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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
df = pd.read_csv("heart_disease_uci.csv")
df.head()
```

| | id | age | sex | dataset | cp | trestbps | chol | fbs | restecg | thalch | exang | oldpeak | slope | ca | th |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 63 | Male | Cleveland | typical angina | 145.0 | 233.0 | True | lv hypertrophy | 150.0 | False | 2.3 | downsloping | 0.0 | fix defe |
| **1** | 2 | 67 | Male | Cleveland | asymptomatic | 160.0 | 286.0 | False | lv hypertrophy | 108.0 | True | 1.5 | flat | 3.0 | norm |
| **2** | 3 | 67 | Male | Cleveland | asymptomatic | 120.0 | 229.0 | False | lv hypertrophy | 129.0 | True | 2.6 | flat | 2.0 | reversab defe |
| **3** | 4 | 37 | Male | Cleveland | non-anginal | 130.0 | 250.0 | False | normal | 187.0 | False | 3.5 | downsloping | 0.0 | norm |
| **4** | 5 | 41 | Female | Cleveland | atypical angina | 130.0 | 204.0 | False | lv hypertrophy | 172.0 | False | 1.4 | upsloping | 0.0 | norm |

Next steps: Generate code with df New interactive sheet

```
df.info()
df.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   id        920 non-null    int64  
 1   age       920 non-null    int64  
 2   sex       920 non-null    object 
 3   dataset   920 non-null    object 
 4   cp        920 non-null    object 
 5   trestbps  861 non-null    float64
 6   chol      890 non-null    float64
 7   fbs       830 non-null    object 
 8   restecg   918 non-null    object 
 9   thalch    865 non-null    float64
 10  exang     865 non-null    object 
 11  oldpeak   858 non-null    float64
 12  slope     611 non-null    object 
 13  ca        309 non-null    float64
 14  thal      434 non-null    object 
 15  num       920 non-null    int64  
dtypes: float64(5), int64(3), object(8)
memory usage: 115.1+ KB
```

```
              0
      id      0
     age      0
     sex      0
 dataset      0
      cp      0
trestbps     59
    chol     30
     fbs     90
 restecg      2
  thalch     55
   exang     55
 oldpeak     62
   slope    309
      ca    611
    thal    486
     num      0

dtype: int64
```

```
df = df.fillna(df.mean(numeric_only=True))
```

```
df['num'].value_counts()
```

```
     count
num
  0    411
  1    265
  2    109
  3    107
  4     28

dtype: int64
```

```
X = df.drop('num', axis=1)
y = df['num']

# Identify categorical columns
categorical_cols = X.select_dtypes(include=['object', 'bool']).columns

# Apply one-hot encoding
X = pd.get_dummies(X, columns=categorical_cols, drop_first=True)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Initialize and train the KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5) # Using 5 neighbors as a common starting point
knn.fit(X_train, y_train)

# Make predictions on the test set
y_pred = knn.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")
```

```
Accuracy: 0.5326
Precision: 0.4861
Recall: 0.5326
F1-Score: 0.5053
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-de
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
# Define a range of k values to test
k_values = range(1, 21) # Test k from 1 to 20

# Initialize lists to store metrics
accuracy_list = []
precision_list = []
recall_list = []
f1_list = []

for k in k_values:
    # Initialize and train the KNeighborsClassifier for the current k
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)

    # Make predictions on the test set
    y_pred = knn.predict(X_test)

    # Evaluate the model and append to lists
    accuracy_list.append(accuracy_score(y_test, y_pred))
    precision_list.append(precision_score(y_test, y_pred, average='weighted', zero_division=0))
    recall_list.append(recall_score(y_test, y_pred, average='weighted', zero_division=0))
    f1_list.append(f1_score(y_test, y_pred, average='weighted', zero_division=0))

# Now, print the results for each k
for i in range(len(k_values)):
    print(f"k = {k_values[i]}")
    print(f"Accuracy: {accuracy_list[i]:.4f}")
    print(f"Precision: {precision_list[i]:.4f}")
    print(f"Recall: {recall_list[i]:.4f}")
    print(f"F1 Score: {f1_list[i]:.4f}")
    print("---------------------------")
```

```
Recall: 0.5489
```

```
Recall: 0.5489
F1 Score: 0.5145
---------------------------
k = 15
Accuracy: 0.5598
Precision: 0.5010
Recall: 0.5598
F1 Score: 0.5226
---------------------------
k = 16
Accuracy: 0.5598
Precision: 0.4955
Recall: 0.5598
F1 Score: 0.5188
---------------------------
k = 17
Accuracy: 0.5598
Precision: 0.4947
Recall: 0.5598
F1 Score: 0.5191
---------------------------
k = 18
Accuracy: 0.5598
Precision: 0.4993
Recall: 0.5598
F1 Score: 0.5160
---------------------------
k = 19
Accuracy: 0.5489
Precision: 0.4560
Recall: 0.5489
F1 Score: 0.4972
---------------------------
k = 20
Accuracy: 0.5380
Precision: 0.4588
Recall: 0.5380
F1 Score: 0.4869
---------------------------
```

```
plt.plot(k_values, f1_list, marker='o')
plt.xlabel("k value")
plt.ylabel("F1 Score")
plt.title("k vs F1 Score in KNN")
plt.show()
```