Name : Tanishq Thuse

Div : SY CSAI - B

Roll no. : 60

PRN : 12310237

Subject : DV Assignment - 6

Assignment-6

Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

Dataset link : https://www.kaggle.com/datasets/akram24/social-network-ads

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
path = "/content/Social_Network_Ads.csv"
df = pd.read_csv(path)
```

Dataframe creation and EDA

```
df.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0

Next steps: Generate code with df View recommended plots New interactive sheet

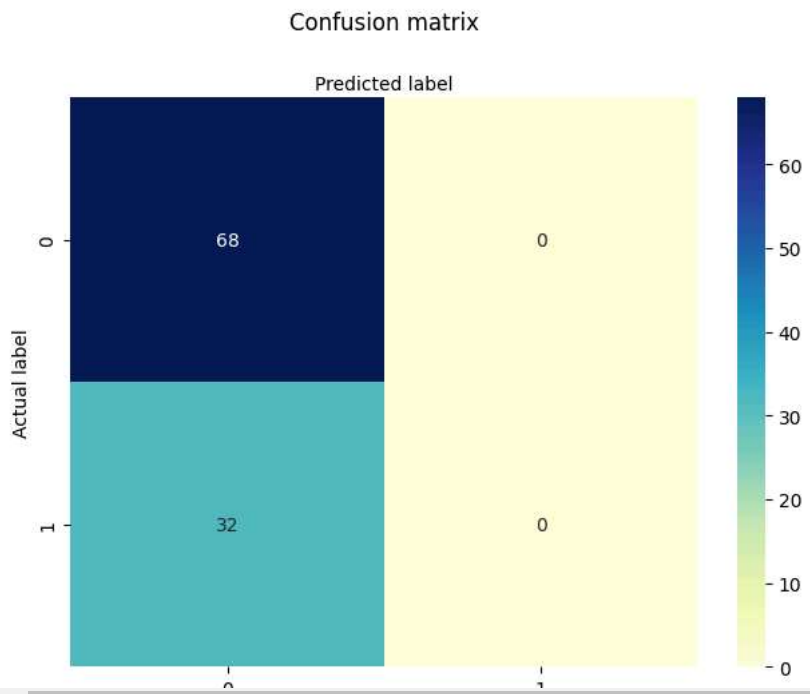
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

# Assuming 'Purchased' is the target variable
X = df[['Age', 'EstimatedSalary']]  #
y = df['Purchased']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)

logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)

cnf_matrix = metrics.confusion_matrix(y_test, y_pred)

# Extract values from confusion matrix
TN = cnf_matrix[0][0]
FP = cnf_matrix[0][1]
FN = cnf_matrix[1][0]
TP = cnf_matrix[1][1]

# Compute metrics
accuracy = (TP + TN) / (TP + FP + TN + FN)
error_rate = 1 - accuracy
precision = TP / (TP + FP)
```

```
recall = TP / (TP + FN)

print("Confusion Matrix:")
print(cnf_matrix)
print("\nMetrics:")
print("Accuracy:", accuracy)
print("Error Rate:", error_rate)
print("Precision:", precision)
print("Recall:", recall)
```

```
Confusion Matrix:
[[68  0]
 [32  0]]

Metrics:
Accuracy: 0.68
Error Rate: 0.31999999999999995
Precision: nan
Recall: 0.0
<ipython-input-10-ccb3e6b34667>:26: RuntimeWarning: invalid value encountered in scalar divide
  precision = TP / (TP + FP)
```

```
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

```
Text(0.5, 427.9555555555555, 'Predicted label')
```



```
print(f"True Positives (TP): {TP}")
print(f"False Positives (FP): {FP}")
print(f"True Negatives (TN): {TN}")
print(f"False Negatives (FN): {FN}")
print(f"Accuracy: {accuracy:.4f}")
print(f"Error Rate: {error_rate:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
```

```
True Positives (TP): 0
False Positives (FP): 0
```

```
True Negatives (TN): 68
False Negatives (FN): 32
Accuracy: 0.6800
Error Rate: 0.3200
Precision: nan
Recall: 0.0000
```

```
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Error Rate: {error_rate}")
print(f"Recall: {recall}")
```

```
Accuracy: 0.68
Precision: nan
Error Rate: 0.31999999999999995
Recall: 0.0
```