DETECCIÓN DEL GRADO DE ACNÉ CON IMÁGENES

Laura Vanessa Mendez Rivera- 2191951

Alejandro Nunez Herrera-2191947

Juan David Morantes Vergara-2191931

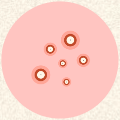


Tabla de Contenidos

01 Planteamiento del problema

Descripcion del proyecto

02 Dataset

Visualizacion del contenido del dataset utilizado

03 Modelos clasificación

04 Deep Learning

05 Aplicativilidad

01

Planteamiento del problema

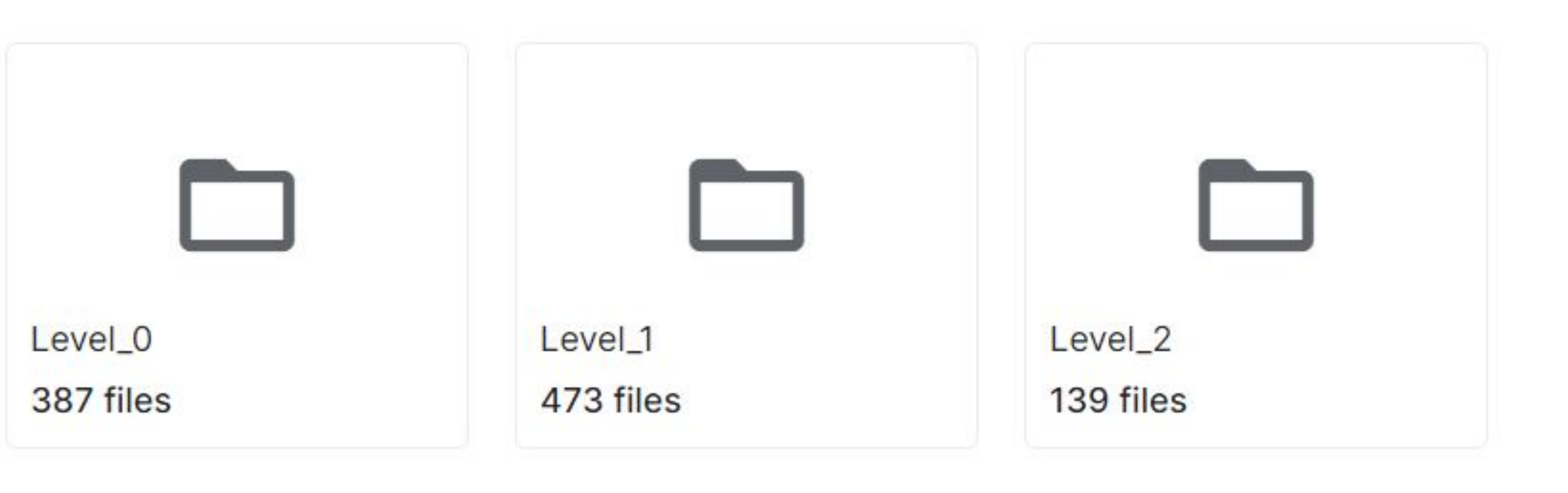
Planteamiento del problema

Detectar el grado de acne que presenta una persona por medio de imagenes

02

DATASET

Dataset



Level_0

387 files

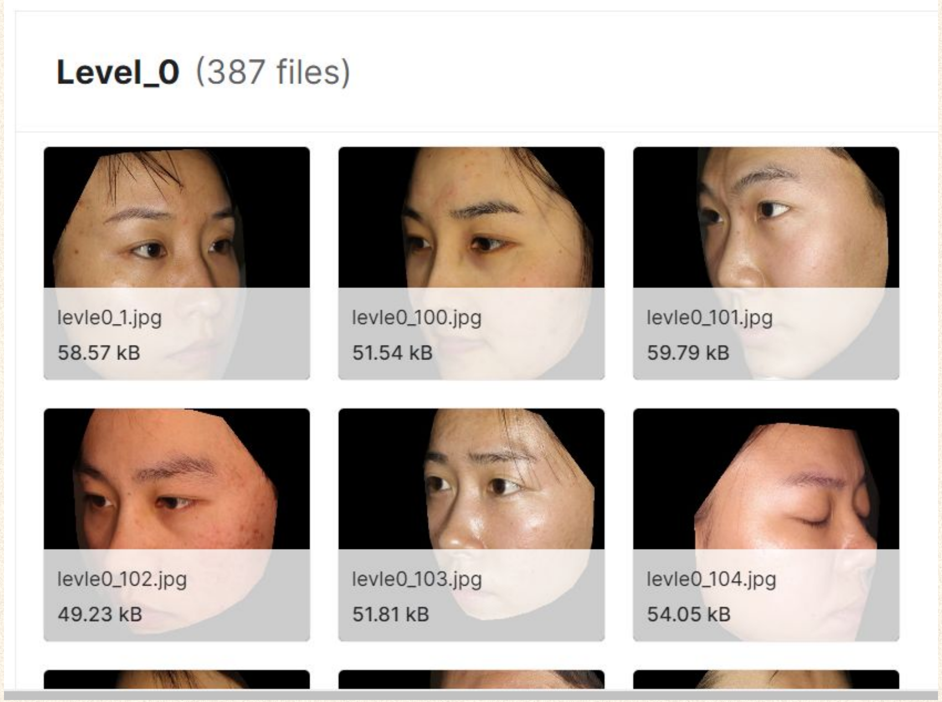
Level_1

473 files

Level_2

139 files

Level_0



Level_0 (387 files)

levle0_1.jpg
58.57 kB

levle0_100.jpg
51.54 kB

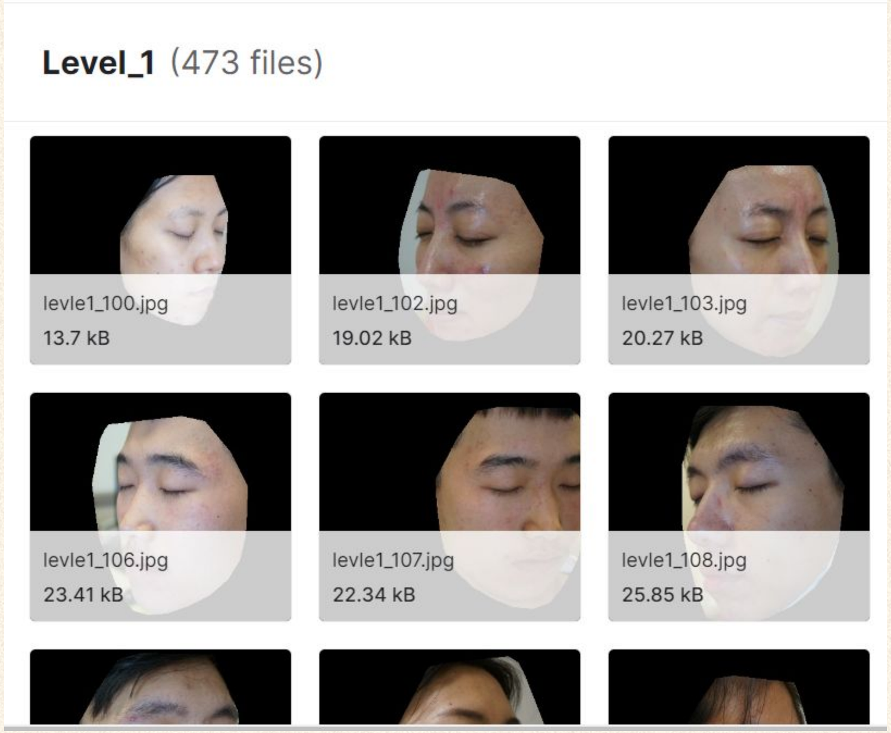
levle0_101.jpg
59.79 kB

levle0_102.jpg
49.23 kB

levle0_103.jpg
51.81 kB

levle0_104.jpg
54.05 kB

Level_1



Level_1 (473 files)

levle1_100.jpg
13.7 kB

levle1_102.jpg
19.02 kB

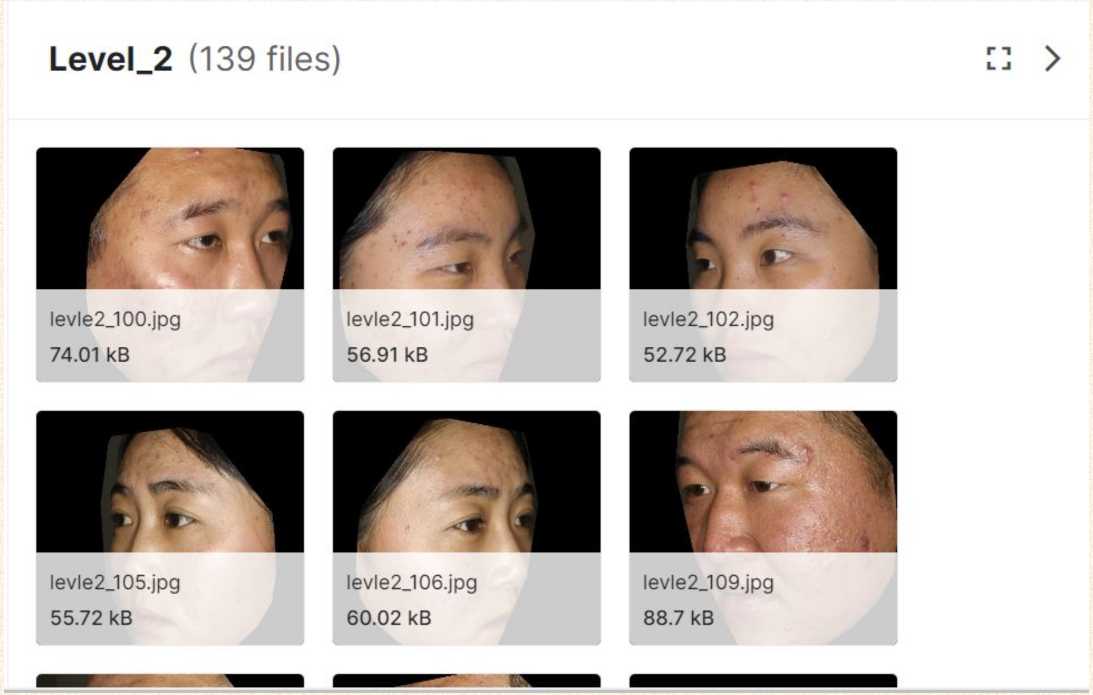
levle1_103.jpg
20.27 kB

levle1_106.jpg
23.41 kB

levle1_107.jpg
22.34 kB

levle1_108.jpg
25.85 kB

Level_2



Level_2 (139 files)

levle2_100.jpg
74.01 kB

levle2_101.jpg
56.91 kB

levle2_102.jpg
52.72 kB

levle2_105.jpg
55.72 kB

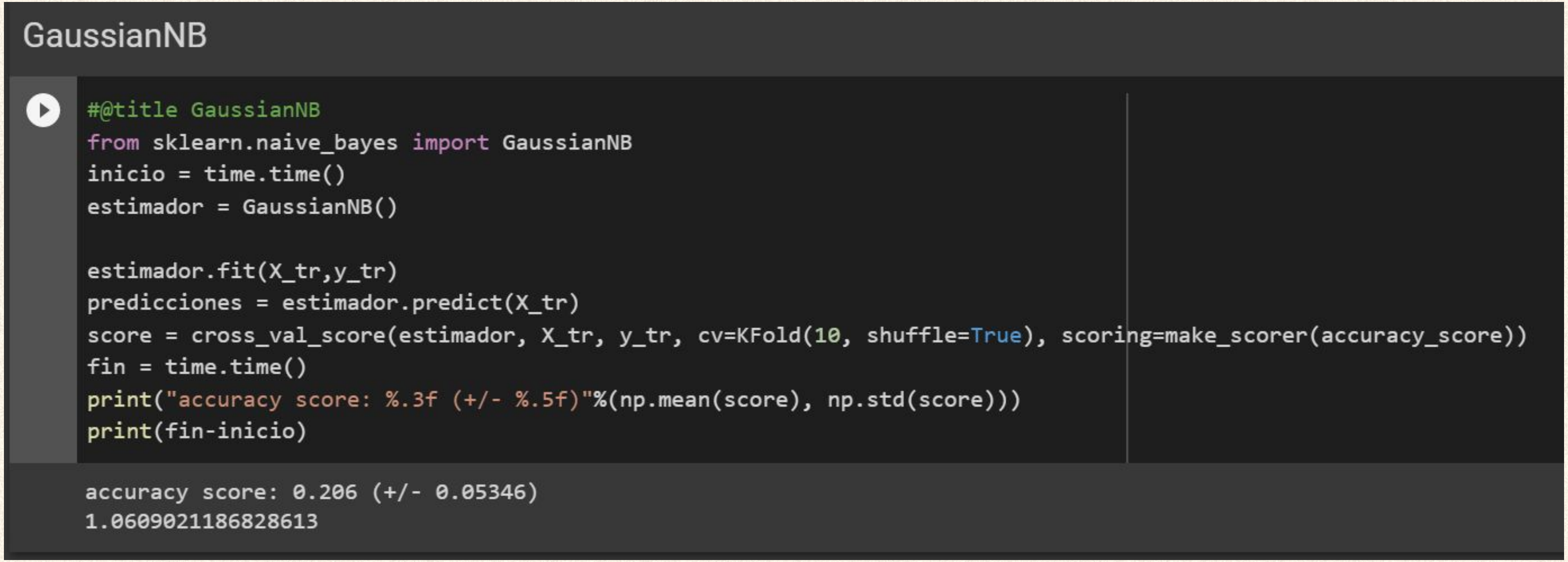
levle2_106.jpg
60.02 kB

levle2_109.jpg
88.7 kB

03

Modelos de clasificación

GaussianNB



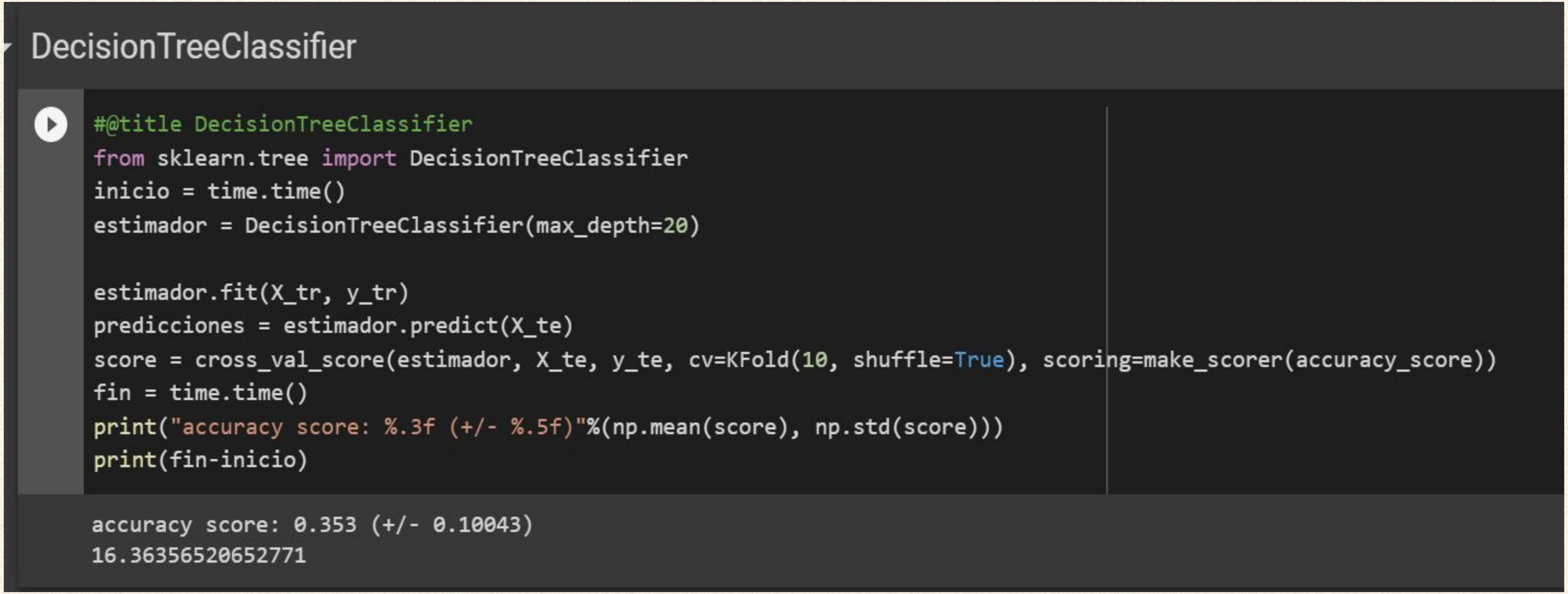
GaussianNB

```
#@title GaussianNB
from sklearn.naive_bayes import GaussianNB
inicio = time.time()
estimador = GaussianNB()

estimador.fit(X_tr,y_tr)
predicciones = estimador.predict(X_tr)
score = cross_val_score(estimador, X_tr, y_tr, cv=KFold(10, shuffle=True), scoring=make_scorer(accuracy_score))
fin = time.time()
print("accuracy score: %.3f (+/- %.5f)"%(np.mean(score), np.std(score)))
print(fin-inicio)
```

```
accuracy score: 0.206 (+/- 0.05346)
1.0609021186828613
```

Decision Tree Classifier



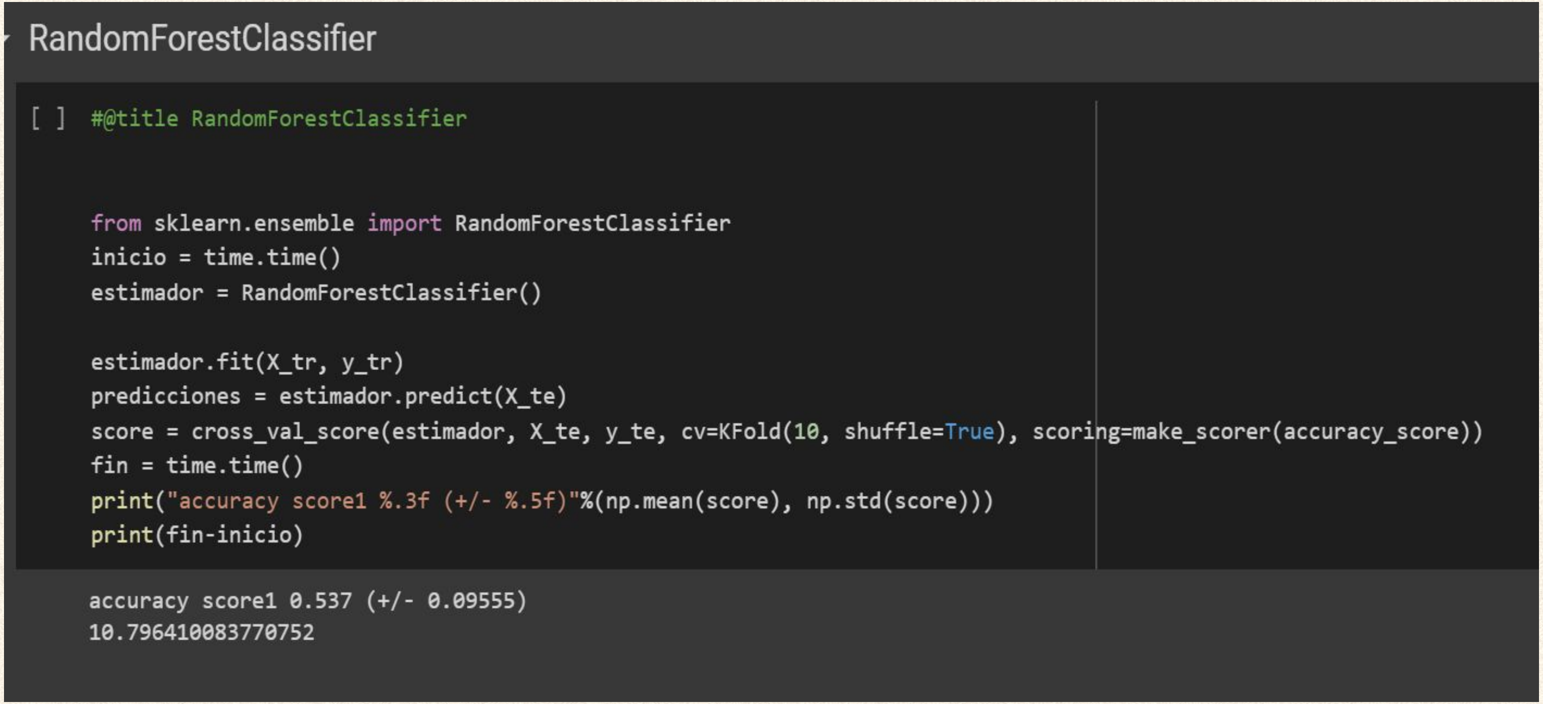
DecisionTreeClassifier

```
#@title DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
inicio = time.time()
estimador = DecisionTreeClassifier(max_depth=20)

estimador.fit(X_tr, y_tr)
predicciones = estimador.predict(X_te)
score = cross_val_score(estimador, X_te, y_te, cv=KFold(10, shuffle=True), scoring=make_scorer(accuracy_score))
fin = time.time()
print("accuracy score: %.3f (+/- %.5f)"%(np.mean(score), np.std(score)))
print(fin-inicio)
```

```
accuracy score: 0.353 (+/- 0.10043)
16.36356520652771
```

Random Forest Classifier



RandomForestClassifier

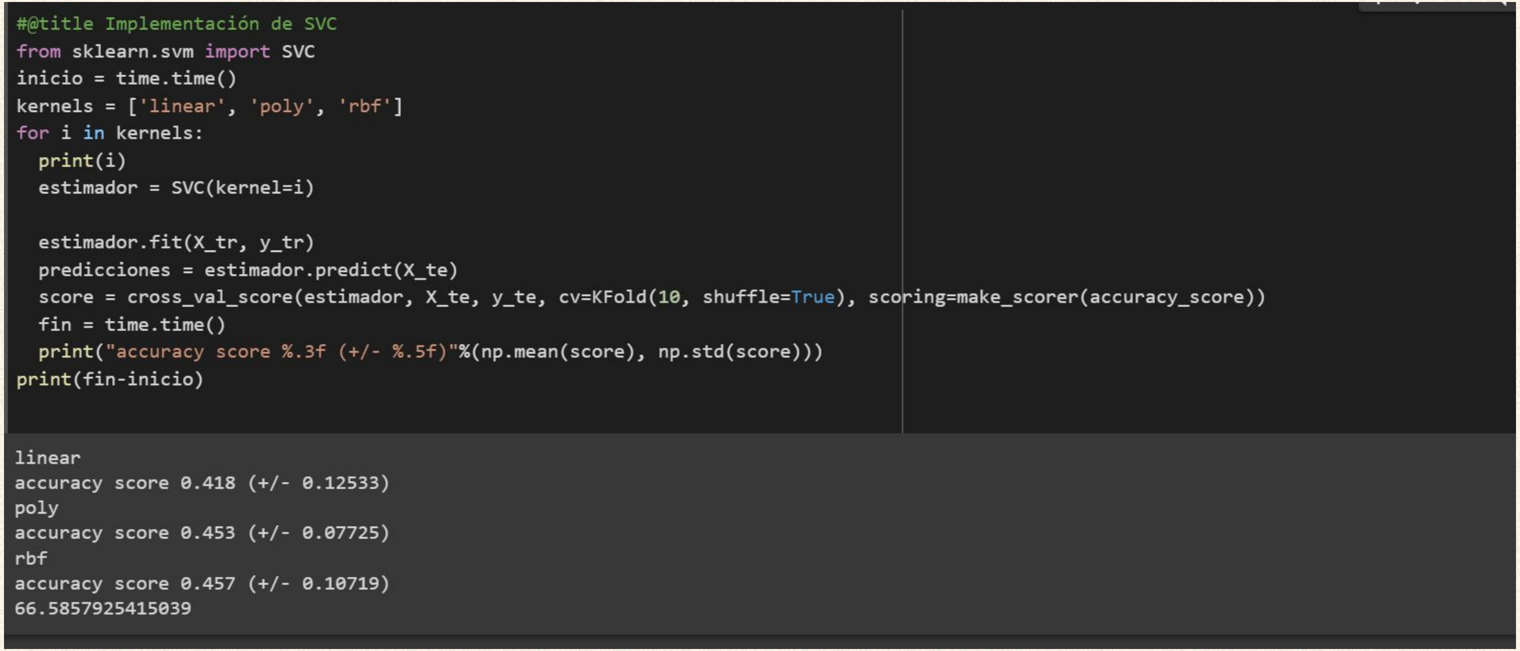
```
[ ]  #@title RandomForestClassifier


     from sklearn.ensemble import RandomForestClassifier
     inicio = time.time()
     estimador = RandomForestClassifier()

     estimador.fit(X_tr, y_tr)
     predicciones = estimador.predict(X_te)
     score = cross_val_score(estimador, X_te, y_te, cv=KFold(10, shuffle=True), scoring=make_scorer(accuracy_score))
     fin = time.time()
     print("accuracy score1 %.3f (+/- %.5f)"%(np.mean(score), np.std(score)))
     print(fin-inicio)
```

```
accuracy score1 0.537 (+/- 0.09555)
10.796410083770752
```

Implementación de SVC



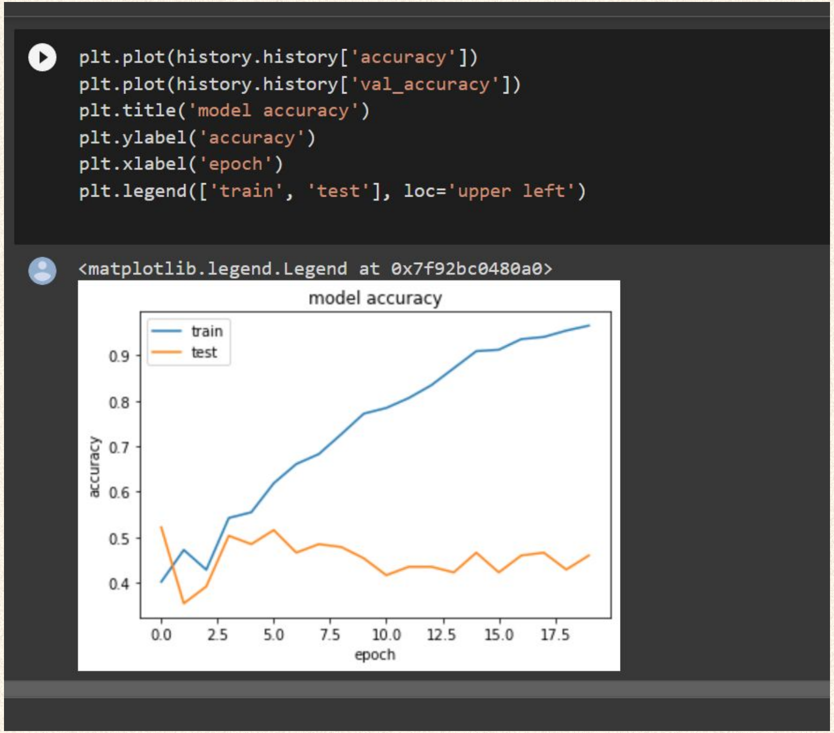
```
#@title Implementación de SVC
from sklearn.svm import SVC
inicio = time.time()
kernels = ['linear', 'poly', 'rbf']
for i in kernels:
  print(i)
  estimador = SVC(kernel=i)

  estimador.fit(X_tr, y_tr)
  predicciones = estimador.predict(X_te)
  score = cross_val_score(estimador, X_te, y_te, cv=KFold(10, shuffle=True), scoring=make_scorer(accuracy_score))
  fin = time.time()
  print("accuracy score %.3f (+/- %.5f)"%(np.mean(score), np.std(score)))
print(fin-inicio)
```

```
linear
accuracy score 0.418 (+/- 0.12533)
poly
accuracy score 0.453 (+/- 0.07725)
rbf
accuracy score 0.457 (+/- 0.10719)
66.5857925415039
```

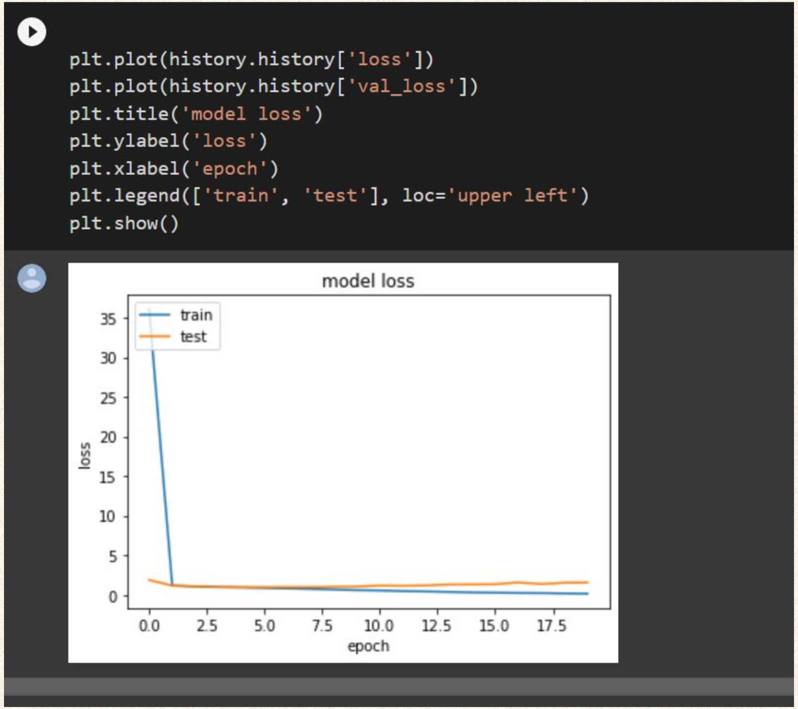
04

Deep Learning

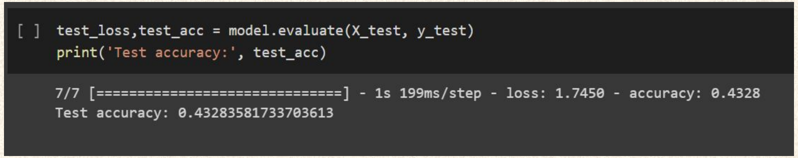


```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
```

```
<matplotlib.legend.Legend at 0x7f92bc0480a0>
```



```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



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

     7/7 [==============================] - 1s 199ms/step - loss: 1.7450 - accuracy: 0.4328
     Test accuracy: 0.43283581733703613
```

Aplicativilidad