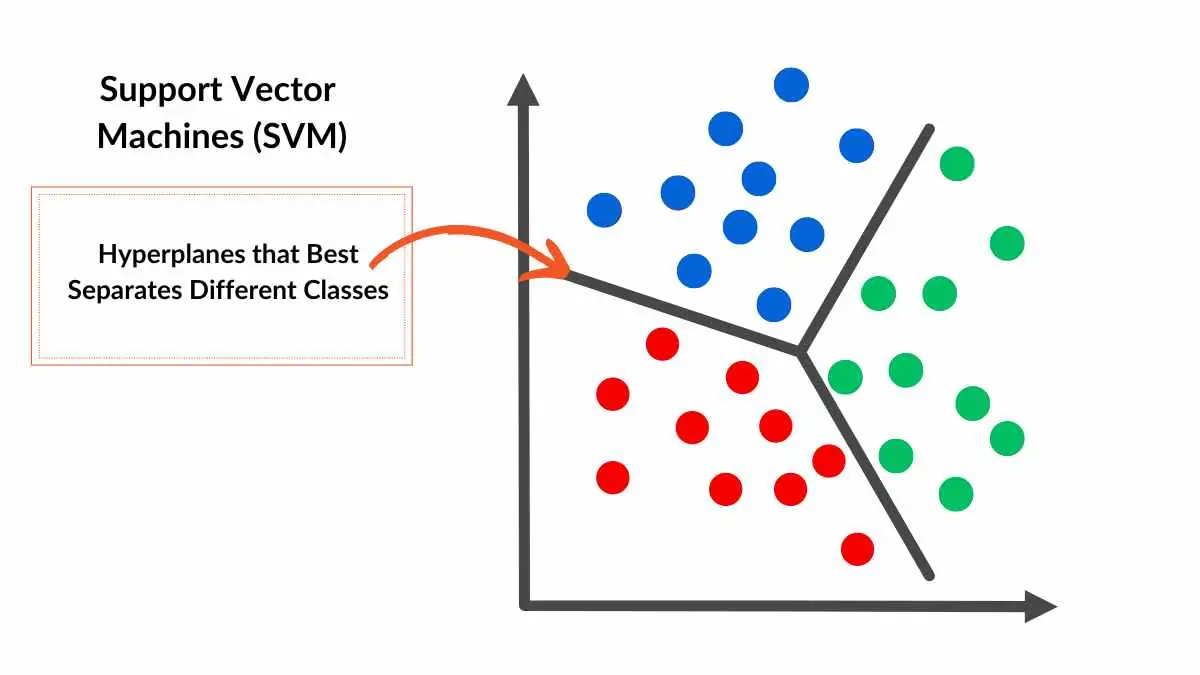
SVM



Adrián Yared Armas de la Nuez

**Contenido**

[**1. Enunciado 3**](#_2c35568x6wg8)

[**2. Explicación 3**](#_mptzybqzhbp3)

[**2.1 Librerías y dataset 3**](#_rjpj17ru25h3)

[**2.1.1 Código 3**](#_wty4o7wp6klp)

[**2.1.2 Resultado 3**](#_fmiqhou1ra7b)

[**2.2.1 Código 4**](#_jnipm3n8znm7)

[**2.2.2 Resultado 4**](#_kv8qoma0l3xs)

[**2.1.3 Explicación 4**](#_eks8p0i4ovxd)

[**3.1 Comprobación de datos perdidos 4**](#_52q2nik5ci9t)

[**3.1.1 Código 4**](#_n5zdf9qpfcjj)

[**3.1.2 Resultado 4**](#_ku5zzsaj6h0p)

[**3.2.1 Código 4**](#_1qlrvbe10pml)

[**3.2.2 Resultado 5**](#_ozq0edsq90yg)

[**3.3.1 Código 5**](#_5i56niuz90o6)

[**3.3.2 Resultado 6**](#_vikg01pety1f)

[**3.4.1 Código 6**](#_8ygk0o3rkcxq)

[**3.4.2 Resultado 7**](#_szep3lcvzf0l)

[**3.5.1 Código 7**](#_mxbtnktbaypv)

[**3.5.2 Resultado 7**](#_tm31k2ivujs7)

[**3.6.3 Explicación 8**](#_xqtkfb1ee2b6)

[**4.1 train\_test\_split usando 20% de los datos para la prueba 8**](#_6ro2307iwt27)

[**4.1.1 Código 8**](#_p91e8zbi9wzn)

[**4.1.2 Resultado 8**](#_fmum56z0pjfb)

[**4.2 Explicación 8**](#_9rf754pfnwqd)

[**5.1 Entrenamiento y predicción con Regresión Logística 9**](#_wtpru9g6gh4e)

[**5.1.1 Código 9**](#_kt4bvbxhgryb)

[**5.1.2 Resultado 9**](#_r0sbzwir0obr)

[**5.2 Explicación 9**](#_xzdfm4myfbu7)

[**6.1 Cálculo de métricas de evaluación 9**](#_tybhtlv718r2)

[**6.1.1 Código 9**](#_e4pv0oxht49h)

[**7.2 Explicación 10**](#_y2k9mg6tthee)

[**7.1 SGClassifier 10**](#_gechtpb5vxdj)

[**7.1.1 Código 10**](#_hkq9tu4u3jh3)

[**7.1.2 Resultado 11**](#_omdjhz5ptiw1)

[**7.2 Explicación 11**](#_q6mp0ek1r14l)

[**8.1 SVM 11**](#_sayuaai8oiee)

[**8.1.1 Código 11**](#_bpiq69v9bo8g)

[**8.1.2 Resultado 12**](#_qg7evlcbcv29)

[**8.1.1 Código 12**](#_y65fll77ew6u)

[**8.1.2 Resultado 13**](#_g06x0tm2f5fl)

[**8.2 Explicación 13**](#_e3hj1hoo66n3)

[**9.1 Matriz de confusión para LR y SVM 13**](#_vtbxf2jkxcle)

[**9.1.1 Código 13**](#_jwth6dz8qr84)

[**9.1.2 Resultado 14**](#_v1fmwli75mzq)

[**9.2 Explicación 14**](#_qwxkcyyu2xep)

[**10 Código completo 15**](#_qyd3x486w033)

[**11 Resultado submission.csv 18**](#_ci80hdnnbqq1)

[**12 Enlaces 18**](#_dnvh0mj4fhkq)

## 

## 

## 

## 

## 

## 

## 

## **1. Enunciado**

Usa un modelo de SVM <https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html>

sobre el data set de especies de flores Iris <https://www.kaggle.com/datasets/uciml/iris> (u otro de tu elección), para predecir la especie de una flor en función de los parámetros dados.

## **2. Explicación**

### **2.1 Librerías y dataset**

#### **2.1.1 Código**

**import** pandas **as** pd

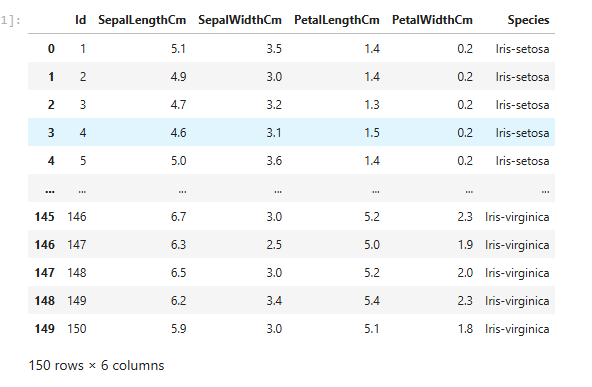
**.***read\_csv***(**'../input/ds-francis/iris.csv'**)**

df**from** matplotlib **import** pyplot **as** plt

**%**matplotlib inline

df**=** pd

#### **2.1.2 Resultado**

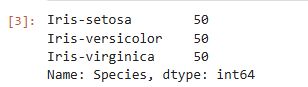


#### **2.2.1 Código**

df1 **=** df**.***drop***(**'Id'**,** axis**=**1**)**

df**[**'Species'**].***value\_counts***()**

#### **2.2.2 Resultado**



#### **2.1.3 Explicación**

f = pd.read\_csv('../input/ds-francis/iris.csv') carga el conjunto de datos iris.csv

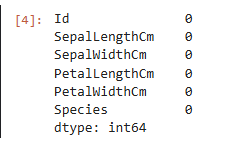
df['Species'].value\_counts() cuenta las instancias de cada especie para confirmar que hay 50 flores de cada especie, después se verifica la existencia de valores nulos en el conjunto de datos con df.isna().sum(), que devuelve 0 para cada columna.

### **3.1 Comprobación de datos perdidos**

#### **3.1.1 Código**

df**.***isna***().sum()**

#### **3.1.2 Resultado**



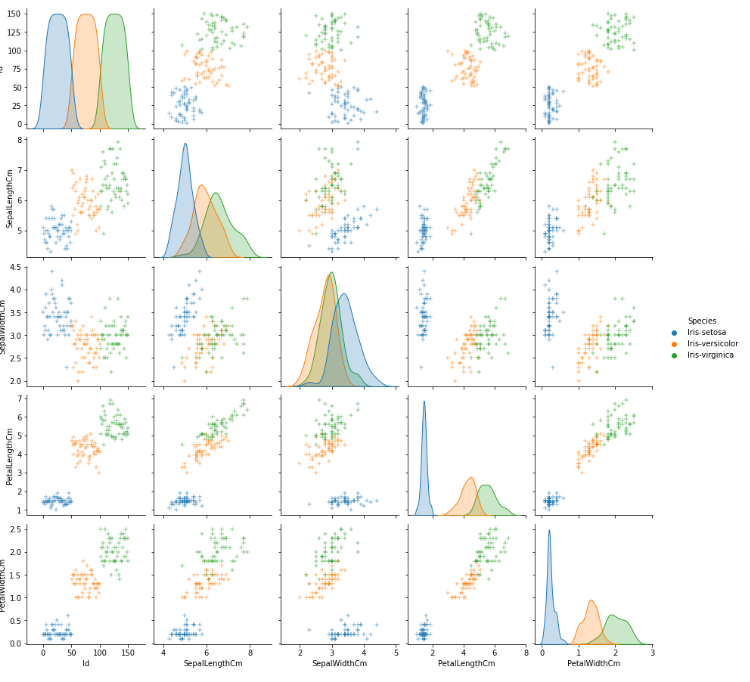
#### **3.2.1 Código**

**import** seaborn **as** sns

g **=** sns**.***pairplot***(**df**,** hue**=**'Species'**,** markers**=**'+'**)**

plt**.***show***()**

#### **3.2.2 Resultado**



#### **3.3.1 Código**

fig **=** df1**[**df1**.***Species***==**'Iris-setosa'**].***plot***(**kind**=**'scatter'**,**x**=**'SepalLengthCm'**,**y**=**'SepalWidthCm'**,**color**=**'red'**,** label**=**'Setosa'**)**

df1**[**df1**.***Species***==**'Iris-versicolor'**].***plot***(**kind**=**'scatter'**,**x**=**'SepalLengthCm'**,**y**=**'SepalWidthCm'**,**color**=**'blue'**,** label**=**'versicolor'**,**ax**=**fig**)**

df1**[**df1**.***Species***==**'Iris-virginica'**].***plot***(**kind**=**'scatter'**,**x**=**'SepalLengthCm'**,**y**=**'SepalWidthCm'**,**color**=**'green'**,** label**=**'virginica'**,** ax**=**fig**)**

fig**.***set\_xlabel***(**"Sepal Length"**)**

fig**.***set\_ylabel***(**"Sepal Width"**)**

fig**.***set\_title***(**"Sepal Length VS Width"**)**

fig**=**plt**.***gcf***()**

fig**.***set\_size\_inches***(**12**,**8**)**

plt**.***show***()**

#### **3.3.2 Resultado**

#### 

#### **3.4.1 Código**

**fig = df1[df1.*Species*=='Iris-setosa'].*plot*.*scatter*(x='PetalLengthCm',y='PetalWidthCm',color='red', label='Setosa')**

**df1[df1.*Species*=='Iris-versicolor'].*plot*.*scatter*(x='PetalLengthCm',y='PetalWidthCm',color='blue', label='versicolor',ax=fig)**

**df1[df1.*Species*=='Iris-virginica'].*plot*.*scatter*(x='PetalLengthCm',y='PetalWidthCm',color='green', label='virginica', ax=fig)**

**fig.*set\_xlabel*("Petal Length")**

**fig.*set\_ylabel*("Petal Width")**

**fig.*set\_title*(" Petal Length VS Width")**

**fig=plt.*gcf*()**

**fig.*set\_size\_inches*(12,8)**

**plt.*show*()**

#### **3.4.2 Resultado**

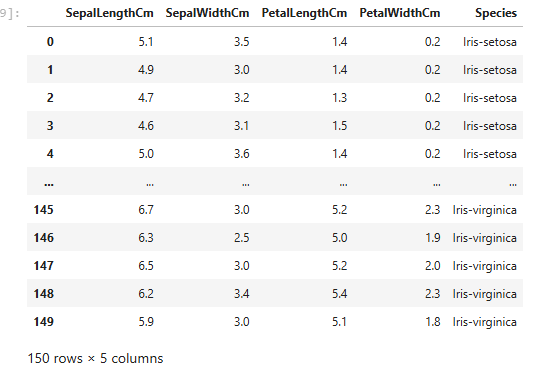
#### 

#### **3.5.1 Código**

df**.***drop***([**'Id'**]** **,**inplace **=True** **,** axis **=**1**)**

df

#### **3.5.2 Resultado**



#### **3.6 Explicación**

Creé un gráfico de pares (pairplot) para observar las relaciones entre las características SepalLengthCm, SepalWidthCm, PetalLengthCm, y PetalWidthCm en función de la especie de flor.

después, se generan gráficos de dispersión para observar la relación entre el ancho y largo de sépalo y pétalo.

### **4.1 train\_test\_split usando 20% de los datos para la prueba**

#### **4.1.1 Código**

**from** sklearn**.***model\_selection* **import** train\_test\_split

train**,** test **=** train\_test\_split**(**df**,** test\_size**=**0.2**)**

X\_train **=** train**.***drop***(**columns**=[**'Species'**],**axis**=**1**)**

y\_train **=** train**[**'Species'**]**

X\_train **=** X\_train **/**10

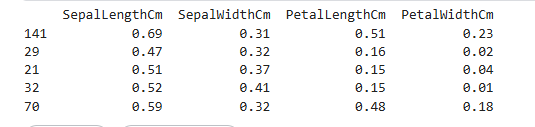
X\_test **=** test**.***drop***(**columns**=[**'Species'**],**axis**=**1**)**

y\_test **=** test**[**'Species'**]**

X\_test **/=**10

**print(**X\_train**.***head***())**

#### **4.1.2 Resultado**



#### **4.2 Explicación**

Se dividen los datos de entrenamiento y test con sklearn.

X\_train y X\_test contienen solo las características numéricas de cada flor, mientras que y\_train y y\_test contienen las etiquetas de especie. Se dividen las características para sacar un número normal.

### **5.1 Entrenamiento y predicción con Regresión Logística**

#### **5.1.1 Código**

**from** sklearn**.***linear\_model* **import** SGDClassifier

**from** sklearn**.***linear\_model* **import** LogisticRegression

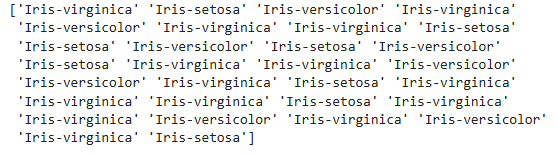
LR\_model **=** LogisticRegression**()**

LR\_model**.***fit***(**X\_train**,**y\_train**)**

LR\_predict **=** LR\_model**.***predict***(**X\_test**)**

**print(**LR\_predict**)**

#### **5.1.2 Resultado**



#### **5.2 Explicación**

Se entrena un modelo de Regresión Logística (LogisticRegression) y se realiza la predicción (LR\_predict) sobre los datos de prueba.

Se calculan métricas de rendimiento: accuracy, precision, recall, f1\_score, y cross\_val\_score.

### **6.1 Cálculo de métricas de evaluación**

#### **6.1.1 Código**

from sklearn.*metrics* import accuracy\_score,precision\_score,recall\_score,f1\_score

print(accuracy\_score(LR\_predict, y\_test))

print(precision\_score(LR\_predict, y\_test,average='macro'))

print(recall\_score(LR\_predict, y\_test,average='macro'))

print(f1\_score(LR\_predict, y\_test,average='macro'))

# Result

# 0.8666666666666667

# 0.8722222222222222

# 0.8869047619047619

# 0.8746438746438746

from sklearn.*model\_selection* import cross\_val\_score

print( cross\_val\_score(LR\_model,X\_train,y\_train,cv=5,scoring='accuracy'))

# Result [0.79166667 0.95833333 0.875 0.83333333 0.95833333]

#### 

#### **6.2 Explicación**

Esta parte mide qué tan bien funciona el modelo de **Regresión Logística** al predecir las especies de flores en el conjunto de prueba.

accuracy\_score: mide el porcentaje total de predicciones correctas.

precision\_score, recall\_score, y f1\_score: son métricas que evalúan el balance entre predicciones correctas y errores

cross\_val\_score comprueba la estabilidad del rendimiento en cda grupo de 5 conjuntos.

### 

### **7.1 SGClassifier**

#### **7.1.1 Código**

**from sklearn.*linear\_model* import SGDClassifier**

**from sklearn.*svm* import SVC**

**smv\_model = SVC()**

**smv\_model.*fit*(X\_test, y\_test)**

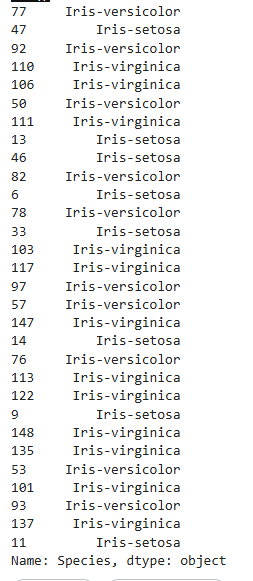
**# Result SVC()**

**smv\_predict = smv\_model.*fit*(X\_test, y\_test)**

**print(smv\_predict)**

**print(y\_test)**

#### **7.1.2 Resultado**



#### **7.2 Explicación**

Esta parte del código entrena y evalúa un modelo de **Máquinas de Vectores de Soporte** (SVM) usando SVC() para clasificar las especies de flores en el conjunto de prueba (X\_test y y\_test)

### **8.1 SVM**

#### **8.1.1 Código**

**from sklearn.*linear\_model* import SGDClassifier**

**from sklearn.*svm* import SVC**

**smv\_model = SVC()**

**smv\_model.*fit*(X\_test, y\_test)**

**#Result SVC()**

**accuracy\_score = smv\_model.*score*(X\_test,y\_test)**

**print(accuracy\_score)**

**#Result 0.9666666666666667**

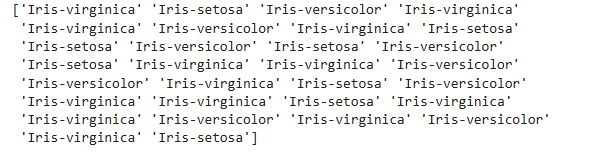
**from sklearn import metrics**

**from sklearn.*metrics* import classification\_report**

**predict = smv\_model.*predict*(X\_test)**

**print(predict)**

#### **8.1.2 Resultado**



#### **8.1.1 Código**

**from sklearn.*metrics* import classification\_report**

**ClassR= metrics.*classification\_report*(y\_test, predict)**

**print(ClassR)**

#### **8.1.2 Resultado**

#### 

#### **8.2 Explicación**

Se imprimen las matrices de confusión para ambos modelos (LR y SVM) para evaluar cómo cada modelo clasifica las especies de iris y el resultado se imprime en submission.csv.

### **9.1 Matriz de confusión para LR y SVM**

#### **9.1.1 Código**

**from sklearn import metrics**

**from sklearn.*metrics* import confusion\_matrix**

**print(y\_test)**

**print('LR: \n', confusion\_matrix(LR\_predict,y\_test))**

**print('SMV: \n', confusion\_matrix(LR\_predict,y\_test))**

**df.*to\_csv*('submission.csv', index=None)**

#### **9.1.2 Resultado**

#### 

#### **9.2 Explicación**

Compara las predicciones de los modelos con los valores reales (y\_test) para mostrar el número de aciertos y errores por clase.

Hay un error porque usa las predicciones de Regresión Logística (LR\_predict) en ambas matrices.

### **10 Código completo**

**import** pandas **as** pd

**from** matplotlib **import** pyplot **as** plt

**%**matplotlib inline

df**=** pd**.***read\_csv***(**'../input/ds-francis/iris.csv'**)**

df

df1 **=** df**.***drop***(**'Id'**,** axis**=**1**)**

df**[**'Species'**].***value\_counts***()**

df**.***isna***().sum()**

**import** seaborn **as** sns

g **=** sns**.***pairplot***(**df**,** hue**=**'Species'**,** markers**=**'+'**)**

plt**.***show***()**

fig **=** df1**[**df1**.***Species***==**'Iris-setosa'**].***plot***(**kind**=**'scatter'**,**x**=**'SepalLengthCm'**,**y**=**'SepalWidthCm'**,**color**=**'red'**,** label**=**'Setosa'**)**

df1**[**df1**.***Species***==**'Iris-versicolor'**].***plot***(**kind**=**'scatter'**,**x**=**'SepalLengthCm'**,**y**=**'SepalWidthCm'**,**color**=**'blue'**,** label**=**'versicolor'**,**ax**=**fig**)**

df1**[**df1**.***Species***==**'Iris-virginica'**].***plot***(**kind**=**'scatter'**,**x**=**'SepalLengthCm'**,**y**=**'SepalWidthCm'**,**color**=**'green'**,** label**=**'virginica'**,** ax**=**fig**)**

fig**.***set\_xlabel***(**"Sepal Length"**)**

fig**.***set\_ylabel***(**"Sepal Width"**)**

fig**.***set\_title***(**"Sepal Length VS Width"**)**

fig**=**plt**.***gcf***()**

fig**.***set\_size\_inches***(**12**,**8**)**

plt**.***show***()**

fig **=** df1**[**df1**.***Species***==**'Iris-setosa'**].***plot***.***scatter***(**x**=**'PetalLengthCm'**,**y**=**'PetalWidthCm'**,**color**=**'red'**,** label**=**'Setosa'**)**

df1**[**df1**.***Species***==**'Iris-versicolor'**].***plot***.***scatter***(**x**=**'PetalLengthCm'**,**y**=**'PetalWidthCm'**,**color**=**'blue'**,** label**=**'versicolor'**,**ax**=**fig**)**

df1**[**df1**.***Species***==**'Iris-virginica'**].***plot***.***scatter***(**x**=**'PetalLengthCm'**,**y**=**'PetalWidthCm'**,**color**=**'green'**,** label**=**'virginica'**,** ax**=**fig**)**

fig**.***set\_xlabel***(**"Petal Length"**)**

fig**.***set\_ylabel***(**"Petal Width"**)**

fig**.***set\_title***(**" Petal Length VS Width"**)**

fig**=**plt**.***gcf***()**

fig**.***set\_size\_inches***(**12**,**8**)**

plt**.***show***()**

df**.***drop***([**'Id'**]** **,**inplace **=True** **,** axis **=**1**)**

df

**from** sklearn**.***model\_selection* **import** train\_test\_split

train**,** test **=** train\_test\_split**(**df**,** test\_size**=**0.2**)**

X\_train **=** train**.***drop***(**columns**=[**'Species'**],**axis**=**1**)**

y\_train **=** train**[**'Species'**]**

X\_train **=** X\_train **/**10

X\_test **=** test**.***drop***(**columns**=[**'Species'**],**axis**=**1**)**

y\_test **=** test**[**'Species'**]**

X\_test **/=**10

**print(**X\_train**.***head***())**

**from** sklearn**.***linear\_model* **import** SGDClassifier

**from** sklearn**.***linear\_model* **import** LogisticRegression

LR\_model **=** LogisticRegression**()**

LR\_model**.***fit***(**X\_train**,**y\_train**)**

LR\_predict **=** LR\_model**.***predict***(**X\_test**)**

**print(**LR\_predict**)**

**from** sklearn**.***metrics* **import** accuracy\_score**,**precision\_score**,**recall\_score**,**f1\_score

**print(**accuracy\_score**(**LR\_predict**,** y\_test**))**

**print(**precision\_score**(**LR\_predict**,** y\_test**,**average**=**'macro'**))**

**print(**recall\_score**(**LR\_predict**,** y\_test**,**average**=**'macro'**))**

**print(**f1\_score**(**LR\_predict**,** y\_test**,**average**=**'macro'**))**

# Result

# 0.8666666666666667

# 0.8722222222222222

# 0.8869047619047619

# 0.8746438746438746

**from** sklearn**.***model\_selection* **import** cross\_val\_score

**print(** cross\_val\_score**(**LR\_model**,**X\_train**,**y\_train**,**cv**=**5**,**scoring**=**'accuracy'**))**

# Result [0.79166667 0.95833333 0.875 0.83333333 0.95833333]

**from** sklearn**.***linear\_model* **import** SGDClassifier

**from** sklearn**.***svm* **import** SVC

smv\_model **=** SVC**()**

smv\_model**.***fit***(**X\_test**,** y\_test**)**

# Result SVC()

smv\_predict **=** smv\_model**.***fit***(**X\_test**,** y\_test**)**

**print(**smv\_predict**)**

**print(**y\_test**)**

**from** sklearn**.***linear\_model* **import** SGDClassifier

**from** sklearn**.***svm* **import** SVC

smv\_model **=** SVC**()**

smv\_model**.***fit***(**X\_test**,** y\_test**)**

#Result SVC()

accuracy\_score **=** smv\_model**.***score***(**X\_test**,**y\_test**)**

**print(**accuracy\_score**)**

#Result 0.9666666666666667

**from** sklearn **import** metrics

**from** sklearn**.***metrics* **import** classification\_report

predict **=** smv\_model**.***predict***(**X\_test**)**

**print(**predict**)**

**from** sklearn**.***metrics* **import** classification\_report

ClassR**=** metrics**.***classification\_report***(**y\_test**,** predict**)**

**print(**ClassR**)**

**from** sklearn **import** metrics

**from** sklearn**.***metrics* **import** confusion\_matrix

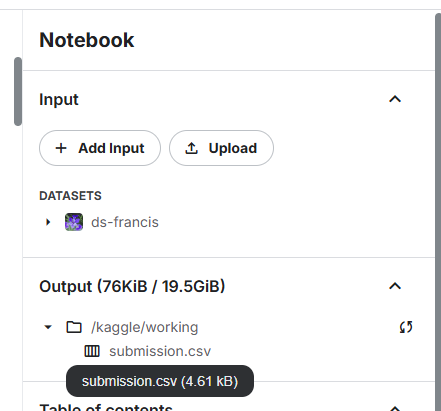
**print(**y\_test**)**

**print(**'LR: \n'**,** confusion\_matrix**(**LR\_predict**,**y\_test**))**

**print(**'SMV: \n'**,** confusion\_matrix**(**LR\_predict**,**y\_test**))**

df**.***to\_csv***(**'submission.csv'**,** index**=None)**

### **11 Resultado submission.csv**



### **12 Enlaces**

[](https://www.kaggle.com/code/adrian0120/notebookc707a43ed5)