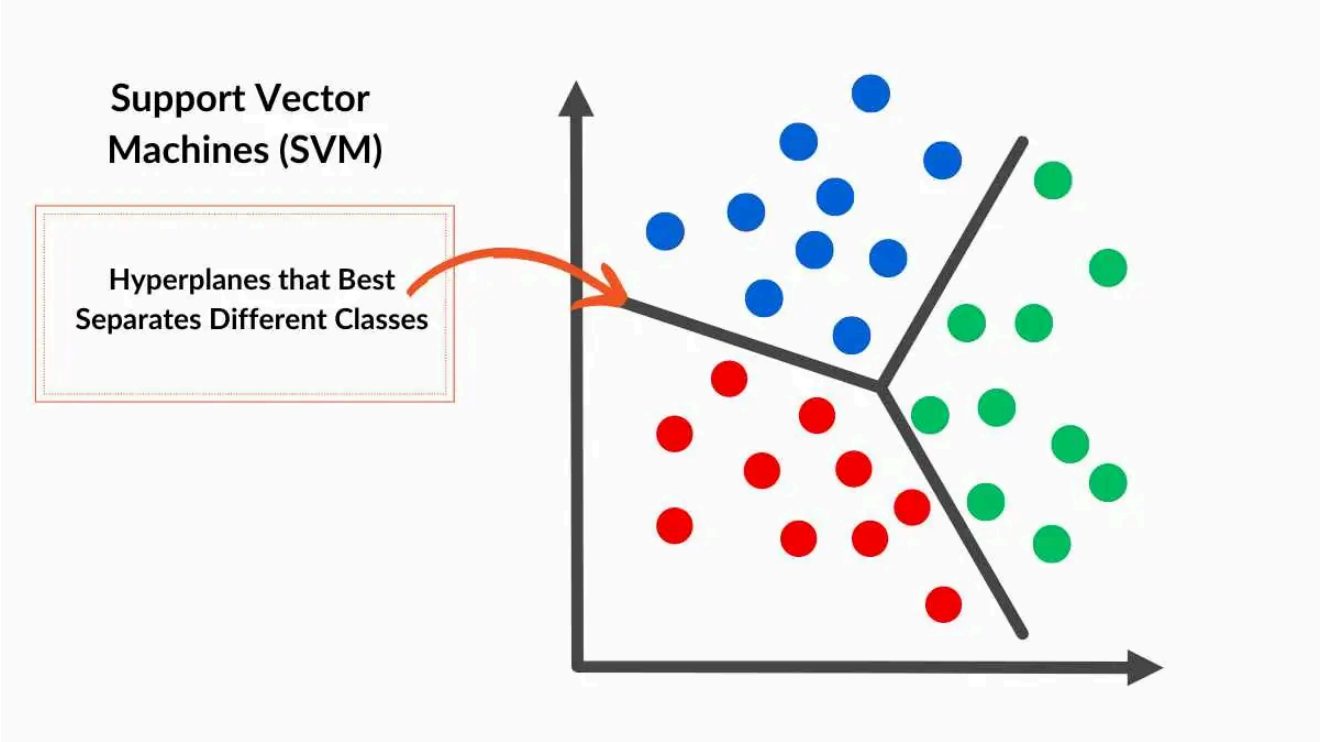
SVM



Adrián Yared Armas de la Nuez

Contenido

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')
dffrom matplotlib import pyplot as plt
%matplotlib inline
df= pd
```

2.1.2 Resultado

1]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

2.2.1 Código

```
df1 = df.drop('Id', axis=1)
df['Species'].value_counts()
```

2.2.2 Resultado

```
[3]: Iris-setosa        50
     Iris-versicolor    50
     Iris-virginica     50
     Name: Species, dtype: int64
```

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()
```

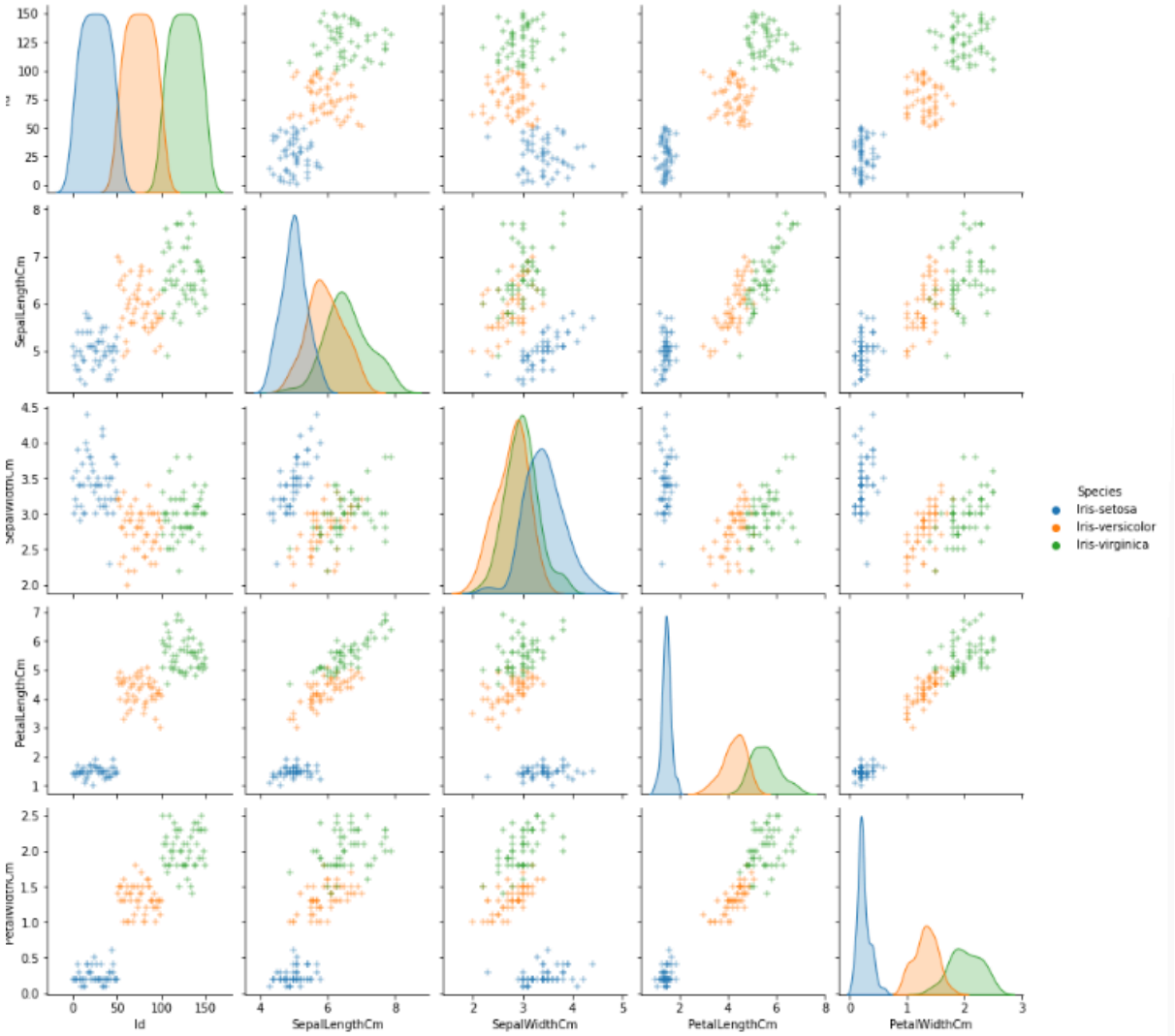
3.1.2 Resultado

```
[4]: Id              0
     SepalLengthCm   0
     SepalWidthCm    0
     PetalLengthCm   0
     PetalWidthCm    0
     Species         0
     dtype: int64
```

3.2.1 Código

```
import seaborn as sns
g = sns.pairplot(df, hue='Species', markers='+')
plt.show()
```

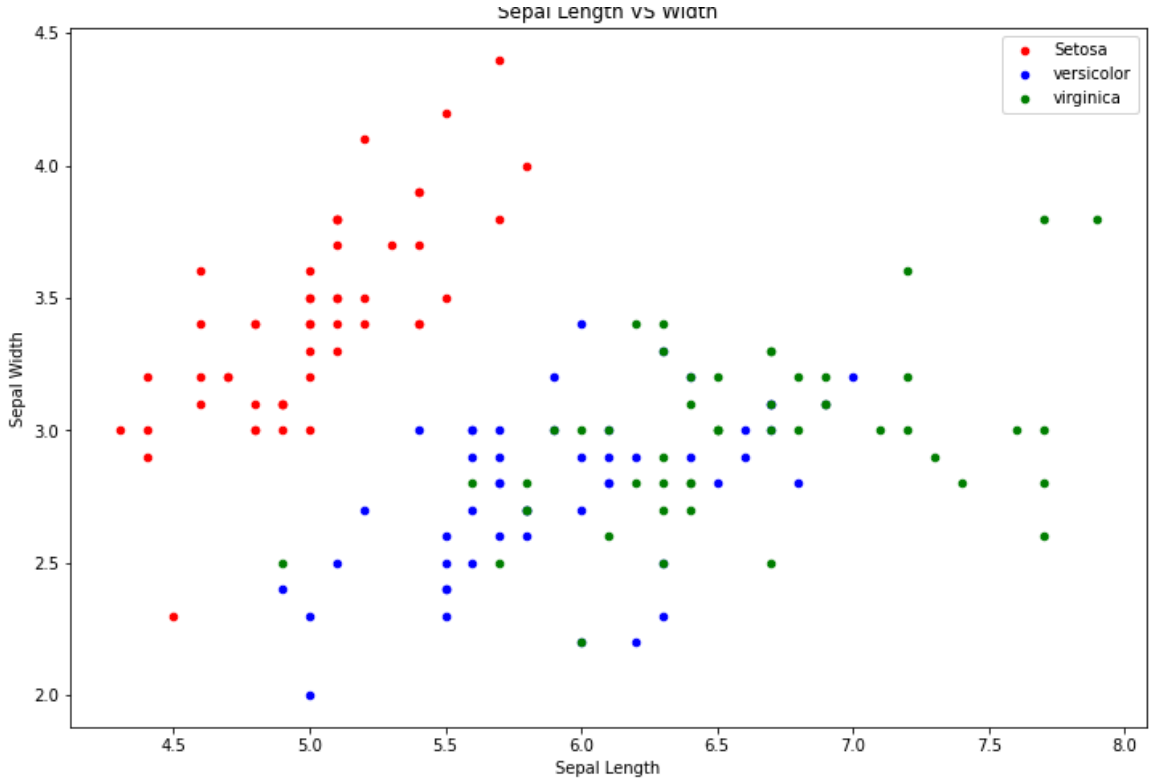
3.2.2 Resultado



3.3.1 Código

```
fig =
df1[df1.Species=='Iris-setosa'].plot(kind='scatter',x='SepalLengthCm',y='Sep
alWidthCm',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()
```

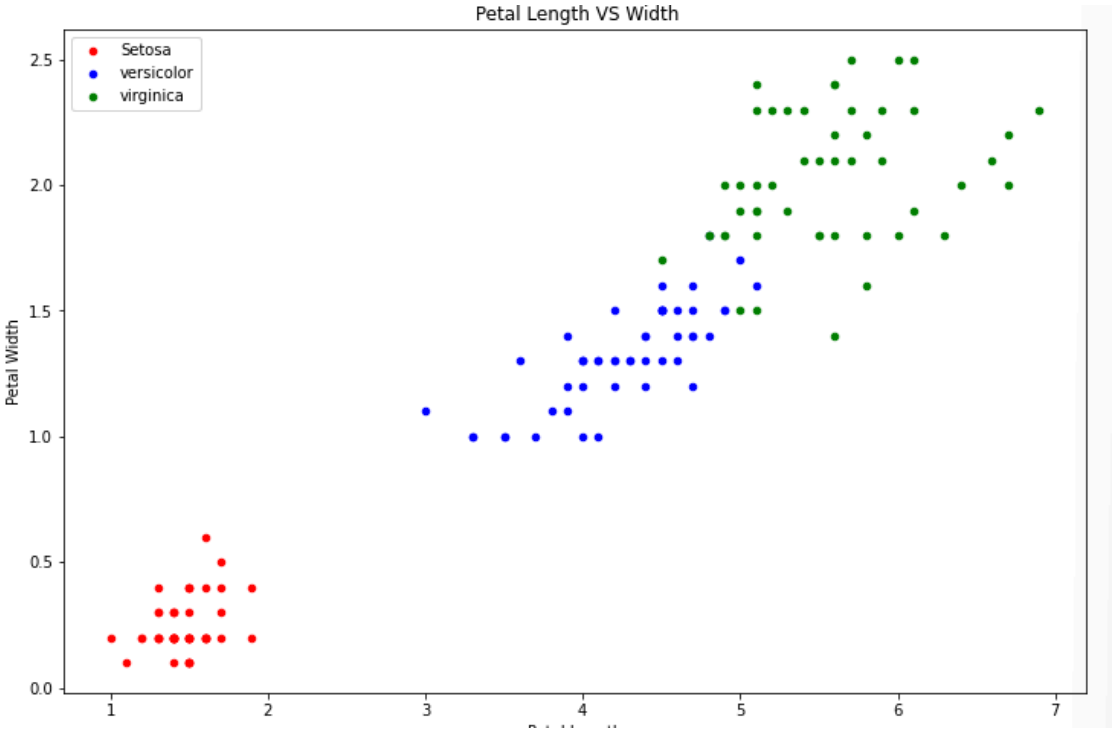
3.3.2 Resultado



3.4.1 Código

```
fig =
df1[df1.Species=='Iris-setosa'].plot.scatter(x='PetalLengthCm',y='PetalWidt
hCm',color='red', label='Setosa')
df1[df1.Species=='Iris-versicolor'].plot.scatter(x='PetalLengthCm',y='Petal
WidthCm',color='blue', label='versicolor',ax=fig)
df1[df1.Species=='Iris-virginica'].plot.scatter(x='PetalLengthCm',y='PetalW
idthCm',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

9]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

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

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
141	0.69	0.31	0.51	0.23
29	0.47	0.32	0.16	0.02
21	0.51	0.37	0.15	0.04
32	0.52	0.41	0.15	0.01
70	0.59	0.32	0.48	0.18

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

```
['Iris-virginica' 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica'
 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-virginica' 'Iris-setosa']
```

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

```
77     Iris-versicolor
47         Iris-setosa
92     Iris-versicolor
110     Iris-virginica
106     Iris-virginica
50     Iris-versicolor
111     Iris-virginica
13         Iris-setosa
46         Iris-setosa
82     Iris-versicolor
6          Iris-setosa
78     Iris-versicolor
33         Iris-setosa
103     Iris-virginica
117     Iris-virginica
97     Iris-versicolor
57     Iris-versicolor
147     Iris-virginica
14         Iris-setosa
76     Iris-versicolor
113     Iris-virginica
122     Iris-virginica
9          Iris-setosa
148     Iris-virginica
135     Iris-virginica
53     Iris-versicolor
101     Iris-virginica
93     Iris-versicolor
137     Iris-virginica
11         Iris-setosa
Name: Species, dtype: object
```

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

```
['Iris-virginica' 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa'
 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor'
 'Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-virginica' 'Iris-setosa']
```

8.1.1 Código

```
from sklearn.metrics import classification_report

ClassR= metrics.classification_report(y_test, predict)
print(ClassR)
```

8.1.2 Resultado

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         8
Iris-versicolor       1.00      0.90      0.95        10
 Iris-virginica       0.92      1.00      0.96        12

       accuracy                           0.97        30
      macro avg       0.97      0.97      0.97        30
   weighted avg       0.97      0.97      0.97        30
```

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

```
77     Iris-versicolor
47         Iris-setosa
92     Iris-versicolor
110     Iris-virginica
106     Iris-virginica
50     Iris-versicolor
111     Iris-virginica
13         Iris-setosa
46         Iris-setosa
82     Iris-versicolor
6          Iris-setosa
78     Iris-versicolor
33         Iris-setosa
103     Iris-virginica
117     Iris-virginica
97     Iris-versicolor
57     Iris-versicolor
147     Iris-virginica
14         Iris-setosa
76     Iris-versicolor
113     Iris-virginica
122     Iris-virginica
9          Iris-setosa
148     Iris-virginica
135     Iris-virginica
53     Iris-versicolor
101     Iris-virginica
93     Iris-versicolor
137     Iris-virginica
11         Iris-setosa
Name: Species, dtype: object
LR:
 [[ 8  0  0]
 [ 0  7  1]
 [ 0  3 11]]
SMV:
 [[ 8  0  0]
 [ 0  7  1]
 [ 0  3 11]]
```

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='Sep
alWidthCm',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='PetalWidth
Cm',color='red', label='Setosa')
df1[df1.Species=='Iris-versicolor'].plot.scatter(x='PetalLengthCm',y='PetalW
idthCm',color='blue', label='versicolor',ax=fig)
df1[df1.Species=='Iris-virginica'].plot.scatter(x='PetalLengthCm',y='PetalWi
dthCm',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]
```

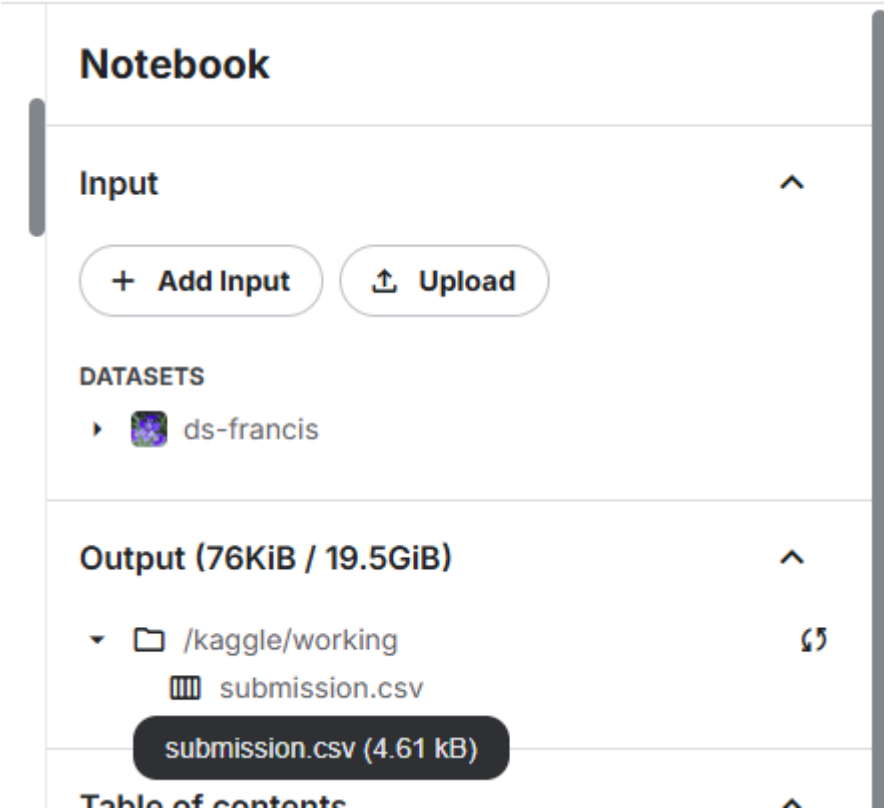
```
from sklearn.linear_model import SGDClassifier
from sklearn.svm import SVC

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