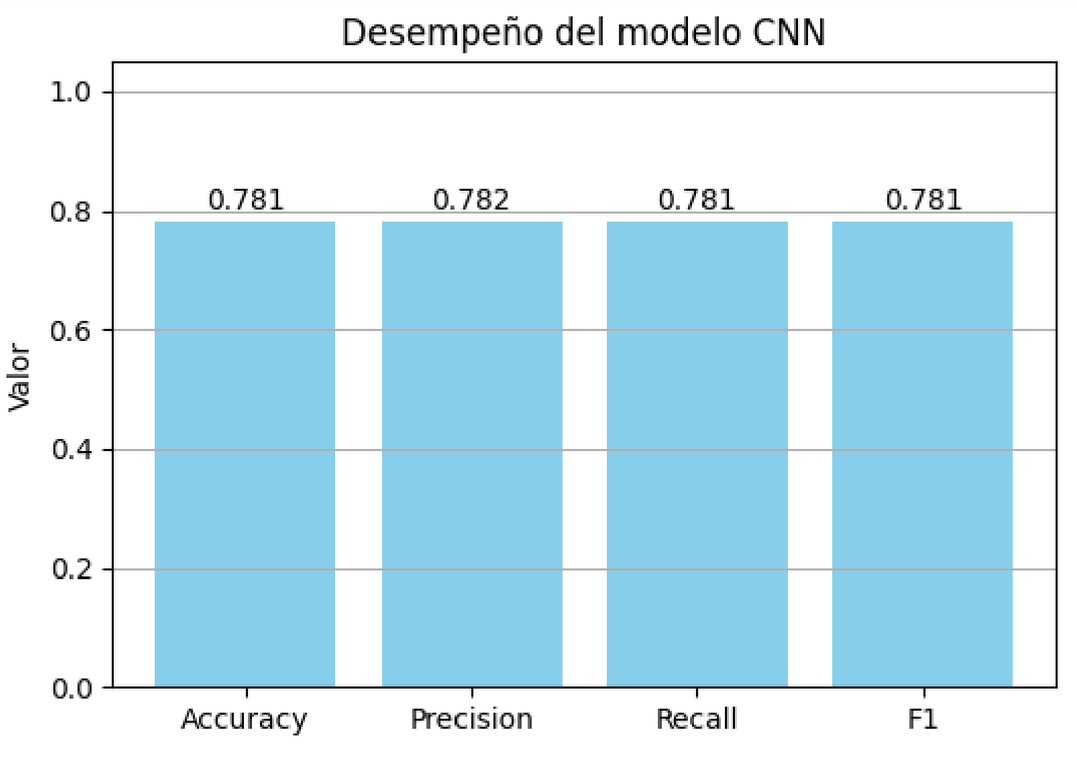





Distribución de Métricas
0.85
0.80
0.75
0.70
0.65
Accuracy
Precision
Recall


Radar de desempeño por modelo
Accuracy
1.0
0.8
0.6
0.4
0.2
ANN ReLU
ANN Otro
CNN
F1
Precision
Recall

REDES NEURONALES CONVOLUCIONALES (CNN)

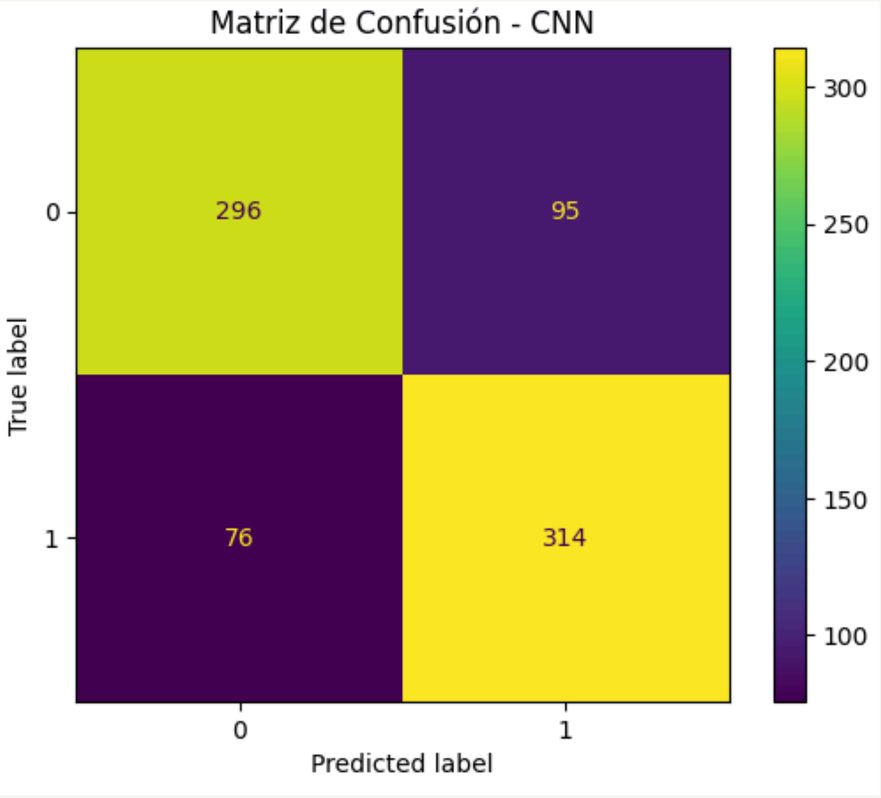


- Especializadas en Imágenes: Detectan automáticamente patrones espaciales (bordes, texturas) usando "filtros".
- Eficientes: Reutilizan filtros en toda la imagen, reduciendo complejidad.
- Robustas: Manejan variaciones en la posición u orientación de los objetos.

El Mejor Modelo es:

Modelo	CNN
Accuracy	0.78105
Precision	0.78173
Recall	0.78105
F1-Score	0.780927
Combined_Score	0.781149

Name: 2, dtype: object



Conclusiones

- La CNN fue el modelo supervisado más eficaz, destacando en la clasificación de imágenes de cáncer de piel gracias a su arquitectura.
- En el análisis no supervisado, PCA con SVC ofreció mejores resultados que KMEANS.
- DBSCAN mostró dificultades para formar clústeres claros, requiriendo un ajuste muy preciso de sus parámetros de densidad.


¿ALGUNA PREGUNTA?