Implementasi dengan Algoritma KNN (scikit-learn)

In [24]:

```
import numpy as nm
import matplotlib.pyplot as plt
import pandas as pd
dataset1 = pd.read_csv('data_train.csv')
dataset2 = pd.read_csv('data_validation.csv')
```

Setup

In [25]:

```
# dataset splitting
x_train = dataset1.iloc[:, :-1].values
x_test = dataset2.iloc[:, :-1].values
y_train = dataset1.iloc[:, -1].values
y_test = dataset2.iloc[:, -1].values
```

Mencari K terbaik dan menunjukkan hasil akurasi

In [26]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, ConfusionMatrixDisplay

# Mencari nilai k terbaik dengan Scikit
k = []
accuracy = 0
best_y_pred = None

for i in range (1,len(x_test)):
  KNN_Scikit = KNeighborsClassifier(i, weights="distance", p=2)
  KNN_Scikit.fit(x_train, y_train)
  y_pred = KNN_Scikit.predict(x_test)
  accuracy_temp = accuracy_score(y_test, y_pred)
  if (accuracy_temp > accuracy):
    accuracy = accuracy_temp
    k = [i]
    best_y_pred = y_pred
  elif (accuracy_temp == accuracy):
    accuracy = accuracy_temp
    k.append(i)

print(f'k = {k}')
print(f'accuracy = {accuracy}')
print()
print("Confusion Matrix:")
print(confusion_matrix(y_test, best_y_pred))
print()
print("Classification Report:")
print(classification_report(y_test, best_y_pred))
cmd = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y_test, best_y_pred), display_labels=KNN_Scikit.classes_)
fig, ax = plt.subplots(figsize=(5, 5))
cmd.plot(ax=ax)
```

```
k = [56]
accuracy = 0.94

Confusion Matrix:
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
d:\JASON\KULIAH ITB\MATKUL\SEMESTER 5\Inteligensi Buatan\Tubes2AI-EDAImplemetation\src\KNN_Scikit.ipynb Cell 6 line 2
     <a href='vscode-notebook-cell:/d%3A/JASON/KULIAH%20ITB/MATKUL/SEMESTER%205/Inteligensi%20Buatan/Tubes2AI-EDAImplemetation/src/KNN_Scikit.ipynb#X16sZmlsZQ%3D%3D?line=20'>21</a> print()
     <a href='vscode-notebook-cell:/d%3A/JASON/KULIAH%20ITB/MATKUL/SEMESTER%205/Inteligensi%20Buatan/Tubes2AI-EDAImplemetation/src/KNN_Scikit.ipynb#X16sZmlsZQ%3D%3D?line=21'>22</a> print("Confusion Matrix:")
---> <a href='vscode-notebook-cell:/d%3A/JASON/KULIAH%20ITB/MATKUL/SEMESTER%205/Inteligensi%20Buatan/Tubes2AI-EDAImplemetation/src/KNN_Scikit.ipynb#X16sZmlsZQ%3D%3D?line=22'>23</a> print(confusion_matrix(k, y_pred))
     <a href='vscode-notebook-cell:/d%3A/JASON/KULIAH%20ITB/MATKUL/SEMESTER%205/Inteligensi%20Buatan/Tubes2AI-EDAImplemetation/src/KNN_Scikit.ipynb#X16sZmlsZQ%3D%3D?line=23'>24</a> print()
     <a href='vscode-notebook-cell:/d%3A/JASON/KULIAH%20ITB/MATKUL/SEMESTER%205/Inteligensi%20Buatan/Tubes2AI-EDAImplemetation/src/KNN_Scikit.ipynb#X16sZmlsZQ%3D%3D?line=24'>25</a> print("Classification Report:")

File c:\Users\jason\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\_param_validation.py:214, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    208 try:
    209     with config_context(
    210         skip_parameter_validation=(
    211             prefer_skip_nested_validation or global_skip_validation
    212         )
    213     ):
--> 214         return func(*args, **kwargs)
    215 except InvalidParameterError as e:
    216     # When the function is just a wrapper around an estimator, we allow
    217     # the function to delegate validation to the estimator, but we replace
    218     # the name of the estimator by the name of the function in the error
    219     # message to avoid confusion.
    220     msg = re.sub(
    221         r"parameter of \w+ must be",
    222         f"parameter of {func.__qualname__} must be",
    223         str(e),
    224     )

File c:\Users\jason\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metrics\_classification.py:326, in confusion_matrix(y_true, y_pred, labels, sample_weight, normalize)
    231 @validate_params(
    232     {
    233         "y_true": ["array-like"],
   (...)
    242     y_true, y_pred, *, labels=None, sample_weight=None, normalize=None
    243 ):
    244     """Compute confusion matrix to evaluate the accuracy of a classification.
    245 
    246     By definition a confusion matrix :math:`C` is such that :math:`C_{i, j}`
   (...)
    324     (0, 2, 1, 1)
    325     """
--> 326     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    327     if y_type not in ("binary", "multiclass"):
    328         raise ValueError("%s is not supported" % y_type)

File c:\Users\jason\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metrics\_classification.py:84, in _check_targets(y_true, y_pred)
     57 def _check_targets(y_true, y_pred):
     58     """Check that y_true and y_pred belong to the same classification task.
     59 
     60     This converts multiclass or binary types to a common shape, and raises a
   (...)
     82     y_pred : array or indicator matrix
     83     """
---> 84     check_consistent_length(y_true, y_pred)
     85     type_true = type_of_target(y_true, input_name="y_true")
     86     type_pred = type_of_target(y_pred, input_name="y_pred")

File c:\Users\jason\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\validation.py:407, in check_consistent_length(*arrays)
    405 uniques = np.unique(lengths)
    406 if len(uniques) > 1:
--> 407     raise ValueError(
    408         "Found input variables with inconsistent numbers of samples: %r"
    409         % [int(l) for l in lengths]
    410     )

ValueError: Found input variables with inconsistent numbers of samples: [1, 600]
```

In [ ]:

```
y_pred = KNN_Scikit.predict(x_test)
y_pred
```

Out[ ]:

```
array([2, 2, 3, 0, 3, 1, 3, 0, 3, 1, 3, 2, 3, 0, 3, 0, 2, 1, 0, 2, 3, 1,
       0, 1, 1, 0, 2, 1, 0, 2, 1, 3, 3, 0, 2, 3, 1, 3, 1, 1, 0, 2, 0, 2,
       2, 1, 1, 2, 2, 3, 1, 2, 3, 0, 1, 3, 2, 3, 3, 2, 2, 3, 3, 1, 3, 2,
       3, 2, 2, 3, 2, 3, 1, 0, 1, 2, 0, 3, 1, 0, 3, 3, 0, 2, 3, 1, 3, 3,
       0, 2, 1, 1, 1, 2, 1, 0, 3, 2, 1, 3, 3, 3, 1, 2, 3, 2, 3, 3, 3, 3,
       3, 3, 2, 2, 3, 1, 0, 3, 1, 3, 1, 2, 2, 3, 2, 0, 2, 2, 1, 3, 3, 1,
       0, 0, 3, 0, 0, 1, 3, 0, 1, 3, 3, 1, 2, 3, 1, 2, 1, 2, 3, 0, 0, 2,
       1, 0, 2, 0, 0, 3, 3, 3, 0, 2, 3, 0, 0, 1, 0, 2, 3, 1, 0, 1, 0, 1,
       3, 0, 2, 0, 3, 1, 1, 2, 0, 2, 0, 3, 2, 0, 3, 2, 0, 0, 2, 0, 1, 3,
       3, 1, 1, 2, 2, 3, 2, 3, 3, 3, 0, 2, 0, 2, 3, 1, 2, 2, 2, 0, 0, 2,
       2, 1, 3, 3, 0, 2, 0, 3, 0, 2, 2, 2, 2, 1, 3, 0, 1, 3, 3, 3, 0, 0,
       1, 1, 3, 3, 1, 2, 0, 2, 1, 2, 3, 3, 0, 2, 2, 1, 1, 1, 0, 2, 2, 3,
       3, 0, 1, 3, 3, 0, 0, 2, 1, 1, 1, 0, 3, 2, 2, 2, 0, 1, 0, 2, 1, 1,
       0, 0, 3, 0, 3, 0, 1, 0, 1, 3, 3, 2, 0, 2, 1, 0, 0, 1, 2, 2, 0, 3,
       1, 1, 0, 2, 3, 1, 3, 2, 3, 1, 2, 2, 0, 0, 2, 1, 2, 1, 0, 1, 1, 0,
       3, 2, 3, 2, 2, 2, 3, 0, 1, 0, 1, 1, 0, 2, 2, 2, 2, 2, 1, 2, 3, 0,
       2, 1, 2, 3, 2, 3, 2, 0, 3, 0, 0, 3, 2, 0, 3, 2, 1, 2, 0, 0, 1, 2,
       1, 3, 0, 0, 2, 2, 0, 2, 1, 0, 3, 2, 3, 3, 0, 0, 0, 3, 1, 0, 3, 0,
       0, 2, 0, 3, 1, 0, 1, 1, 1, 2, 0, 1, 1, 2, 3, 1, 2, 2, 3, 1, 1, 2,
       2, 2, 0, 1, 3, 0, 0, 1, 0, 0, 0, 3, 0, 1, 2, 0, 0, 2, 1, 0, 3, 2,
       1, 0, 2, 0, 2, 1, 1, 1, 2, 0, 1, 2, 0, 2, 0, 3, 0, 0, 1, 2, 1, 1,
       3, 1, 1, 2, 3, 0, 3, 2, 0, 1, 2, 2, 0, 1, 2, 0, 2, 0, 1, 3, 0, 0,
       3, 0, 2, 2, 3, 3, 1, 3, 3, 0, 1, 2, 1, 3, 1, 0, 3, 1, 3, 3, 1, 1,
       3, 0, 3, 3, 1, 2, 0, 3, 0, 2, 2, 0, 2, 3, 0, 3, 0, 2, 3, 3, 0, 2,
       1, 2, 3, 3, 3, 1, 2, 0, 2, 3, 3, 1, 2, 3, 1, 1, 0, 0, 3, 0, 2, 1,
       1, 3, 2, 1, 1, 3, 0, 3, 0, 0, 3, 3, 2, 1, 2, 2, 0, 2, 0, 1, 0, 2,
       0, 1, 0, 0, 2, 3, 0, 1, 1, 1, 0, 1, 2, 0, 1, 3, 0, 2, 1, 1, 0, 1,
       3, 1, 3, 0, 1, 3], dtype=int64)
```