```
# Lab 9: K-Nearest Neighbor Algorithm
# 0. Installation and Imports

# Uncomment if not installed
!pip install ucimlrepo

from ucimlrepo import fetch_ucirepo
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
```

```
Requirement already satisfied: ucimlrepo in /usr/local/lib/python3.12/dist-packages (0.0.7)
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from ucimlrepo) (2.2.2)
Requirement already satisfied: certifi>=2020.12.5 in /usr/local/lib/python3.12/dist-packages (from ucimlrepo) (2025.1
Requirement already satisfied: numpy>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0->ucimlrep
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0-
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0->ucimlrepo
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0->ucimlre
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->panda
```

```
# Lab 9: K-Nearest Neighbor Algorithm
# 1. Load Dataset and Inspect

japanese_credit_screening = fetch_ucirepo(id=28)

X = japanese_credit_screening.data.features
y = japanese_credit_screening.data.targets

print("Metadata:\n", japanese_credit_screening.metadata)
print("\nVariables:\n", japanese_credit_screening.variables)
print("Sample features:\n", X.head())
print("Sample targets:\n", y.head())
```

```
Metadata:
 {'uci_id': 28, 'name': 'Japanese Credit Screening', 'repository_url': 'https://archive.ics.uci.edu/dataset/28/japane

Variables:
    name     role         type demographic description units missing_values
0    A1  Feature  Categorical        None        None  None            yes
1    A2  Feature   Continuous        None        None  None            yes
2    A3  Feature   Continuous        None        None  None             no
3    A4  Feature  Categorical        None        None  None            yes
4    A5  Feature  Categorical        None        None  None            yes
5    A6  Feature  Categorical        None        None  None            yes
6    A7  Feature  Categorical        None        None  None            yes
7    A8  Feature   Continuous        None        None  None             no
8    A9  Feature       Binary        None        None  None             no
9   A10  Feature       Binary        None        None  None             no
10  A11  Feature      Integer        None        None  None             no
11  A12  Feature       Binary        None        None  None             no
12  A13  Feature  Categorical        None        None  None            yes
13  A14  Feature      Integer        None        None  None             no
14  A15  Feature      Integer        None        None  None             no
15  A16   Target       Binary        None        None  None             no
Sample features:
   A1     A2     A3 A4 A5 A6 A7    A8 A9 A10  A11 A12 A13    A14  A15
0  b  30.83  0.000  u  g  w  v  1.25  t   t    1   f   g  00202    0
1  a  58.67  4.460  u  g  q  h  3.04  t   t    6   f   g  00043  560
2  a  24.50  0.500  u  g  q  h  1.50  t   f    0   f   g  00280  824
3  b  27.83  1.540  u  g  w  v  3.75  t   t    5   t   g  00100    3
4  b  20.17  5.625  u  g  w  v  1.71  t   f    0   f   s  00120    0
Sample targets:
   A16
0   +
1   +
2   +
3   +
4   +
```

Toggle Gemini

Gemini

```
# Lab 9: K-Nearest Neighbor Algorithm
# 2. Train-Test Split

X_train, X_test, y_train, y_test = train_test_split(
    X_encoded, y.values.ravel(), test_size=0.2, random_state=42
)

print("Training set size:", X_train.shape)
print("Testing set size:", X_test.shape)
```

```
Training set size: (552, 570)
Testing set size: (138, 570)
```

```
# Lab 9: K-Nearest Neighbor Algorithm
# 3. KNN Training and Prediction

# You can set k as needed, e.g., n_neighbors=5
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

```
# Lab 9: K-Nearest Neighbor Algorithm
# 2.1 Preprocessing: One-Hot Encoding

X_encoded = pd.get_dummies(X, columns=X.select_dtypes(include='object').columns, dummy_na=False)

print("Sample features after one-hot encoding:\n", X_encoded.head())
```

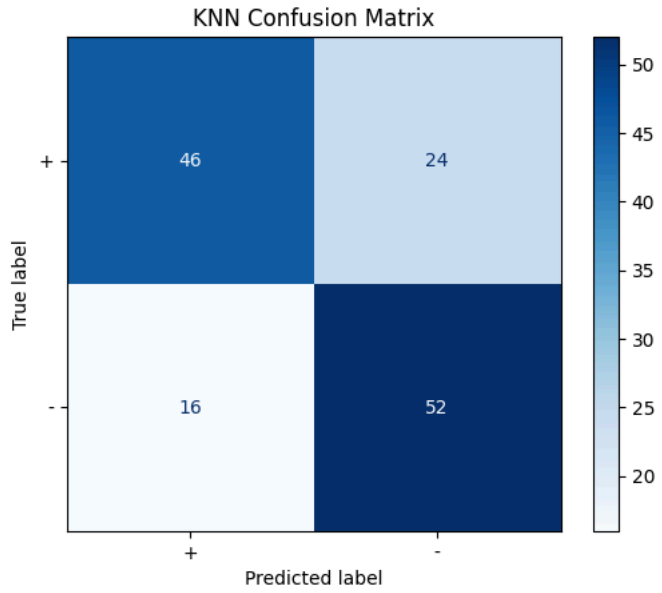
```
Sample features after one-hot encoding:
       A3    A8  A11  A15   A1_?   A1_a   A1_b  A2_13.75  A2_15.17  A2_15.75  \
0  0.000  1.25    1    0  False  False   True     False     False     False
1  4.460  3.04    6  560  False   True  False     False     False     False
2  0.500  1.50    0  824  False   True  False     False     False     False
3  1.540  3.75    5    3  False  False   True     False     False     False
4  5.625  1.71    0    0  False  False   True     False     False     False

   ...  A14_00680  A14_00711  A14_00720  A14_00760  A14_00840  A14_00928  \
0  ...      False      False      False      False      False      False
1  ...      False      False      False      False      False      False
2  ...      False      False      False      False      False      False
3  ...      False      False      False      False      False      False
4  ...      False      False      False      False      False      False

   A14_00980  A14_01160  A14_02000  A14_?
0      False      False      False  False
1      False      False      False  False
2      False      False      False  False
3      False      False      False  False
4      False      False      False  False

[5 rows x 570 columns]
```

```
# Lab 9: K-Nearest Neighbor Algorithm
# 4. Evaluation

print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

Toggle Gemini

```
Accuracy Score: 0.7101449275362319
Classification Report:
              precision    recall  f1-score   support

           +       0.74      0.66      0.70        70
           -       0.68      0.76      0.72        68

    accuracy                           0.71       138
   macro avg       0.71      0.71      0.71       138
weighted avg       0.71      0.71      0.71       138

Confusion Matrix:
 [[46 24]
 [16 52]]
```

```
# Lab 9: K-Nearest Neighbor Algorithm
# 5. Visualize Confusion Matrix

from sklearn.metrics import ConfusionMatrixDisplay

cm = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=knn.classes_).plot(cmap="Blues")
plt.title("KNN Confusion Matrix")
plt.show()
```




Start coding or generate with AI.

Toggle Gemini