K-Nearest Neighbors (KNN)

In [46]:
```
import pandas as pd

data_table= {
    'X1': [7,7,3,1 ],
    'X2': [7,4,4,4],
    'Kelas': ['Sunny', 'Sunny', 'Rain', 'Rain']
  }

df = pd.DataFrame(data_table)

df
```

Out[46]:

	X1	X2	Kelas
0	7	7	Sunny
1	7	4	Sunny
2	3	4	Rain
3	1	4	Rain

In [47]:
```
df.describe()
```

Out[47]:

	X1	X2
count	4.0	4.00
mean	4.5	4.75
std	3.0	1.50
min	1.0	4.00
25%	2.5	4.00
50%	5.0	4.00
75%	7.0	4.75
max	7.0	7.00

In [48]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   X1      4 non-null      int64 
 1   X2      4 non-null      int64 
 2   Kelas   4 non-null      object
dtypes: int64(2), object(1)
memory usage: 224.0+ bytes
```

In [49]:
```
X = df.drop('Kelas',axis=1)
y = df.Kelas
```

```
X.iloc[[0]]
```

Out[49]:

	X1	X2
0	7	7

In [50]:
```
import numpy as np

def dist(X1,X2):
  return np.sqrt(np.sum((X1-X2)**2))
```

In [51]:
```
Xtest = [3,7]
```

In [52]:
```
D = np.array([[]])
for i in range(0, len(X)):
  D = np.append(D, [i, dist(np.array(X.iloc[[i]]),Xtest), y[i]])

D = np.reshape(D, (len(X), len(Xtest)+1))
D
```

Out[52]:
```
array([['0', '4.0', 'Sunny'],
       ['1', '5.0', 'Sunny'],
       ['2', '3.0', 'Rain'],
       ['3', '3.605551275463989', 'Rain']], dtype='<U32')
```

In [53]:
```
D = D[D[:,1].argsort()]
D
```

Out[53]:
```
array([['2', '3.0', 'Rain'],
       ['3', '3.605551275463989', 'Rain'],
       ['0', '4.0', 'Sunny'],
       ['1', '5.0', 'Sunny']], dtype='<U32')
```

In [54]:
```
k = 3
FinalD = D[0:k,:]
print(FinalD)
```

```
[['2' '3.0' 'Rain']
 ['3' '3.605551275463989' 'Rain']
 ['0' '4.0' 'Sunny']]
```

KNN in Machine Learning (Sklearn)

Pada contoh kali ini, kita akan menggunakan data cuaca untuk bermain Tennis sebagai contoh dalam klasifikasi dnegan KNN.

Memanggil semua library yang digunakan

In [55]:
```
import pandas as pd

data = pd.read_csv('tennis.csv')
data
```

Out[55]:

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes
5	rainy	cool	normal	True	no
6	overcast	cool	normal	True	yes
7	sunny	mild	high	False	no
8	sunny	cool	normal	False	yes
9	rainy	mild	normal	False	yes
10	sunny	mild	normal	True	yes
11	overcast	mild	high	True	yes
12	overcast	hot	normal	False	yes
13	rainy	mild	high	True	no

Cek Missing Value

In [56]:
```
data.isna().sum()
```

Out[56]:
```
outlook     0
temp        0
humidity    0
windy       0
```

```
play        0
dtype: int64
```

Splitting Data

Menentukan data X (response) dan y (class)

Dalam Contoh Kali ini fitur X hanya menggunakan kolom 'outlook' dan 'temp'. Sedangkan untuk kelas tetap menggunakan fitur 'play'

In [57]:
```
X = data[['outlook','temp','humidity','windy']]
y = data.play

X.head()
```

Out[57]:

	outlook	temp	humidity	windy
0	sunny	hot	high	False
1	sunny	hot	high	True
2	overcast	hot	high	False
3	rainy	mild	high	False
4	rainy	cool	normal	False

Encoding Data categorial into numeric

In [58]:
```
from sklearn import preprocessing

le = preprocessing.LabelEncoder()
X = X.apply(le.fit_transform)

X.head()
```

Out[58]:

	outlook	temp	humidity	windy
0	2	1	0	0
1	2	1	0	1
2	0	1	0	0
3	1	2	0	0
4	1	0	1	0

In [59]:
```
y = le.fit_transform(y)

print(y)
```

```
[0 0 1 1 1 0 1 0 1 1 1 1 1 0]
```

Teknik splitting data dengan Scikit-Learn

Training 60%, testing 40%

In [60]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y , test_size = 0.4)
```

In [61]:
```
print(X_test)
```

```
    outlook  temp  humidity  windy
8         2     0         1      0
6         0     0         1      1
13        1     2         0      1
3         1     2         0      0
12        0     1         1      0
4         1     0         1      0
```

Training Model

In [62]:
```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=3)

# Train the model using the training sets
classifier = model.fit(X_train,y_train)
```

In [63]:
```
y_pred = classifier.predict(X_test)
```

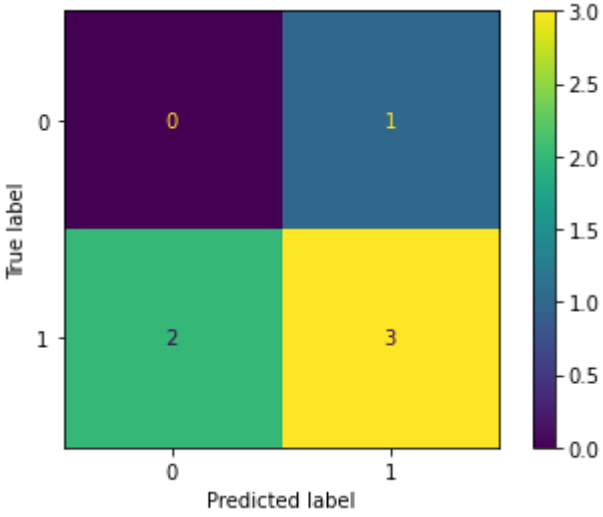
Evaluation

In [64]:
```
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(classifier, X_test, y_test)
```

```
C:\Users\USER\anaconda3\envs\tensorflow\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

Out[64]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f7c2affac0>
```



In [65]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
```

```
[[0 1]
 [2 3]]
```

In [66]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.75      0.60      0.67         5

    accuracy                           0.50         6
   macro avg       0.38      0.30      0.33         6
weighted avg       0.62      0.50      0.56         6
```

EXERCISE/ HOMEWORK

1. Buatlah program klasifikasi Credit Policy pada data Loan menggunakan model KNN. Data dapat diunduh pada tautan berikut ini https://www.kaggle.com/itssuru/loan-data

Fitur predictors:

1. purpose
2. int.rate
3. Installment

fitur class:

1. credit.policy

1. Buatlah model klasifikasi social network menggunakan data berikut ini: https://www.kaggle.com/datasets/rakeshrau/social-network-ads

Fitur prediktor:

1. Gender (Perlu dubah dari text ke angka, gunakan Label Encoder)
2. Age
3. Estimaedsalary

Fitur class:

1. Purchased

Soal no 1.

Memanggil semua library yang digunakan

In [67]:
```
import pandas as pd

data = pd.read_csv('loan_data.csv')
data
```

Out[67]:

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr
0	1	debt_consolidation	0.1189	829.10	11.350407	19.48	737	5639.958
1	1	credit_card	0.1071	228.22	11.082143	14.29	707	2760.000
2	1	debt_consolidation	0.1357	366.86	10.373491	11.63	682	4710.000
3	1	debt_consolidation	0.1008	162.34	11.350407	8.10	712	2699.958
4	1	credit_card	0.1426	102.92	11.299732	14.97	667	4066.000
...	...	...	...	...	...	...	...	
9573	0	all_other	0.1461	344.76	12.180755	10.39	672	10474.000
9574	0	all_other	0.1253	257.70	11.141862	0.21	722	4380.000
9575	0	debt_consolidation	0.1071	97.81	10.596635	13.09	687	3450.041
9576	0	home_improvement	0.1600	351.58	10.819778	19.18	692	1800.000
9577	0	debt_consolidation	0.1392	853.43	11.264464	16.28	732	4740.000

9578 rows × 14 columns

Cek Missing Value

In [68]:
```
data.isna().sum()
```

Out[68]:
```
credit.policy        0
purpose              0
int.rate             0
installment          0
log.annual.inc       0
dti                  0
fico                 0
days.with.cr.line    0
revol.bal            0
revol.util           0
inq.last.6mths       0
delinq.2yrs          0
pub.rec              0
not.fully.paid       0
dtype: int64
```

Splitting Data

Menentukan data X (response) dan y (class)

In [69]:
```
X = data[['purpose','int.rate','installment']]
y = data['credit.policy']

X.head()
```

Out[69]:

	purpose	int.rate	installment
0	debt_consolidation	0.1189	829.10
1	credit_card	0.1071	228.22
2	debt_consolidation	0.1357	366.86

	purpose	int.rate	installment
3	debt_consolidation	0.1008	162.34
4	credit_card	0.1426	102.92

Encoding Data categorial into numeric

In [70]:
```
from sklearn import preprocessing

le = preprocessing.LabelEncoder()
X[['purpose']] = X[['purpose']].apply(le.fit_transform)

X.head()
```

```
C:\Users\USER\AppData\Local\Temp/ipykernel_11252/3834355143.py:4: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  X[['purpose']] = X[['purpose']].apply(le.fit_transform)
```

Out[70]:

	purpose	int.rate	installment
0	2	0.1189	829.10
1	1	0.1071	228.22
2	2	0.1357	366.86
3	2	0.1008	162.34
4	1	0.1426	102.92

In [71]:
```
  y = le.fit_transform(y)

print(y)
```

```
[1 1 1 ... 0 0 0]
```

Teknik splitting data dengan Scikit-Learn

Training 60%, testing 40%

In [72]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y , test_size = 0.4)
```

In [73]:
```
print(X_test)
```

```
      purpose  int.rate  installment
8178        2    0.1425       343.00
2383        2    0.1411        41.08
5287        2    0.1148       441.78
1047        1    0.0976       162.39
3945        2    0.1316       337.71
...       ...       ...          ...
5075        2    0.0894       476.58
7126        0    0.0988       483.16
```

```
9026        5    0.1600       175.79
3593        0    0.0932       127.79
4145        0    0.1322       219.71

[3832 rows x 3 columns]
```

Training Model

In [74]:
```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=5)

# Train the model using the training sets
classifier = model.fit(X_train,y_train)
```

In [75]:
```
y_pred = classifier.predict(X_test)
```

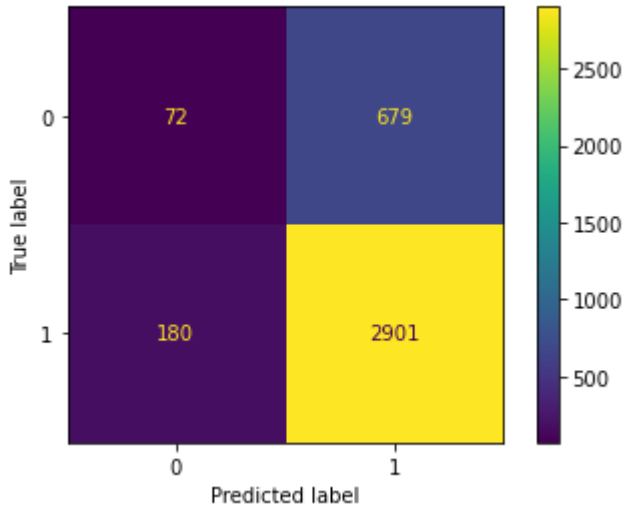
Evaluation

In [76]:
```
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(classifier, X_test, y_test)
```

```
C:\Users\USER\anaconda3\envs\tensorflow\lib\site-packages\sklearn\utils\deprecation.
py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_c
onfusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the cla
ss methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_e
stimator.
  warnings.warn(msg, category=FutureWarning)
```

Out[76]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f7c2ad7040>
```



In [77]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
```

```
[[  72  679]
 [ 180 2901]]
```

In [78]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support
```

```
           0       0.29      0.10      0.14       751
           1       0.81      0.94      0.87      3081

    accuracy                           0.78      3832
   macro avg       0.55      0.52      0.51      3832
weighted avg       0.71      0.78      0.73      3832
```

Soal no. 2

Memanggil semua library yang digunakan

In [79]:
```
import pandas as pd

data = pd.read_csv('Social_Network_Ads.csv')
data
```

Out[79]:

	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
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

Cek Missing Value

In [80]:
```
data.isna().sum()
```

Out[80]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

Splitting Data

Menentukan data X (response) dan y (class)

In [81]:
```
X = data[['Gender','Age','EstimatedSalary']]
y = data['Purchased']
```

```
X.head()
```

Out[81]:

	Gender	Age	EstimatedSalary
0	Male	19	19000
1	Male	35	20000
2	Female	26	43000
3	Female	27	57000
4	Male	19	76000

Encoding Data categorial into numeric

In [82]:

```
from sklearn import preprocessing

le = preprocessing.LabelEncoder()
X[['Gender']] = X[['Gender']].apply(le.fit_transform)

X.head()
```

```
C:\Users\USER\AppData\Local\Temp/ipykernel_11252/3206729699.py:4: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  X[['Gender']] = X[['Gender']].apply(le.fit_transform)
```

Out[82]:

	Gender	Age	EstimatedSalary
0	1	19	19000
1	1	35	20000
2	0	26	43000
3	0	27	57000
4	1	19	76000

In [83]:

```
  y = le.fit_transform(y)

print(y)
```

```
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 1
 1 1 0 0 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0 0 1
 1 0 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 1 0 0 0
 1 1 0 1 1 1 1 1 0 0 0 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 1 0
 0 1 0 1 0 0 1 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 1 0 1
 1 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1]
```

Teknik splitting data dengan Scikit-Learn

Training 60%, testing 40%

In [84]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y , test_size = 0.4)
```

In [85]:
```
print(X_test)
```

```
     Gender  Age  EstimatedSalary
251       1   37            52000
53        0   35            23000
0         1   19            19000
179       0   31            34000
344       1   47           105000
..      ...  ...              ...
256       0   41            72000
385       1   56            60000
76        1   18            52000
94        0   29            83000
259       0   45           131000

[160 rows x 3 columns]
```

Training Model

In [86]:
```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=15)

# Train the model using the training sets
classifier = model.fit(X_train,y_train)
```

In [87]:
```
y_pred = classifier.predict(X_test)
```

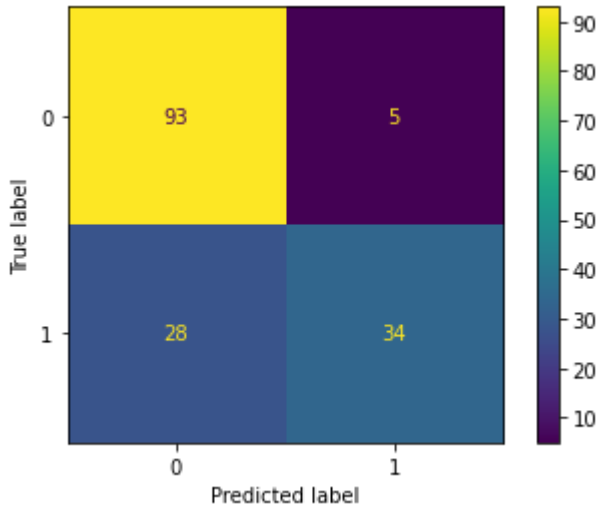
Evaluation

In [88]:
```
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(classifier, X_test, y_test)
```

```
C:\Users\USER\anaconda3\envs\tensorflow\lib\site-packages\sklearn\utils\deprecation.
py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_c
onfusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the cla
ss methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_e
stimator.
  warnings.warn(msg, category=FutureWarning)
```

Out[88]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f7c2cddac0>
```



In [89]:

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

```
[[93  5]
 [28 34]]
```

In [90]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.77      0.95      0.85        98
           1       0.87      0.55      0.67        62

    accuracy                           0.79       160
   macro avg       0.82      0.75      0.76       160
weighted avg       0.81      0.79      0.78       160
```