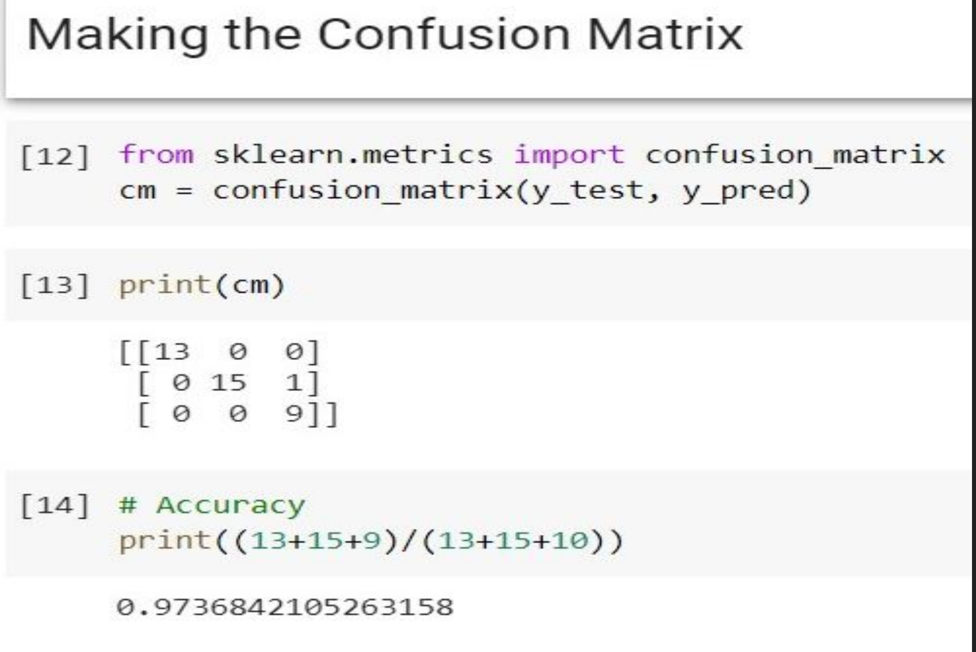
# Comparación Modelos (SNA/iris dataset)

Daniel López,
Jean Pierre Mejicanos,
Gulio Valenzuela,
Luis Pedro Zenteno
Sergio Cuevas

# Decision Tree Classification

## Iris



### Making the Confusion Matrix

```
[12] from sklearn.metrics import confusion_matrix
     cm = confusion_matrix(y_test, y_pred)
```

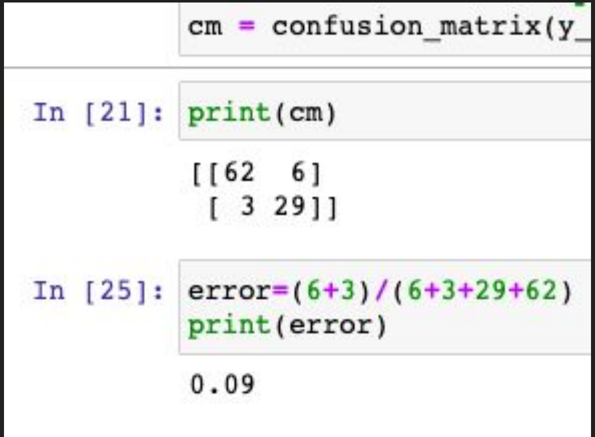
```
[13] print(cm)
```

```
     [[13  0  0]
      [ 0 15  1]
      [ 0  0  9]]
```

```
[14] # Accuracy
     print((13+15+9)/(13+15+10))
```

```
     0.9736842105263158
```

## Social Network Ads



```
          cm = confusion_matrix(y_
```

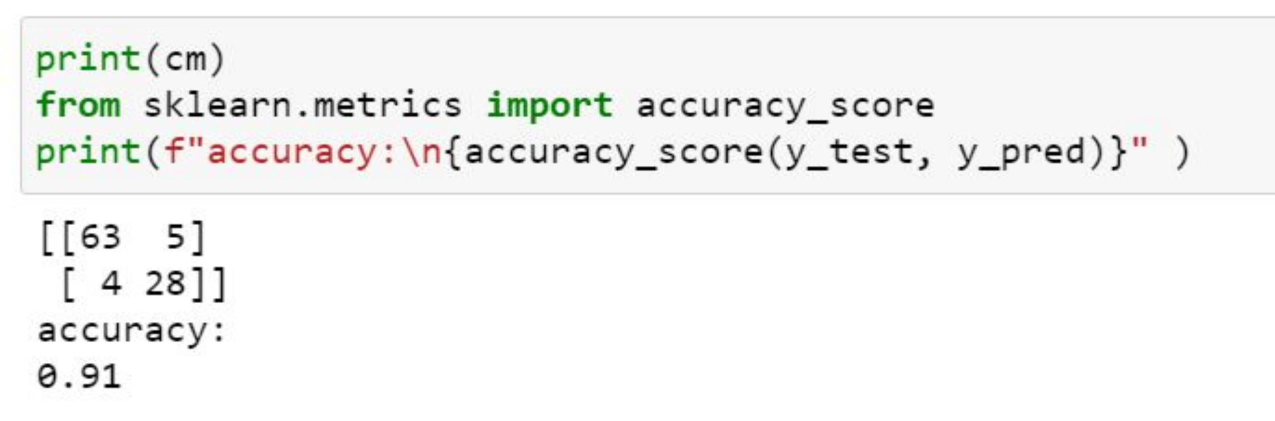
```
In [21]: print(cm)
```

```
         [[62  6]
          [ 3 29]]
```

```
In [25]: error=(6+3)/(6+3+29+62)
         print(error)
```

```
         0.09
```

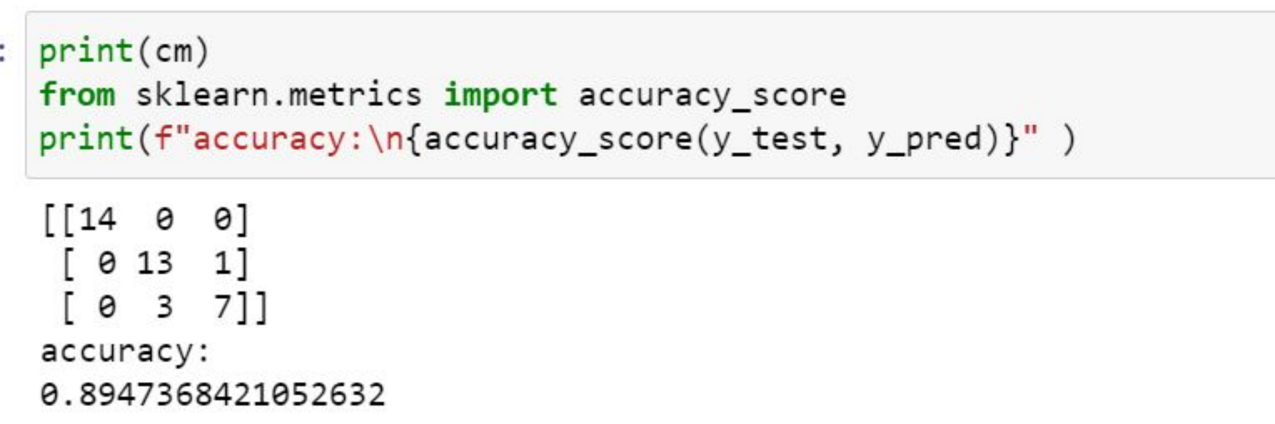
# Random Forest Classification



```
print(cm)
from sklearn.metrics import accuracy_score
print(f"accuracy:\n{accuracy_score(y_test, y_pred)}" )
```

```
[[63  5]
 [ 4 28]]
accuracy:
0.91
```

Social Network Ads



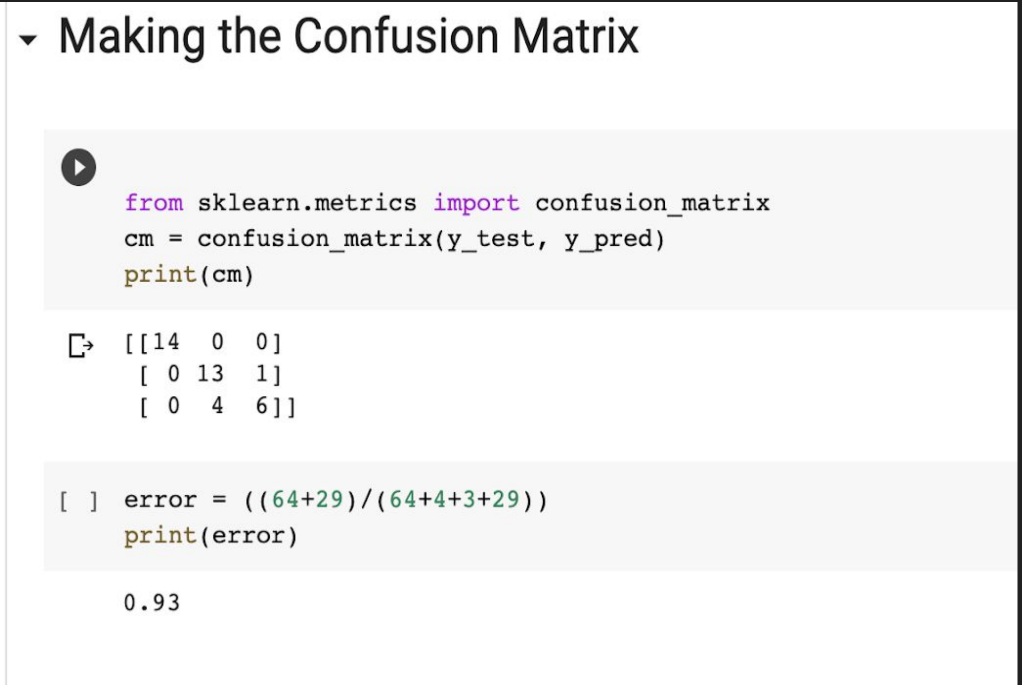
```
print(cm)
from sklearn.metrics import accuracy_score
print(f"accuracy:\n{accuracy_score(y_test, y_pred)}" )
```

```
[[14  0  0]
 [ 0 13  1]
 [ 0  3  7]]
accuracy:
0.8947368421052632
```

Iris

# K-Nearest Neighbours

Iris



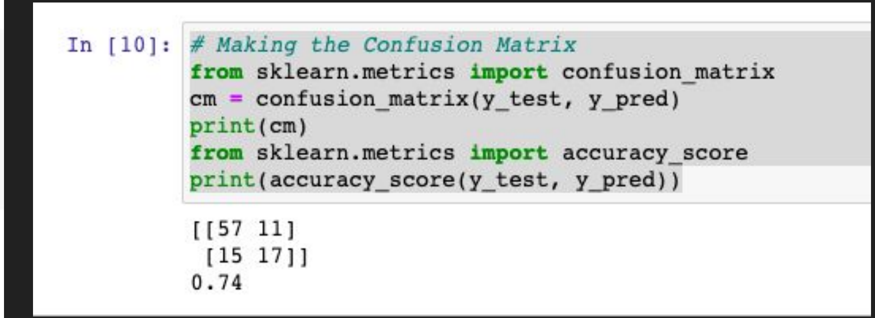
## Making the Confusion Matrix

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[14  0  0]
 [ 0 13  1]
 [ 0  4  6]]
```

```
error = ((64+29)/(64+4+3+29))
print(error)
```

```
0.93
```

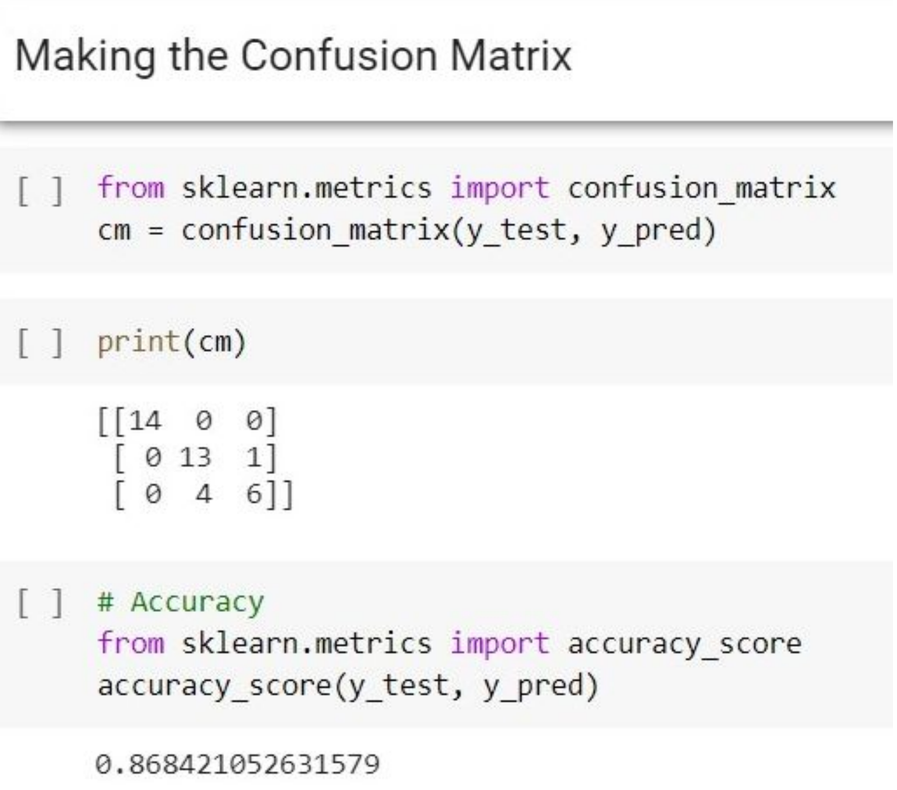


```
In [10]: # Making the Confusion Matrix
         from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test, y_pred)
         print(cm)
         from sklearn.metrics import accuracy_score
         print(accuracy_score(y_test, y_pred))
```

```
[[57 11]
 [15 17]]
0.74
```

# Naive Bayes

## Iris



### Making the Confusion Matrix

```
[ ]  from sklearn.metrics import confusion_matrix
     cm = confusion_matrix(y_test, y_pred)
```

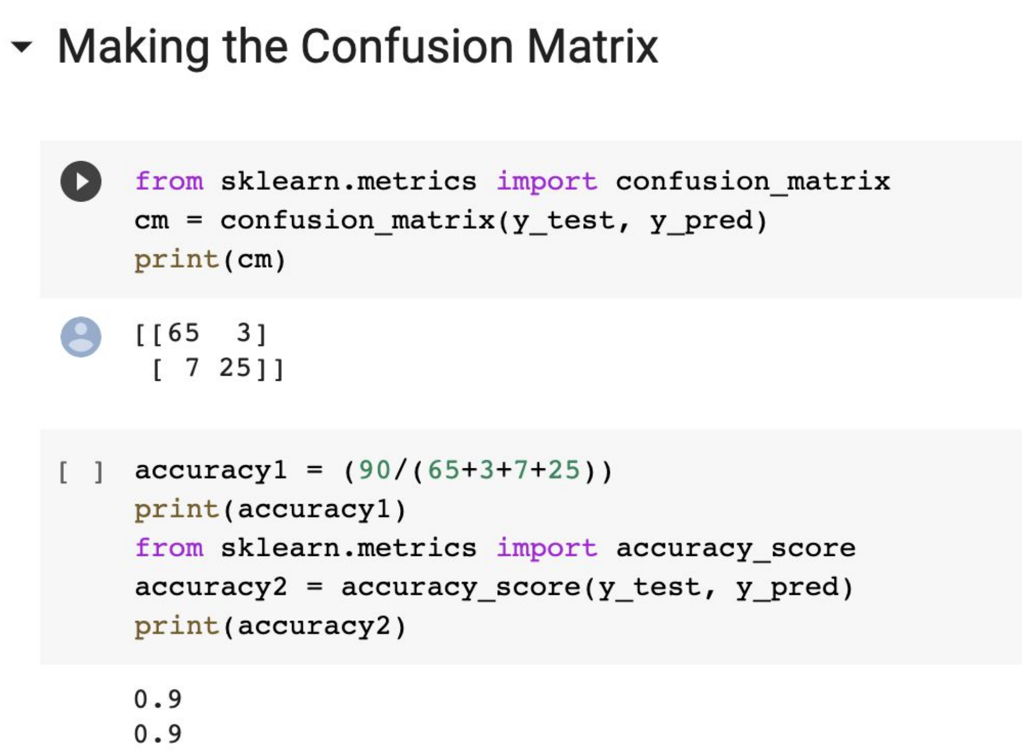
```
[ ]  print(cm)
```

```
[[14  0  0]
 [ 0 13  1]
 [ 0  4  6]]
```

```
[ ]  # Accuracy
     from sklearn.metrics import accuracy_score
     accuracy_score(y_test, y_pred)
```

```
0.868421052631579
```

## Social Network Ads



### Making the Confusion Matrix

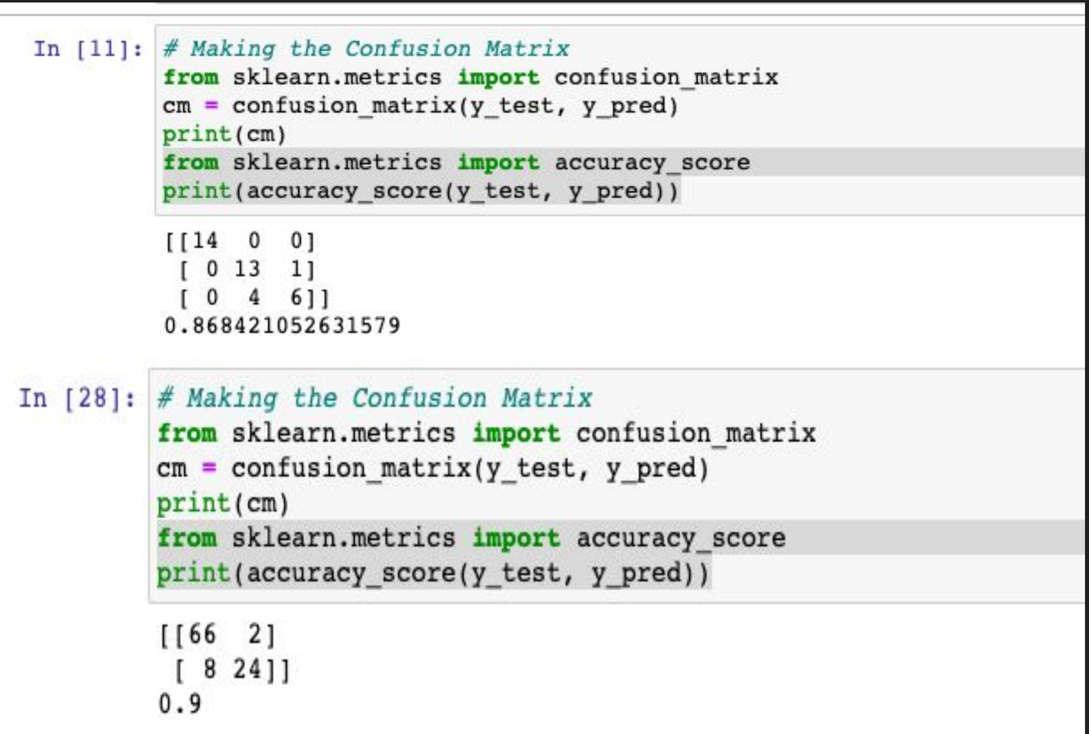
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[65  3]
 [ 7 25]]
```

```
[ ]  accuracy1 = (90/(65+3+7+25))
     print(accuracy1)
     from sklearn.metrics import accuracy_score
     accuracy2 = accuracy_score(y_test, y_pred)
     print(accuracy2)
```

```
0.9
0.9
```

# Support Vector Machine (SVM)



Iris

```
In [11]: # Making the Confusion Matrix
         from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test, y_pred)
         print(cm)
         from sklearn.metrics import accuracy_score
         print(accuracy_score(y_test, y_pred))
```

```
[[14  0  0]
 [ 0 13  1]
 [ 0  4  6]]
0.868421052631579
```

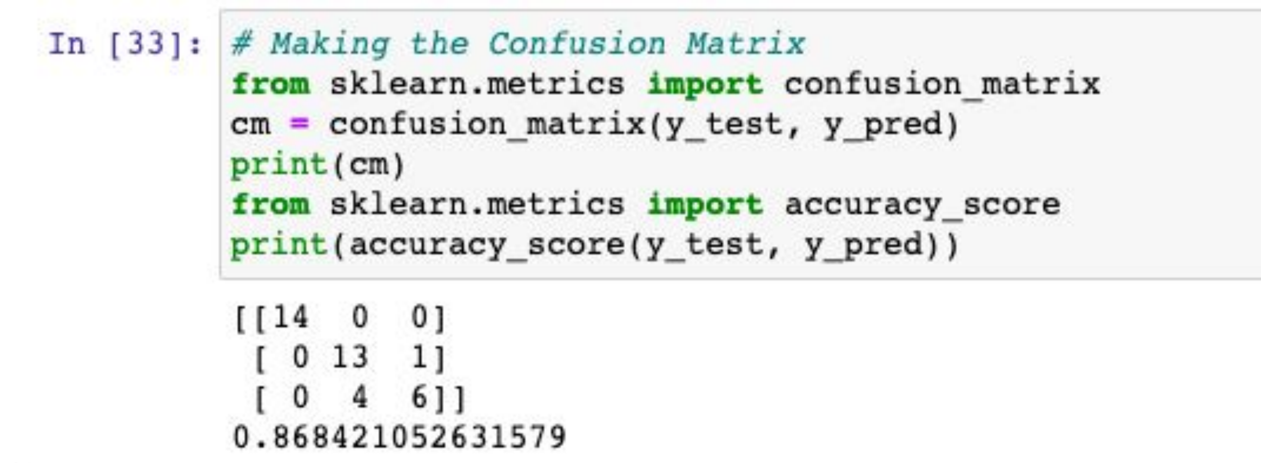
Social Network Ads

```
In [28]: # Making the Confusion Matrix
         from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test, y_pred)
         print(cm)
         from sklearn.metrics import accuracy_score
         print(accuracy_score(y_test, y_pred))
```

```
[[66  2]
 [ 8 24]]
0.9
```

# Logistic Regression

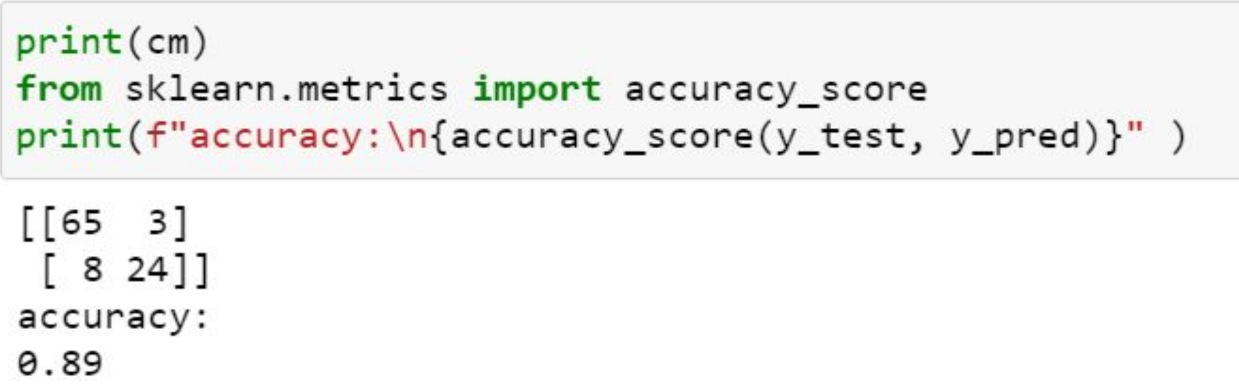
Iris



```
In [33]: # Making the Confusion Matrix
         from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test, y_pred)
         print(cm)
         from sklearn.metrics import accuracy_score
         print(accuracy_score(y_test, y_pred))

         [[14  0  0]
          [ 0 13  1]
          [ 0  4  6]]
         0.868421052631579
```

Social Network Ads



```
print(cm)
from sklearn.metrics import accuracy_score
print(f"accuracy:\n{accuracy_score(y_test, y_pred)}" )

[[65  3]
 [ 8 24]]
accuracy:
0.89
```

# Resultados

Analizando los datos extraídos de la matriz de confusión de los dataset de “social network ads” y “iris” utilizando los diferentes modelos podemos ver el más exacto en el caso de “social network ads” es random forest  con 91% de asertividad. Para “iris” siria decision tree es la mejor con un 97.37% de asertividad. En lo personal el más preciso consideramos que es con la mas alta acertividad en los dos dataset.