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

1. Load the data as a Pandas data frame.

```
# Load of the data
url = "https://raw.githubusercontent.com/empathy87/The-Elements-of-Statistical-Learning-Pytho
raw_data = pd.read_csv(url, sep=',')
raw_data.head()
```

| | x1 | x2 | y |
|---|---|---|---|
| **0** | 2.526093 | 0.321050 | 0 |
| **1** | 0.366954 | 0.031462 | 0 |
| **2** | 0.768219 | 0.717486 | 0 |
| **3** | 0.693436 | 0.777194 | 0 |
| **4** | -0.019837 | 0.867254 | 0 |

2. Split the data into 80% training data and 20% test data.

```
# Let split the data
from sklearn.model_selection import train_test_split
training_data, test_data = train_test_split(raw_data, test_size=0.2)
```

```
# Let display the shape of training data
training_data.shape
```

```
(160, 3)
```

```
# Let display the shape of test data
test_data.shape
```

```
(40, 3)
```

3. Build three k-nearest-neighbor model with k = 1, 5, 25, respectively.

```
# kNN model
# Let built k-nearest-neighbor model with k = 1
```

```
from sklearn.neighbors import KNeighborsClassifier
model_1nn = KNeighborsClassifier(n_neighbors=1)
model_1nn.fit(raw_data[['x1', 'x2']], raw_data['y'])
```

```
     KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                          metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                          weights='uniform')
```

```
# Let built k-nearest-neighbor model with k = 5

from sklearn.neighbors import KNeighborsClassifier
model_5nn = KNeighborsClassifier(n_neighbors=5)
model_5nn.fit(raw_data[['x1', 'x2']], raw_data['y'])
```

```
     KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                          metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                          weights='uniform')
```

```
# Let built k-nearest-neighbor model with k = 25

from sklearn.neighbors import KNeighborsClassifier
model_25nn = KNeighborsClassifier(n_neighbors=25)
model_25nn.fit(raw_data[['x1', 'x2']], raw_data['y'])
```

```
     KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                          metric_params=None, n_jobs=None, n_neighbors=25, p=2,
                          weights='uniform')
```

4. Train the models on the training set, and obtain the model predictions on the test set.

```
# train_test_split

train_test_split, test_data = train_test_split(raw_data, test_size = 0.2)
```

```
# Train the k-nearest-neighbor model on the training set with k = 1

model_1nn_train = KNeighborsClassifier(n_neighbors=1)
input_cols = ['x1', 'x2']
model_1nn_train.fit(training_data[input_cols], training_data['y'])
```

```
     KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                          metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                          weights='uniform')
```

```
# Let find the model's predictions on the test set for k = 1

test_data['prediction'] = model_1nn_train.predict(test_data[input_cols])
```

```
test_data.head()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
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/user_
  This is separate from the ipykernel package so we can avoid doing imports until
```

| | x1 | x2 | y | prediction |
|---|---|---|---|---|
| **108** | 1.301202 | 0.725800 | 1 | 1 |
| **37** | 0.818430 | 0.379000 | 0 | 0 |
| **114** | -2.073319 | 1.735424 | 1 | 1 |
| **187** | 0.259434 | 1.250358 | 1 | 1 |
| **21** | -0.429650 | -0.309811 | 0 | 0 |

```
# Accuracy.score() with k = 1

from sklearn.metrics import accuracy_score
accuracy_score(test_data['y'], test_data['prediction'])
```

```
0.975
```

```
# Train the k-nearest-neighbor model on the training set with k = 5

model_5nn_train = KNeighborsClassifier(n_neighbors=5)
input_cols = ['x1', 'x2']
model_5nn_train.fit(training_data[input_cols], training_data['y'])
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

```
# Find the model's predictions on the test set for k = 5

test_data['prediction'] = model_5nn_train.predict(test_data[input_cols])
test_data.head()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
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/user_
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
# Accuracy.score() with k = 5

from sklearn.metrics import accuracy_score
accuracy_score(test_data['y'], test_data['prediction'])
```

```
0.925
```

187 0.259434 1.250358 1 1

```
# Train the k-nearest-neighbor model on the training set with k = 25

model_25nn_train = KNeighborsClassifier(n_neighbors=25)
input_cols = ['x1', 'x2']
model_25nn_train.fit(training_data[input_cols], training_data['y'])
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=25, p=2,
                     weights='uniform')
```

```
# Find the model's predictions on the test set for k = 25

test_data['prediction'] = model_25nn_train.predict(test_data[input_cols])
test_data.head()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
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/user_
  This is separate from the ipykernel package so we can avoid doing imports until
```

| | x1 | x2 | y | prediction |
|---|---|---|---|---|
| **108** | 1.301202 | 0.725800 | 1 | 1 |
| **37** | 0.818430 | 0.379000 | 0 | 0 |
| **114** | -2.073319 | 1.735424 | 1 | 1 |
| **187** | 0.259434 | 1.250358 | 1 | 1 |
| **21** | -0.429650 | -0.309811 | 0 | 0 |

5. Calculate the test accuracy score for each model. Which k value give the best accuracy score?

```
# Accuracy.score() with k = 25
```

```
from sklearn.metrics import accuracy_score
accuracy_score(test_data['y'], test_data['prediction'])
```

```
0.875
```

The accuracy score with k = 1 is : 0.975

The accuracy score with k = 5 is : 0.925

The accuracy score with k = 25 is : 0.875

So k = 1 gives the best accuracy score.