Classification

Autumn 2020

```
##################################################
##                                              ##
##  Machine Learning Fundamentals with Python  ##
##                                              ##
##################################################
```

§1 Supervised Learning with scikit-learn

§1.1 Classification

1 Supervised learning

1. What is machine learning?

Machine learning gives computers the ability to learn to make decisions from data without being explicitly programmed!

2. What is the difference between supervised learning and unsupervised learning?

- Supervised learning → *labeled data*
- Unsupervised learning → *unlabeled data*

3. What is the aim of supervised learning?

- Predict the target variable, given the features.
  - features = *predictor variables* = *independent variables*
  - target variable = *dependent variable* = *response variable*

4. What is the difference between classification and regression?

- Classification → *target variable consists of categories*
- Regression → *target variable is continuous*

5. What can supervised learning do?

- Automate time-consuming or expensive manual tasks.
- Make predictions about the feature.
- Need labeled data, such as the historical data with labels, the crowd-sourcing labeled data, or the labeled data obtained through some experiments.

6. **Practice question for a classification problem:**

- Which of the 4 example applications below of machine learning is a supervised classification problem?

  ☒ Use labeled financial data to predict whether the value of a stock will go up or go down next week.

  ☐ Use labeled housing price data to predict the price of a new house based on various features.

  ☐ Use unlabeled data to cluster the students of an online education company into different categories based on their learning styles.

  ☐ Use labeled financial data to predict what the value of a stock will be next week.

2 Exploratory data analysis

1. **What are the features and the target variable of the Iris dataset?**

- Features: petal length, petal width, sepal length, sepal width
- Target variable: species (*versicolor*, *virginica*, *setosa*)

2. **Code of the Iris dataset in scikit-learn:**

[1]:
```
from sklearn import datasets

iris = datasets.load_iris()
type(iris)
```

[1]: sklearn.utils.Bunch

[2]:
```
print(iris.keys())
```

```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names',
'filename'])
```

[3]:
```
type(iris.data), type(iris.target)
```

[3]: (numpy.ndarray, numpy.ndarray)

[4]:
```
iris.data.shape
```

[4]: (150, 4)

[5]:
```
iris.target_names
```

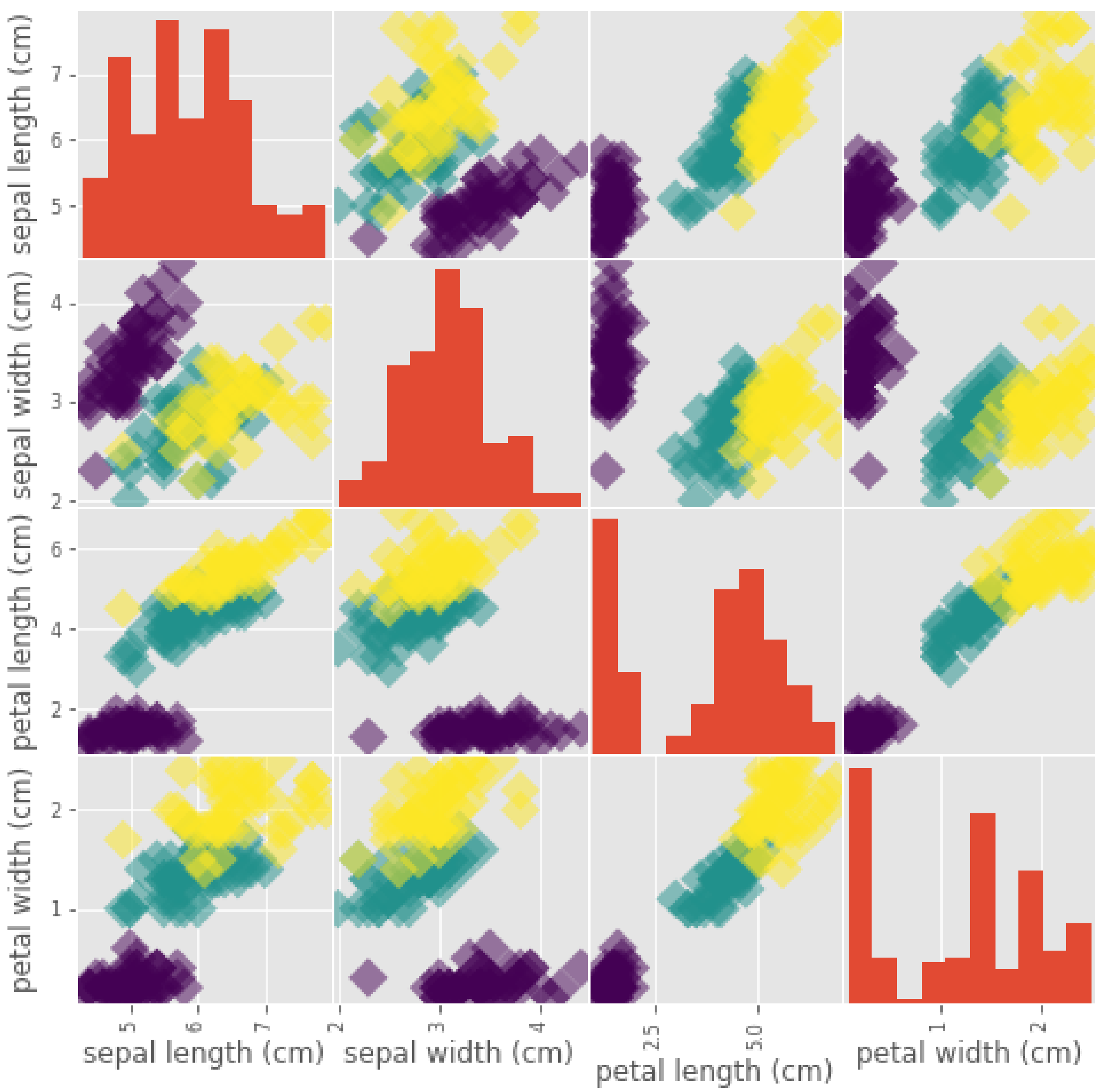
[5]: array(['setosa', 'versicolor', 'virginica'], dtype='<U10')

[6]:
```
import pandas as pd

X = iris.data
```

```
y = iris.target
df = pd.DataFrame(X, columns=iris.feature_names)
print(df.head())
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
```

[7]:
```
import matplotlib.pyplot as plt

plt.style.use('ggplot')
_ = pd.plotting.scatter_matrix(df, c=y, figsize=[8, 8], s=150,
                               marker='D')  # 'D' means diamond.
```



3. Practice exercises for exploratory data analysis (EDA):

▶ **Data pre-loading:**

[8]:
```
import pandas as pd

df = pd.read_csv(
    "https://archive.ics.uci.edu/ml/machine-learning-databases/voting-records/
↪house-votes-84.data",
    header=None)
df.columns = [
    'party', 'infants', 'water', 'budget', 'physician', 'salvador',
    'religious', 'satellite', 'aid', 'missile', 'immigration', 'synfuels',
    'education', 'superfund', 'crime', 'duty_free_exports', 'eaa_rsa'
]
df.replace(['y', 'n', '?'], [1, 0, 0.5], inplace=True)
```

▶ **Numerical EDA practice:**

[9]:
```
df.shape
```

[9]:
```
(435, 17)
```

[10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 435 entries, 0 to 434
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   party              435 non-null    object
 1   infants            435 non-null    float64
 2   water              435 non-null    float64
 3   budget             435 non-null    float64
 4   physician          435 non-null    float64
 5   salvador           435 non-null    float64
 6   religious          435 non-null    float64
 7   satellite          435 non-null    float64
 8   aid                435 non-null    float64
 9   missile            435 non-null    float64
 10  immigration        435 non-null    float64
 11  synfuels           435 non-null    float64
 12  education          435 non-null    float64
 13  superfund          435 non-null    float64
 14  crime              435 non-null    float64
 15  duty_free_exports  435 non-null    float64
```

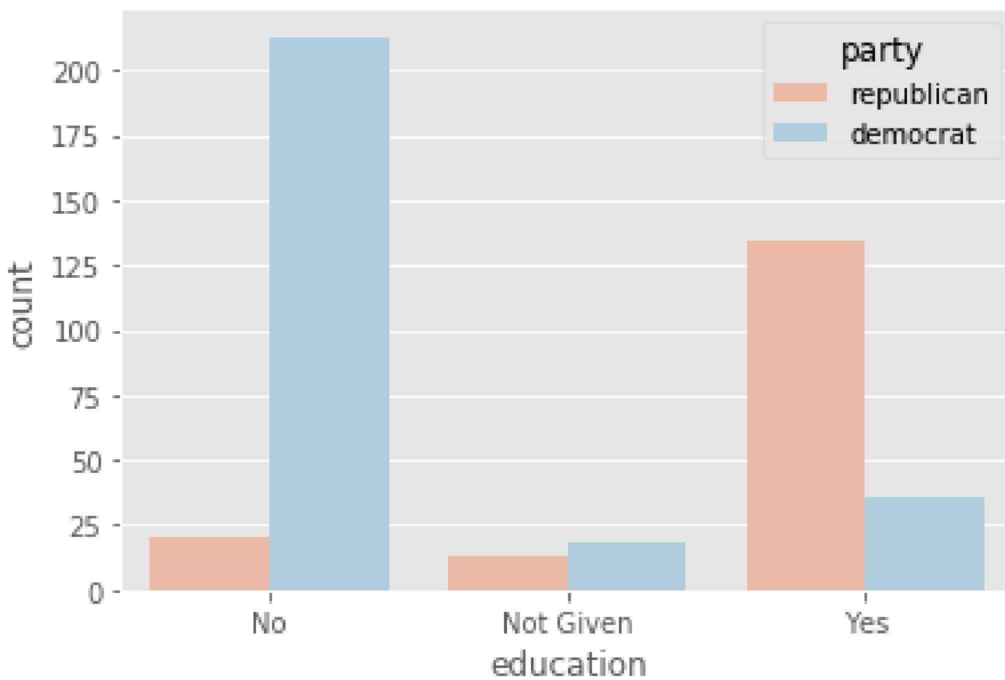
```
 16   eaa_rsa             435 non-null    float64
dtypes: float64(16), object(1)
memory usage: 57.9+ KB
```

[11]:
```
df['party'].head()
```

[11]:
```
0    republican
1    republican
2      democrat
3      democrat
4      democrat
Name: party, dtype: object
```

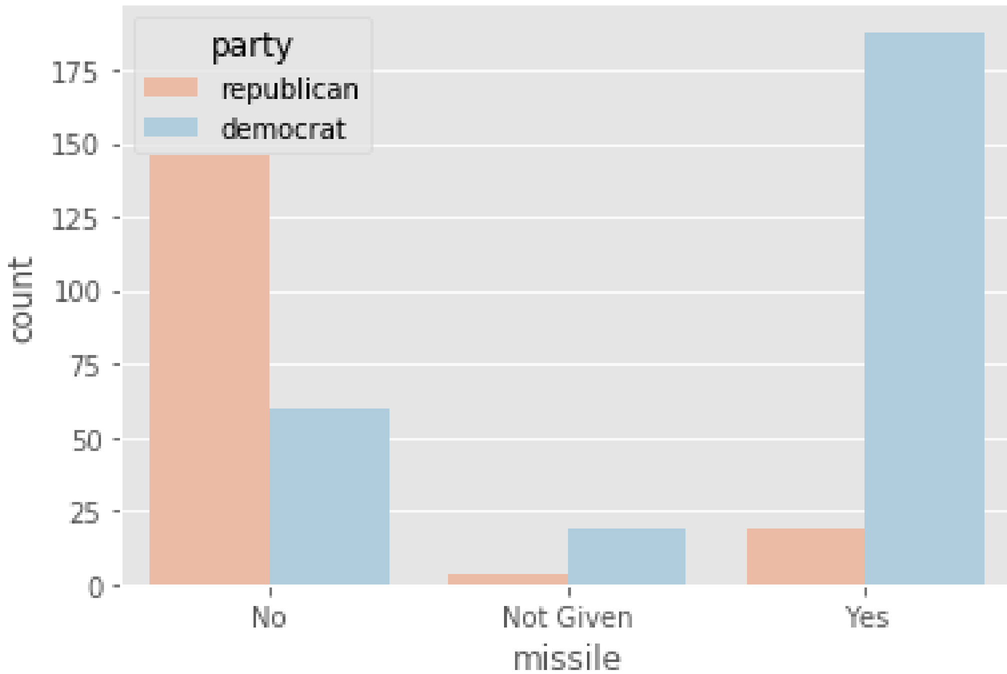
▶ **Visual EDA practice:**

[12]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure()
sns.countplot(x='education', hue='party', data=df, palette='RdBu')
plt.xticks([0, 1, 2], ['No', 'Not Given', 'Yes'])
plt.show()
```

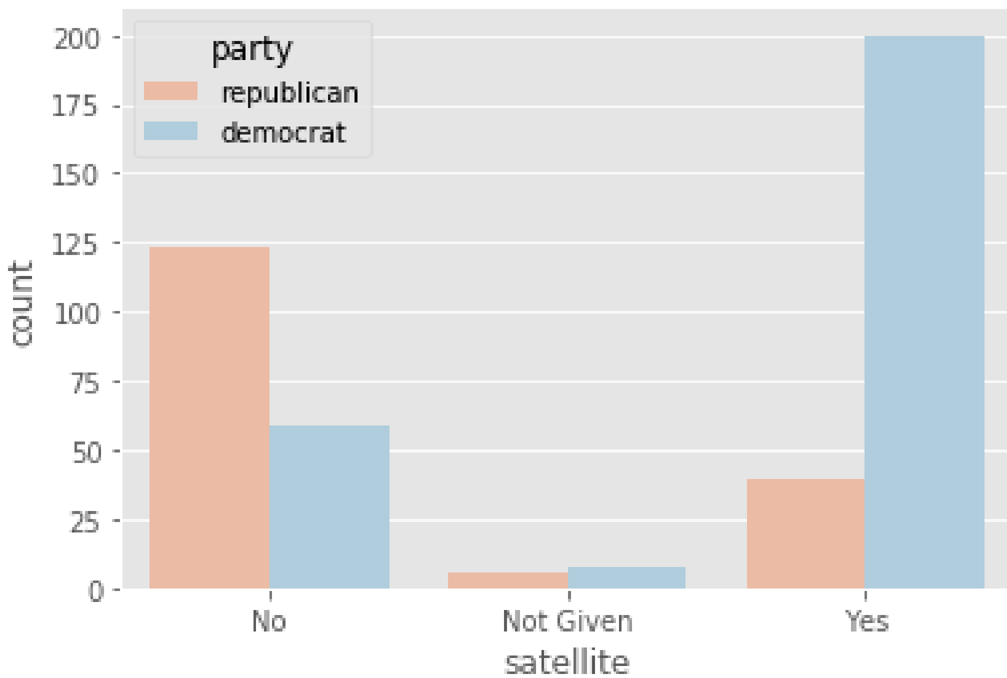


[13]:
```
plt.figure()
sns.countplot(x='missile', hue='party', data=df, palette='RdBu')
```

```
plt.xticks([0, 1, 2], ['No', 'Not Given', 'Yes'])
plt.show()
```



[14]:
```
plt.figure()
sns.countplot(x='satellite', hue='party', data=df, palette='RdBu')
plt.xticks([0, 1, 2], ['No', 'Not Given', 'Yes'])
plt.show()
```



3 The classification challenge

1. What is the basic idea of the k-nearest neighbors algorithm (k-NN)?

- Predict the label of a data point by:
  - looking at the 'k' closet labeled data points;
  - taking a majority vote.

2. How to fit and predict by scikit-learn?

- Training a model on the data = 'fitting' a model to the data:
  - `.fit()` method
- To predict the labels of new data:
  - `.predict()` method

3. Code to fit a classifier by using scikit-learn:

[15]:
```
from sklearn import datasets
from sklearn.neighbors import KNeighborsClassifier

iris = datasets.load_iris()
knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(iris['data'], iris['target'])
```

[15]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=6, p=2,
                     weights='uniform')

```
[16]: iris['data'].shape
```

[16]: (150, 4)

```
[17]: iris['target'].shape
```

[17]: (150,)

4. Code to predict on unlabeled data by using scikit-learn:

```
[18]: import numpy as np

X_new = np.array([[5.6, 2.8, 3.9, 1.1], [5.7, 2.6, 3.8, 1.3],
                  [4.7, 3.2, 1.3, 0.2]])
prediction = knn.predict(X_new)
X_new.shape
```

[18]: (3, 4)

```
[19]: print('Prediction: {}'.format(prediction))
```

Prediction: [1 1 0]

5. Practice exercise for the classification challenge:

▶ **Data pre-loading:**

```
[20]: import pandas as pd

df = pd.read_csv(
    "https://archive.ics.uci.edu/ml/machine-learning-databases/voting-records/
 ↪house-votes-84.data",
    header=None)
df.columns = [
    'party', 'infants', 'water', 'budget', 'physician', 'salvador',
    'religious', 'satellite', 'aid', 'missile', 'immigration', 'synfuels',
    'education', 'superfund', 'crime', 'duty_free_exports', 'eaa_rsa'
]
df.replace(['y', 'n', '?'], [1, 0, 0.5], inplace=True)
X_new = pd.DataFrame([[
    0.69646919, 0.28613933, 0.22685145, 0.55131477, 0.71946897, 0.42310646,
    0.9807642, 0.68482974, 0.4809319, 0.39211752, 0.34317802, 0.72904971,
    0.43857224, 0.0596779, 0.39804426, 0.73799541
]])
```

► **Fitting practice for k-nearest neighbors:**

[21]:
```python
# Import KNeighborsClassifier from sklearn.neighbors
from sklearn.neighbors import KNeighborsClassifier

# Create arrays for the features and the response variable
y = df['party'].values
X = df.drop('party', axis=1).values

# Create a k-NN classifier with 6 neighbors
knn = KNeighborsClassifier(n_neighbors=6)

# Fit the classifier to the data
knn.fit(X, y)
```

[21]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=6, p=2,
                     weights='uniform')
```

► **Predicting practice for k-nearest neighbors:**

[22]:
```python
# Import KNeighborsClassifier from sklearn.neighbors
from sklearn.neighbors import KNeighborsClassifier

# Create arrays for the features and the response variable
y = df['party']
X = df.drop('party', axis=1)

# Create a k-NN classifier with 6 neighbors: knn
knn = KNeighborsClassifier(n_neighbors=6)

# Fit the classifier to the data
knn.fit(X, y)

# Predict the labels for the training data X
y_pred = knn.predict(X)

# Predict and print the label for the new data point X_new
new_prediction = knn.predict(X_new)
print("Prediction: {}".format(new_prediction))
```

```
Prediction: ['democrat']
```

4 Measuring model performance

1. What is the accuracy?

- In classification, accuracy is a commonly used metric.

- Accuracy = *fraction of correct predictions*

2. **How to measuring model performance?**

   - Split data into training and test set.
   - Fit/train the classifier on the training set.
   - Make predictions on the test set.
   - Compare predictions with the known labels.

3. **Code of train/test split:**

[23]:
```python
from sklearn import datasets
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

iris = datasets.load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.3,
                                                    random_state=21,
                                                    stratify=y)
knn = KNeighborsClassifier(n_neighbors=8)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Test set predictions:\n {}".format(y_pred))
```

```
Test set predictions:
 [2 1 2 2 1 0 1 0 0 1 0 2 0 2 2 0 0 0 1 0 2 2 2 0 1 1 1 0 0 1 2 2 0 0 1 2 2
 1 1 2 1 1 0 2 1]
```

[24]:
```python
knn.score(X_test, y_test)
```

[24]: 0.9555555555555556

4. **What is model complexity?**

   - Larger k:
     - smoother decision boundary
     - less complex model
   - Smaller k:
     - more complex model
     - can lead to overfitting

5. **Practice exercise for measuring model performance:**

▶ **Package pre-loading:**

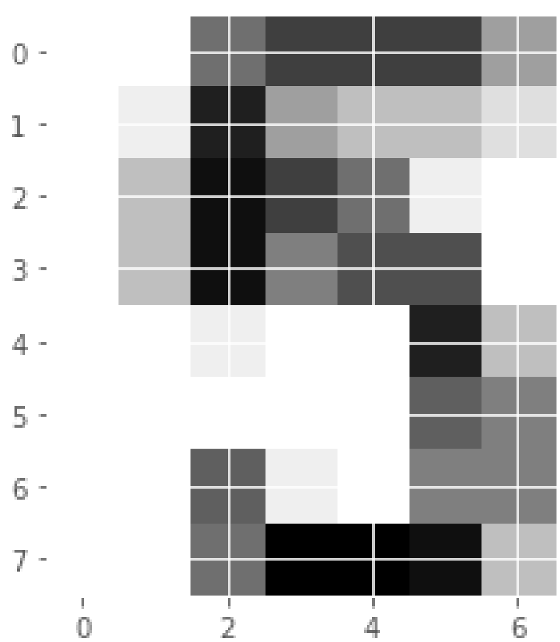
[25]:
```
import numpy as np
```

▶ **The digits recognition dataset practice:**

[26]:
```
# Import necessary modules
from sklearn import datasets
import matplotlib.pyplot as plt

# Load the digits dataset: digits
digits = datasets.load_digits()

# Print the keys and DESCR of the dataset
print(digits.DESCR)

# Print the shape of the images and data keys
print(digits.images.shape)
print(digits.data.shape)

# Display digit 1010
plt.imshow(digits.images[1010], cmap=plt.cm.gray_r, interpolation='nearest')
plt.show()
```

```
.. _digits_dataset:

Optical recognition of handwritten digits dataset
--------------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 5620
    :Number of Attributes: 64
    :Attribute Information: 8x8 image of integer pixels in the range 0..16.
    :Missing Attribute Values: None
    :Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)
    :Date: July; 1998

This is a copy of the test set of the UCI ML hand-written digits datasets
https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digit
s

The data set contains images of hand-written digits: 10 classes where
each class refers to a digit.

Preprocessing programs made available by NIST were used to extract
normalized bitmaps of handwritten digits from a preprinted form. From a
total of 43 people, 30 contributed to the training set and different 13
```

```
to the test set. 32x32 bitmaps are divided into nonoverlapping blocks of
4x4 and the number of on pixels are counted in each block. This generates
an input matrix of 8x8 where each element is an integer in the range
0..16. This reduces dimensionality and gives invariance to small
distortions.

For info on NIST preprocessing routines, see M. D. Garris, J. L. Blue, G.
T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C.
L. Wilson, NIST Form-Based Handprint Recognition System, NISTIR 5469,
1994.

.. topic:: References

  - C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their
    Applications to Handwritten Digit Recognition, MSc Thesis, Institute of
    Graduate Studies in Science and Engineering, Bogazici University.
  - E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.
  - Ken Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin.
    Linear dimensionalityreduction using relevance weighted LDA. School of
    Electrical and Electronic Engineering Nanyang Technological University.
    2005.
  - Claudio Gentile. A New Approximate Maximal Margin Classification
    Algorithm. NIPS. 2000.
(1797, 8, 8)
(1797, 64)
```



▶ **Train/test split and fit/predict/accuracy practice:**

[27]:
```python
# Import necessary modules
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# Create feature and target arrays
X = digits.data
y = digits.target

# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.2,
                                                    random_state=42,
                                                    stratify=y)

# Create a k-NN classifier with 7 neighbors: knn
knn = KNeighborsClassifier(n_neighbors=7)

# Fit the classifier to the training data
knn.fit(X_train, y_train)

# Print the accuracy
print(knn.score(X_test, y_test))
```

```
0.9833333333333333
```

▶ **Overfitting and underfitting practice:**

[28]:
```python
# Setup arrays to store train and test accuracies
neighbors = np.arange(1, 9)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

# Loop over different values of k
for i, k in enumerate(neighbors):
    # Setup a k-NN Classifier with k neighbors: knn
    knn = KNeighborsClassifier(n_neighbors=k)

    # Fit the classifier to the training data
    knn.fit(X_train, y_train)

    #Compute accuracy on the training set
    train_accuracy[i] = knn.score(X_train, y_train)

    #Compute accuracy on the testing set
    test_accuracy[i] = knn.score(X_test, y_test)
```

```python
# Generate plot
plt.title('k-NN: Varying Number of Neighbors')
plt.plot(neighbors, test_accuracy, label='Testing Accuracy')
plt.plot(neighbors, train_accuracy, label='Training Accuracy')
plt.legend()
plt.xlabel('Number of Neighbors')
plt.ylabel('Accuracy')
plt.show()
```