

Course Distribution and Lab Manual

Prepared By: Assoc. Prof. Dr. Salha Alzahrani

Week	Theory Topic	Lab Topic	Comments
1	Data Mining Course Info.	-	
2	Lecture 1: Introduction to Data Mining and Knowledge Discovery	-	
3	Lecture 2: Data Processing	Topic 1: Using Anaconda for Python and Introduction to Python	
4	Lecture 3: Classification using Naïve Bayes Algorithm	Topic 2: More about Python	
5	Lecture 4: Classification using Nearest Neighbour Algorithm	Topic 3: Data Processing in Python	
6	Lecture 5: Classification using Decision Trees	Topic 4: Using SciKitLearn for Data Mining in Python	
7	Handouts & Quiz 1	Topic 5: Using SciKitLearn for Naïve Bayes Classification	
8	Mid-Term Exam	Topic 6: Using SciKitLearn for Nearest Neighbour Classification	
9	Lecture 6: Inducing Modular Rules for Classification	Topic 7: Using SciKitLearn for Decision Trees Classification	
10	Lecture 7: Association Rule Mining I	Topic 8: More Examples on Classification	
11	Lecture 8: Association Rule Mining II	Topic 9: Using SciKitLearn for Clustering	
12	Lecture 9: Clustering	Revision.	
13	Cont. Clustering & Quiz 2	Lab Final.	
14	Revision & End of Course		

Code and Materials are available at:

<https://github.com/SalhaAlzahrani>

Lab Manual

Table of Contents

Data Mining – Spring 2020	1
Prepared By: Assoc. Prof. Dr. Salha Alzahrani	1
Topic 1: Using Anaconda for Python and Introduction to Python	4
Topic 1.1 Using Anaconda for Python	4
What is Anaconda Navigator?	4
Using the command-line install	13
Topic 1.2 Introduction to Python	15
Topic 2: More about Python.....	16
2.1 Lists	16
2.2 If statements.....	16
2.3 User input	16
2.4 While loop	16
2.5 Functions	16
Topic 3: Data Processing in Python	20
3.1 Installing scikit-learn.....	20
3.2 Other important libraries	20
3.3 Preprocessing data	21
3.3.1. Standardization (or mean removal).....	21
3.3.2. Scaling to a range	22
3.3.3. Normalization	23
Topic 4: Using SciKitLearn for Data Mining in Python	24
4.1 Loading an example dataset.....	24
4.2 Feature selection	26
4.2.1. Removing features with low variance	26
4.2.2. Univariate feature selection	27
Topic 5: Using SciKitLearn for Naïve Bayes Classification.....	28
5.1 Play Dataset	28
5.2 Irish Dataset.....	30
Topic 6: Using SciKitLearn for Nearest Neighbour Classification	32
6.1 Play Dataset	32
Using k=3	32

Using k=7	33
6.2 Irish Dataset.....	34
Using k=7	34
Using k=15	36
6.4 Other Numerical Datasets	38
6.4.1. Example of numerical array.....	38
6.4.2. Example of prices.....	39
Topic 7: Using SciKitLearn for Decision Trees Classification.....	40
7.1 Decision Tree Classifier for a Simple Numerical Dataset.....	40
7.2 Decision Tree Classifier for Irish Dataset.....	42
Topic 8: More Examples on Classification	44
8.1 Data	44
Raw dataset:.....	44
Training dataset:.....	45
Test dataset:.....	45
8.2 Comparison between Naïve Bayes, Nearest Neighbour and Decision Tree Classifiers.....	46
Topic 9: Using SciKitLearn for Clustering.....	48
9.1. Overview of clustering methods	48
9.2. K-Means Clustering of Numerical Dataset	49
9.2. K-Means Clustering of Irish Dataset and Visualization.....	50
9.3. K-Means Clustering of Irish Dataset and Visualization using 3 clusters, 8 clusters, and 3 with random/bad initialization.....	52

Lab Manual

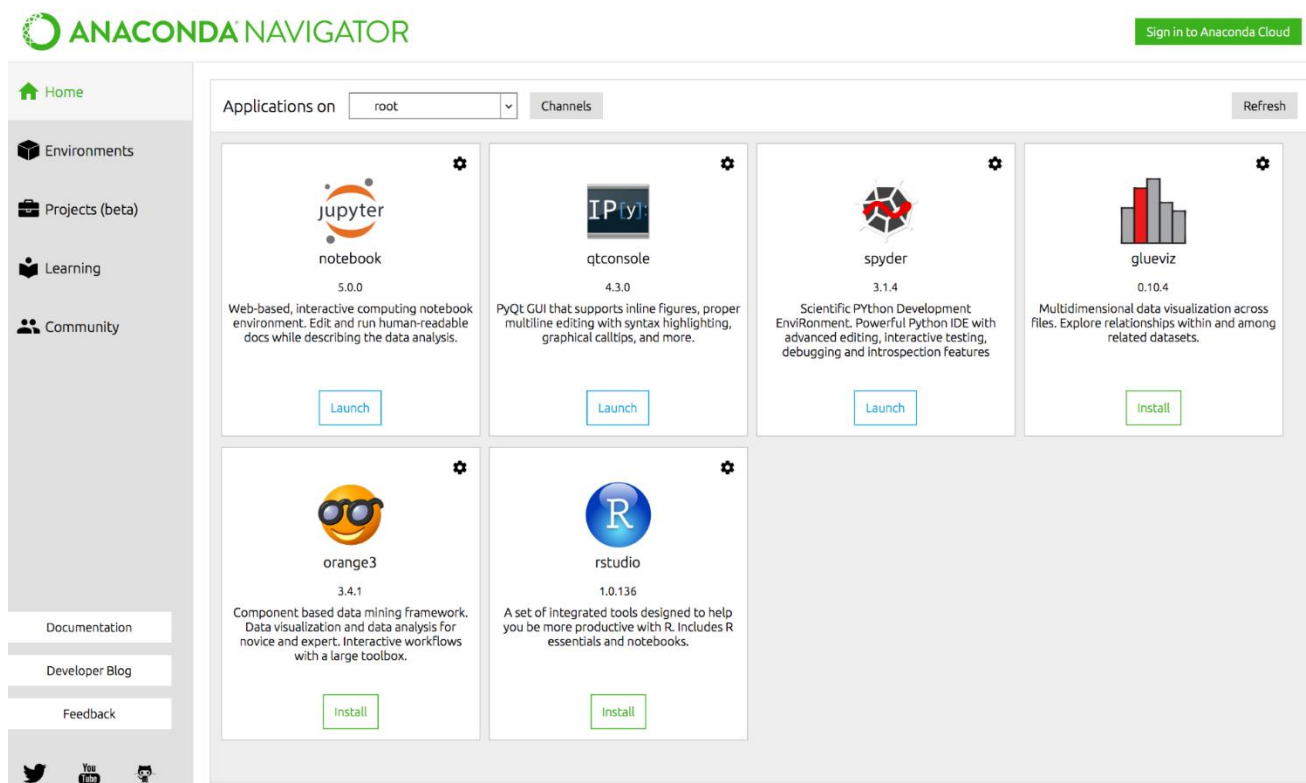
Topic 1: Using Anaconda for Python and Introduction to Python

Topic 1.1 Using Anaconda for Python

Anaconda Navigator

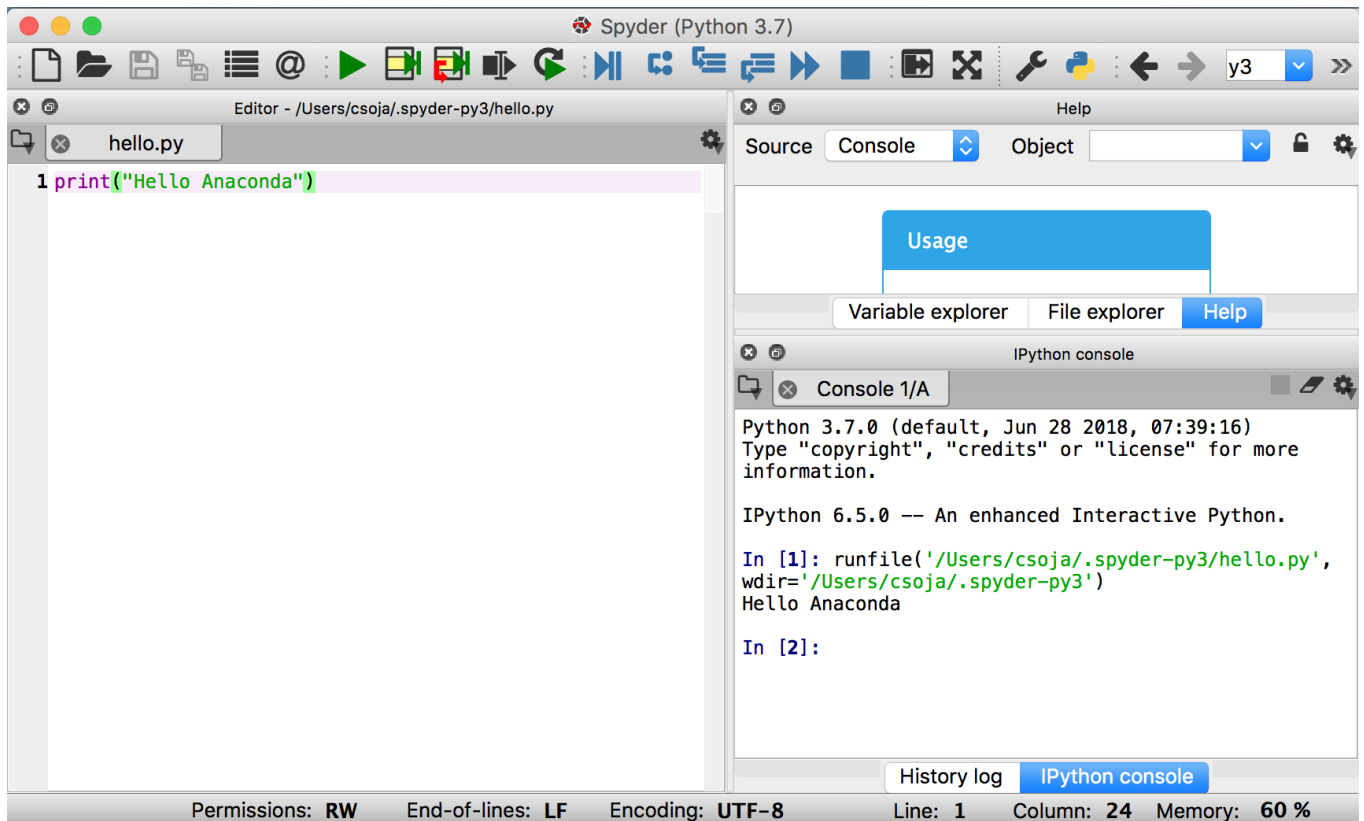
What is Anaconda Navigator?

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, macOS, and Linux.



Run Python in Spyder IDE (integrated development environment)

1. Launch Spyder by clicking Spyder's Launch button.
2. In the new file on the left, delete any placeholder text, then type or copy/paste `print("Hello Anaconda")`.
3. In the top menu, click File - Save As and name your new program `hello.py`.
4. Run your new program by clicking the triangle Run button.
5. You can see your program's output in the bottom right Console pane.



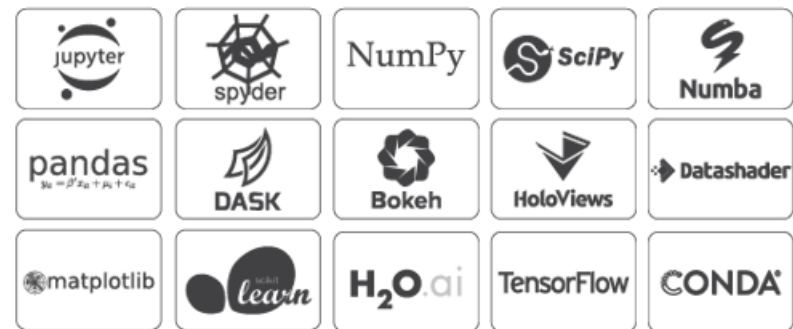
Anaconda Distribution

The World's Most Popular Python/R Data Science Platform

Download

The open-source [Anaconda Distribution](#) is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 15 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with [Conda](#)
- Develop and train machine learning and deep learning models with [scikit-learn](#), [TensorFlow](#), and [Theano](#)
- Analyze data with scalability and performance with [Dask](#), [NumPy](#), [pandas](#), and [Numba](#)
- Visualize results with [Matplotlib](#), [Bokeh](#), [Datashader](#), and [Holoviews](#)



Python 3.7 version

[64-Bit Graphical Installer \(653 MB\)](#)

[64-Bit Command Line Installer \(435 MB\)](#)



Python 3.7 version

[64-Bit Graphical Installer \(486 MB\)](#)

[32-Bit Graphical Installer \(418 MB\)](#)

Installing on Windows

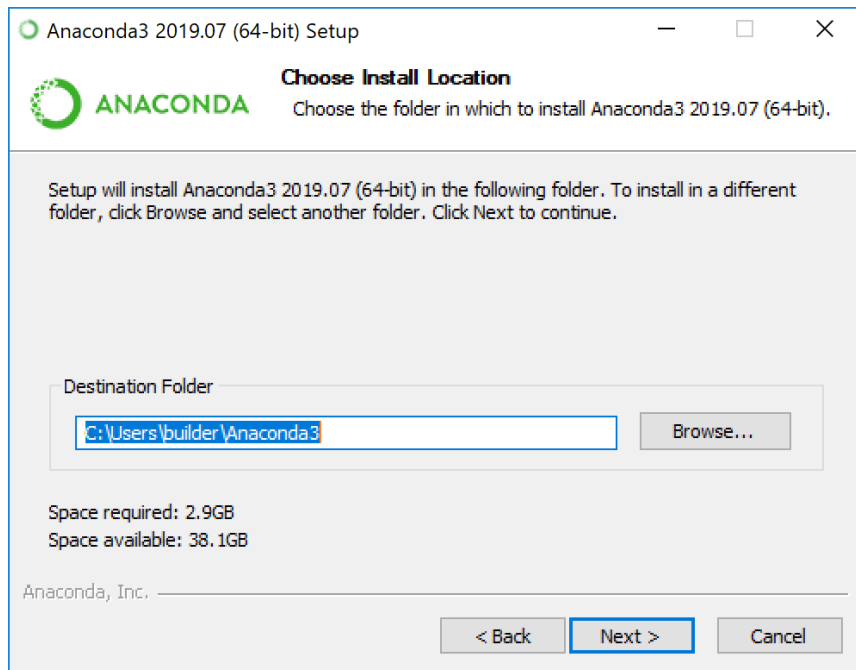
1. Double click the installer to launch.
2. Click Next.
3. Read the licensing terms and click “I Agree”.
4. Select an install for “Just Me” unless you’re installing for all users (which requires Windows Administrator privileges) and click Next.
5. Select a destination folder to install Anaconda and click the Next button. See [FAQ](#).

Note

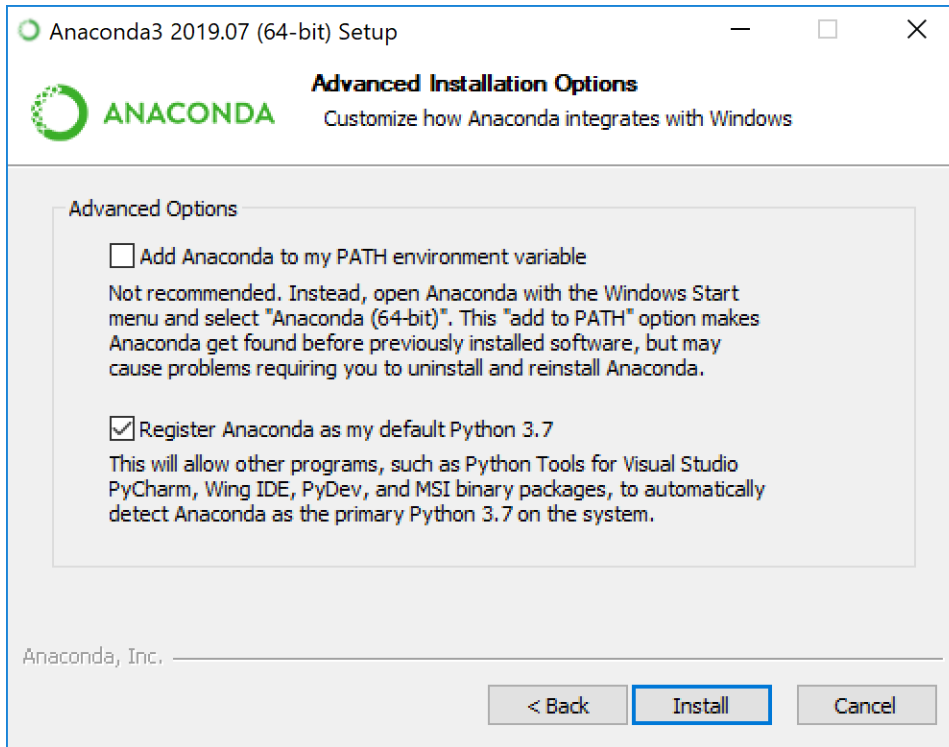
Install Anaconda to a directory path that does not contain spaces or unicode characters.

Note

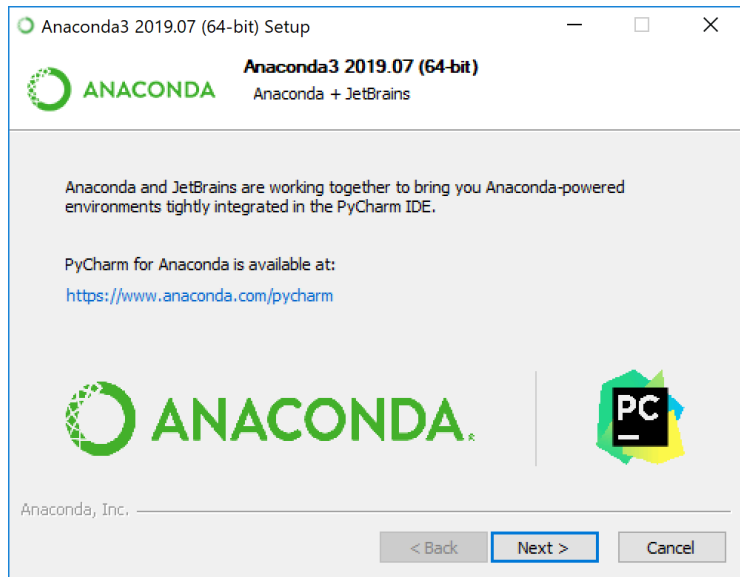
Do not install as Administrator unless admin privileges are required.



6. Choose whether to add Anaconda to your PATH environment variable. We recommend not adding Anaconda to the PATH environment variable, since this can interfere with other software. Instead, use Anaconda software by opening Anaconda Navigator or the Anaconda Prompt from the Start Menu.

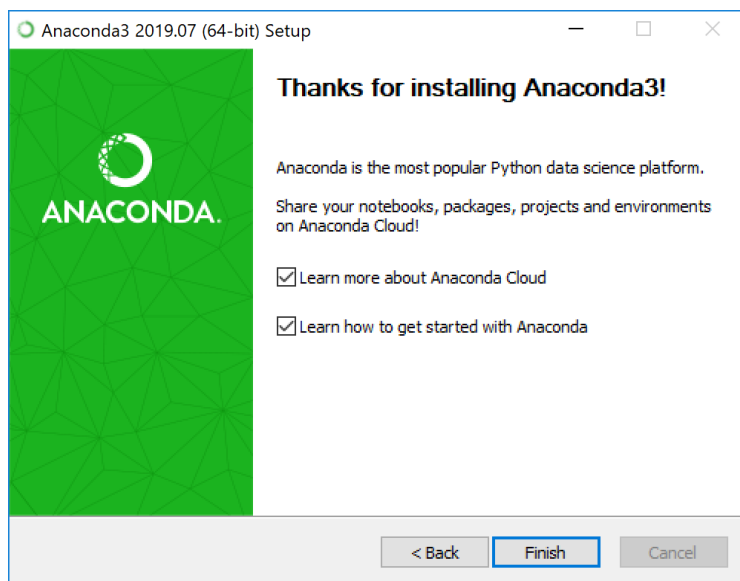


7. Choose whether to register Anaconda as your default Python. Unless you plan on installing and running multiple versions of Anaconda, or multiple versions of Python, accept the default and leave this box checked.
8. Click the Install button. If you want to watch the packages Anaconda is installing, click Show Details.
9. Click the Next button.
10. Optional: To install PyCharm for Anaconda, click on the link to <https://www.anaconda.com/pycharm>.

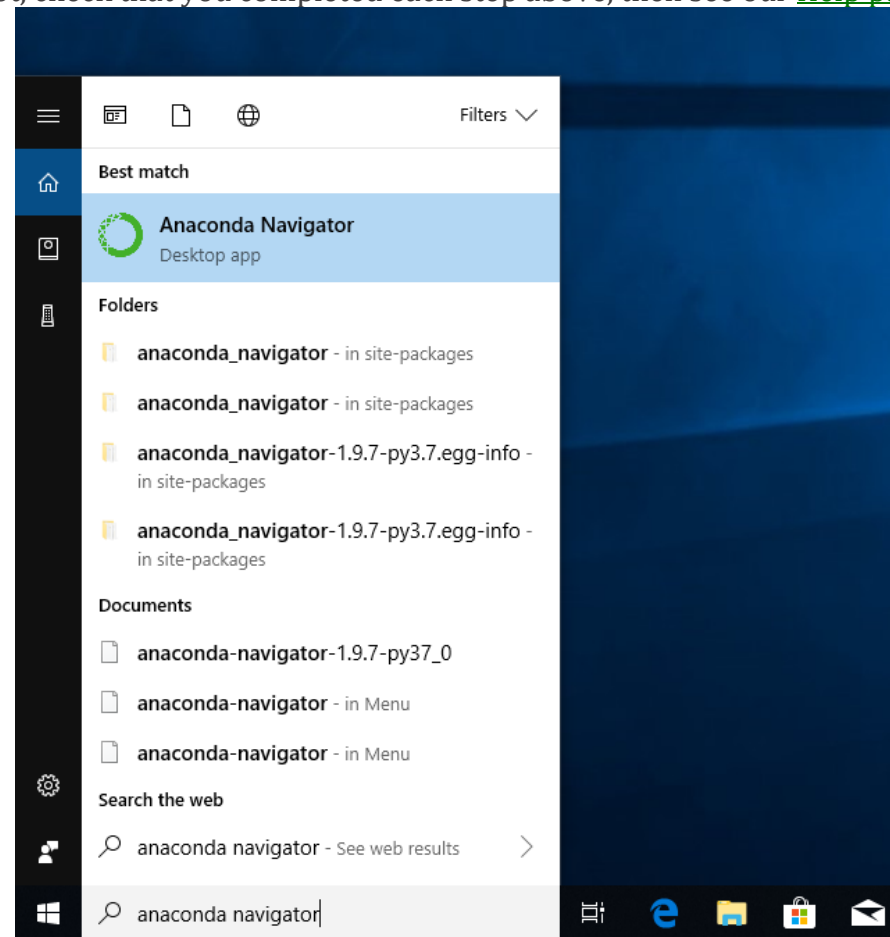


Or to install Anaconda without PyCharm, click the Next button.

11. After a successful installation you will see the “Thanks for installing Anaconda” dialog box:

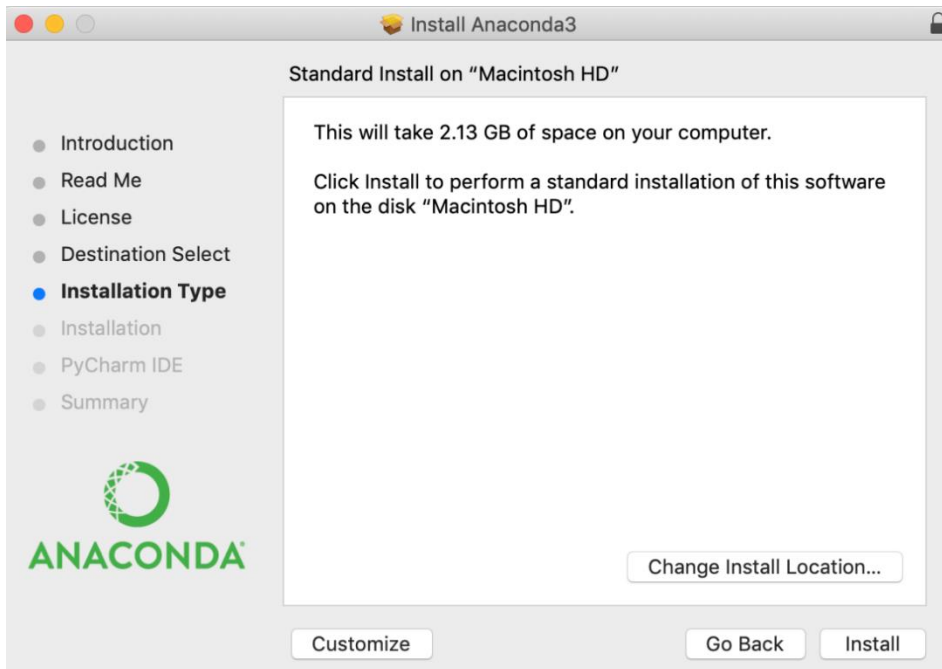


12. If you wish to read more about Anaconda Cloud and how to get started with Anaconda, check the boxes “Learn more about Anaconda Cloud” and “Learn how to get started with Anaconda”. Click the Finish button.
13. After your install is complete, verify it by opening Anaconda Navigator, a program that is included with Anaconda: from your Windows Start menu, select the shortcut Anaconda Navigator from the Recently added or by typing “Anaconda Navigator”. If Navigator opens, you have successfully installed Anaconda. If not, check that you completed each step above, then see our [Help page](#).



Installing on macOS

1. Double-click the downloaded file and click continue to start the installation.
2. Answer the prompts on the Introduction, Read Me, and License screens.
3. Click the Install button to install Anaconda in your home user directory (recommended):

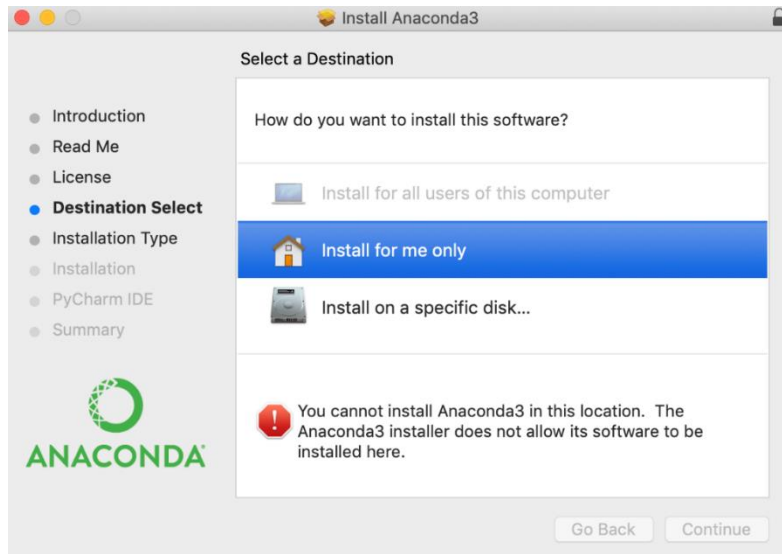


4. OR, click the Change Install Location button to install in another location (not recommended).

On the Destination Select screen, select Install for me only.

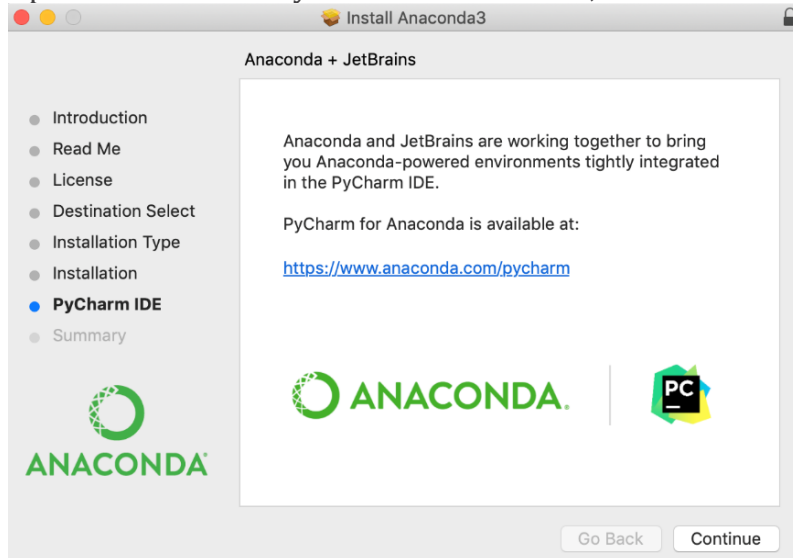
Note

If you get the error message “You cannot install Anaconda in this location,” reselect Install for me only.



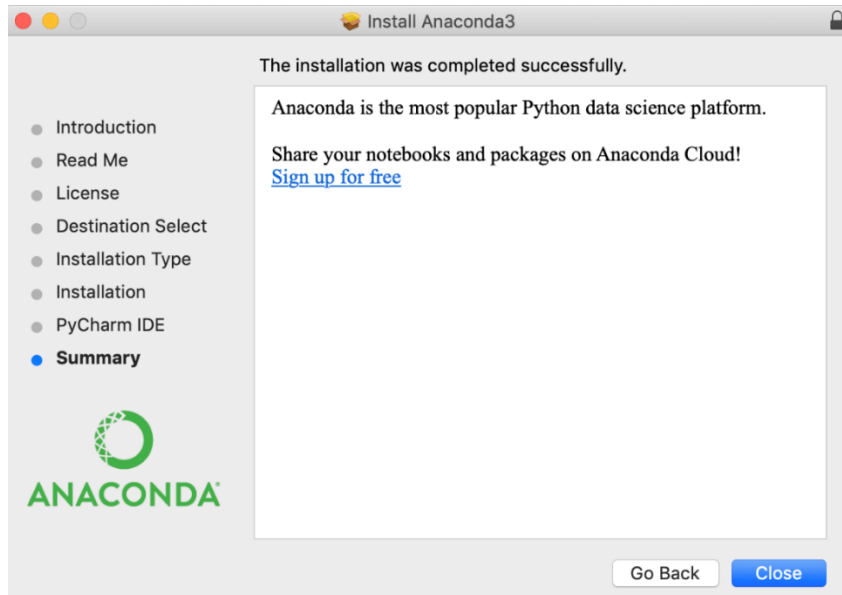
5. Click the continue button.

6. Optional: To install PyCharm for Anaconda, click on the link to <https://www.anaconda.com/pycharm>.



Or to install Anaconda without PyCharm, click the Continue button.

7. A successful installation displays the following screen:



8. After your install is complete, verify it by opening Anaconda Navigator, a program that is included with Anaconda: from Launchpad, select Anaconda Navigator. If Navigator opens, you have successfully installed Anaconda. If not, check that you completed each step above, then see our [Help page](#).

Tip

For more information about using Anaconda-Navigator, see [Navigator](#).

Using the command-line install

Use this method if you prefer to use a terminal window.

1. In your browser, download the command-line version of the [macOS installer](#) for your system.
2. [Verify data integrity with SHA-256](#). For more information on hash verification, see [cryptographic hash validation](#).
- Open a terminal and run the following:

```
• shasum -a 256 /path/filename
```

3. Install for Python 3.7 or 2.7:

- For Python 3.7 enter the following:

```
• bash ~/Downloads/Anaconda3-2019.07-MacOSX-x86_64.sh
```

Note

Include the `bash` command regardless of whether or not you are using the Bash shell.

Note

Replace `~/Downloads` with your actual path and `Anaconda3-2019.07-MacOSX-x86_64.sh` with actual name of the file you downloaded.

4. The installer prompts “In order to continue the installation process, please review the license agreement.” Click Enter to view license terms.
5. Scroll to the bottom of the license terms and enter yes to agree to them.
6. The installer prompts you to Press Enter to confirm the location, Press CTRL-C to cancel the installation or specify an alternate installation directory. If you accept the default install location, the installer displays “PREFIX=/home/<user>/anaconda<2 or 3>” and continues the installation. It may take a few minutes to complete.

Note

We recommend you accept the default install location. Do not choose the path as `/usr` for the Anaconda/Miniconda installation.

7. The installer prompts “Do you wish the installer to initialize Anaconda3 by running `conda init`?” We recommend “yes”.

Note

If you enter “no”, then conda will not modify your shell scripts at all. In order to initialize after the installation process is done, first run `source <path to conda>/bin/activate` and then run `conda init`.

8. The installer displays “Thank you for installing Anaconda!”
9. Optional: The installer describes the partnership between Anaconda and JetBrains and provides a link to install PyCharm for Anaconda at <https://www.anaconda.com/pycharm>.
10. Close and open your terminal window for the Anaconda installation to take effect.
11. To control whether or not each shell session has the base environment activated or not, run `conda config --set auto_activate_base False` or `True`. To run conda from anywhere without having the base environment activated by default, use `conda config --set auto_activate_base False`. This only works if you have run `conda init` first.

Note

`conda init` is available in conda versions 4.6.12 and later.

Variables and Strings

Variables are used to store values. A string is a series of characters, surrounded by single or double quotes.

Hello world

```
print("Hello world!")
```

Hello world with a variable

```
msg = "Hello world!"  
print(msg)
```

Concatenation (combining strings)

```
first_name = 'albert'  
last_name = 'einstein'  
full_name = first_name + ' ' + last_name  
print(full_name)
```

Topic 2: More about Python

2.1 Lists

2.2 If statements

2.3 User input

2.4 While loop

2.5 Functions

Lists

A list stores a series of items in a particular order. You access items using an index, or within a loop.

Make a list

```
bikes = ['trek', 'redline', 'giant']
```

Get the first item in a list

```
first_bike = bikes[0]
```

Get the last item in a list

```
last_bike = bikes[-1]
```

Looping through a list

```
for bike in bikes:  
    print(bike)
```

Adding items to a list

```
bikes = []  
bikes.append('trek')  
bikes.append('redline')  
bikes.append('giant')
```

Making numerical lists

```
squares = []  
for x in range(1, 11):  
    squares.append(x**2)
```


If statements

If statements are used to test for particular conditions and respond appropriately.

Conditional tests

equals	<code>x == 42</code>
not equal	<code>x != 42</code>
greater than	<code>x > 42</code>
or equal to	<code>x >= 42</code>
less than	<code>x < 42</code>
or equal to	<code>x <= 42</code>

Conditional test with lists

```
'trek' in bikes  
'surly' not in bikes
```

Boolean values

```
game_active = True  
can_edit = False
```

A simple if test

```
if age >= 18:  
    print("You can vote!")
```

If-elif-else statements

```
if age < 4:  
    ticket_price = 0  
elif age < 18:  
    ticket_price = 10  
else:  
    ticket_price = 15
```

User input

Your programs can prompt the user for input. All input is stored as a string.

Prompting for a value

```
name = input("What's your name? ")
print("Hello, " + name + "!")
```

Prompting for numerical input

```
age = input("How old are you? ")
age = int(age)

pi = input("What's the value of pi? ")
pi = float(pi)
```

While loops

A while loop repeats a block of code as long as a certain condition is true.

A simple while loop

```
current_value = 1
while current_value <= 5:
    print(current_value)
    current_value += 1
```

Letting the user choose when to quit

```
msg = ''
while msg != 'quit':
    msg = input("What's your message? ")
    print(msg)
```

Functions

Functions are named blocks of code, designed to do one specific job. Information passed to a function is called an argument, and information received by a function is called a parameter.

A simple function

```
def greet_user():  
    """Display a simple greeting."""  
    print("Hello!")  
  
greet_user()
```

Passing an argument

```
def greet_user(username):  
    """Display a personalized greeting."""  
    print("Hello, " + username + "!")  
  
greet_user('jesse')
```

Default values for parameters

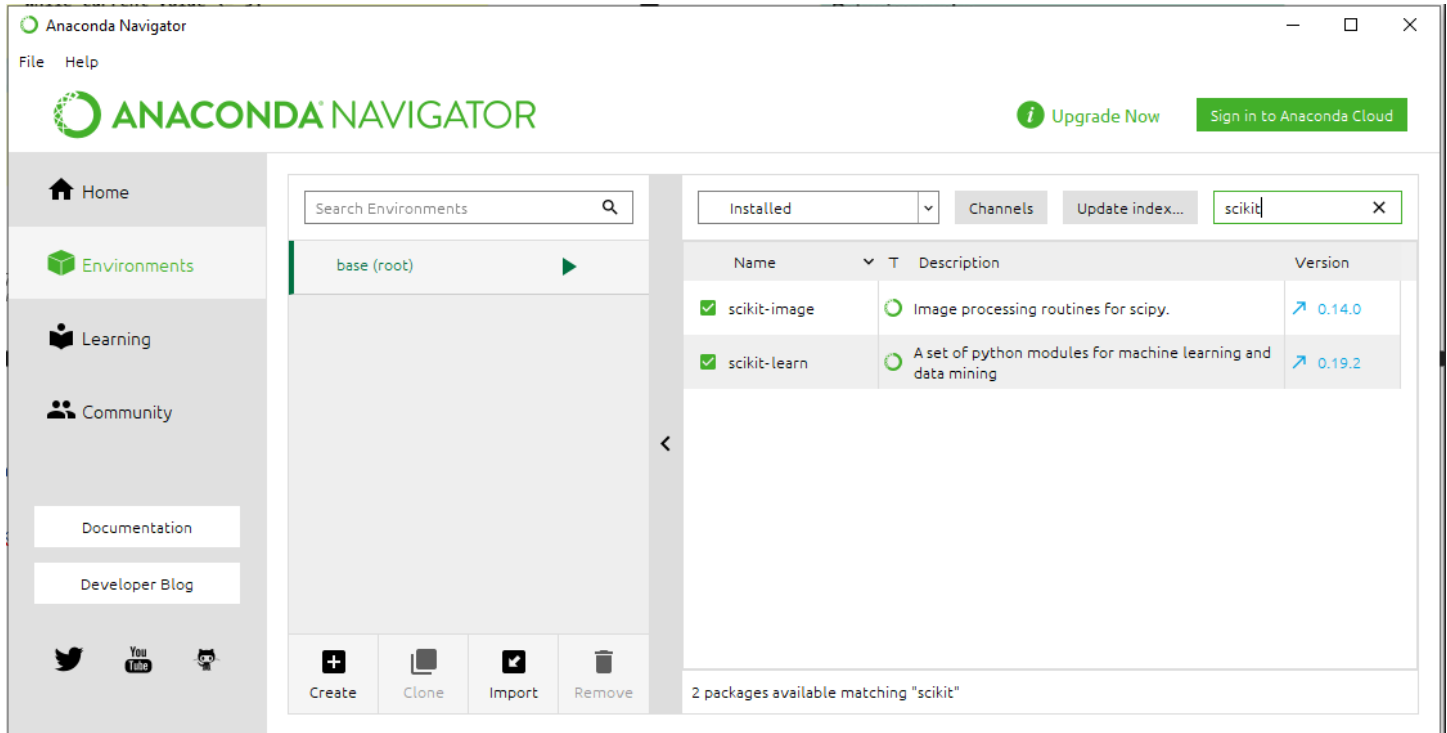
```
def make_pizza(topping='bacon'):  
    """Make a single-topping pizza."""  
    print("Have a " + topping + " pizza!")  
  
make_pizza()  
make_pizza('pepperoni')
```

Returning a value

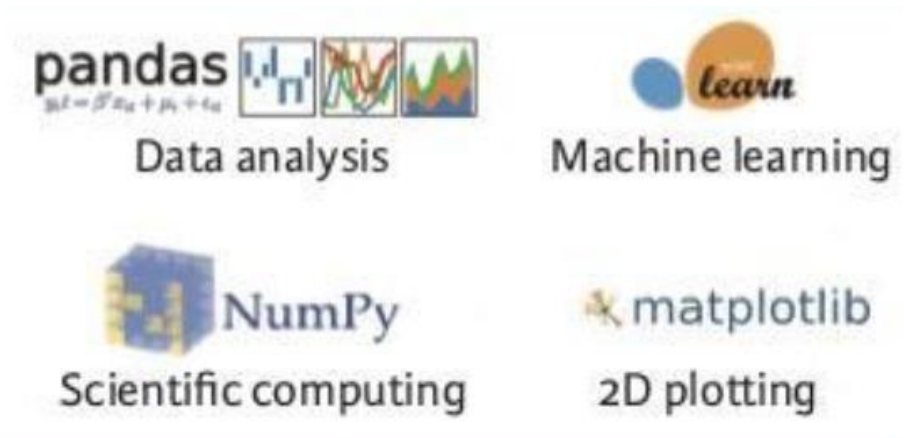
```
def add_numbers(x, y):  
    """Add two numbers and return the sum."""  
    return x + y  
  
sum = add_numbers(3, 5)  
print(sum)
```

Topic 3: Data Processing in Python

3.1 Installing scikit-learn



3.2 Other important libraries



How to use a library?

Libraries

Import libraries

```
>>> import numpy
>>> import numpy as np
```

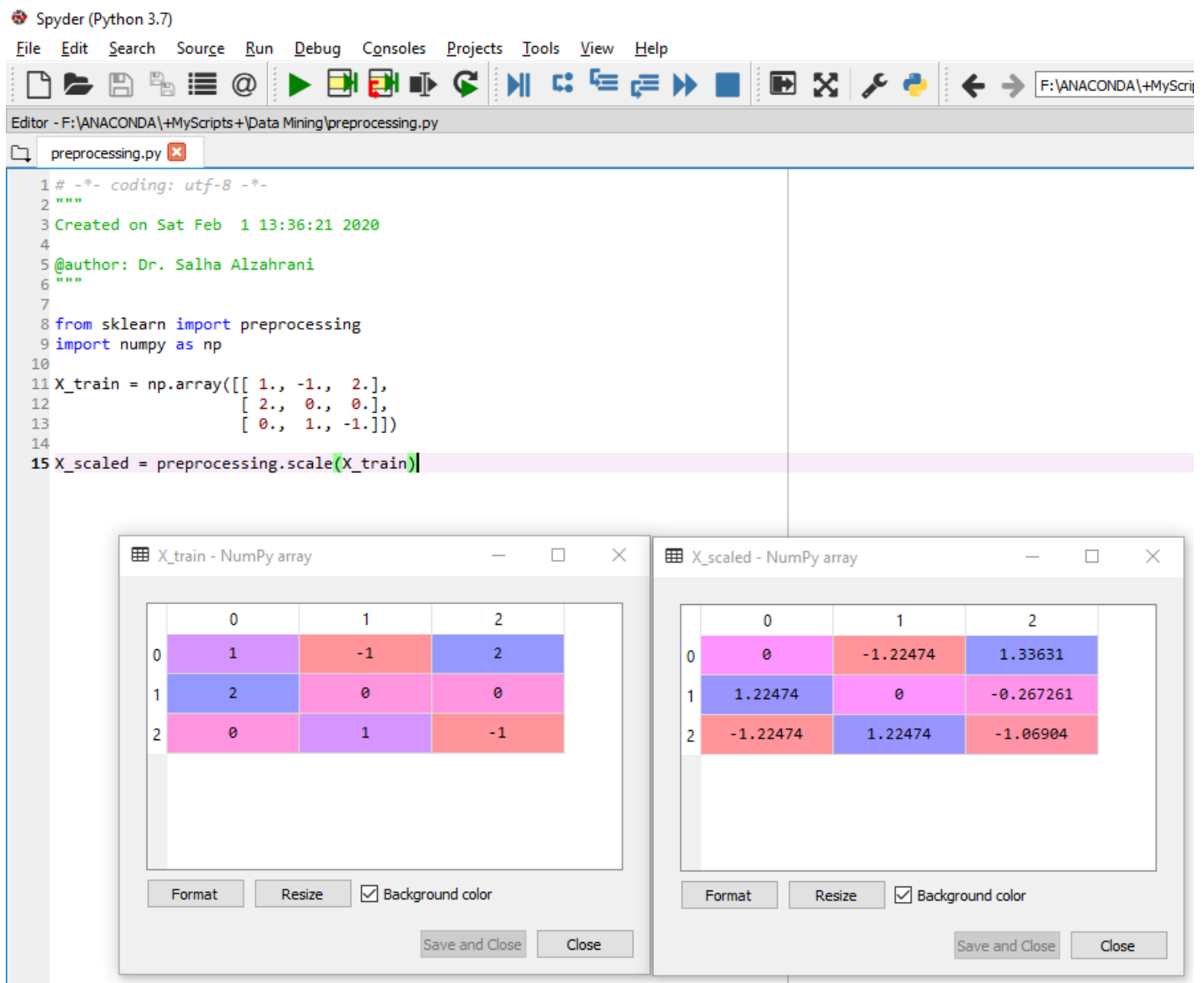
3.3 Preprocessing data

The `sklearn.preprocessing` package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.

3.3.1. Standardization (or mean removal)

Standardization of datasets is a **common requirement for many machine learning estimators** implemented in scikit-learn; they might behave badly if the individual features do not more or less look like standard normally distributed data: Gaussian with **zero mean and unit variance**.

In practice we often ignore the shape of the distribution and just transform the data to center it by removing the mean value of each feature, then scale it by dividing non-constant features by their standard deviation.



```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Feb 1 13:36:21 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7
8 from sklearn import preprocessing
9 import numpy as np
10
11 X_train = np.array([[ 1., -1.,  2.],
12                    [ 2.,  0.,  0.],
13                    [ 0.,  1., -1.]])
14
15 X_scaled = preprocessing.scale(X_train)
```

X_train - NumPy array

	0	1	2
0	1	-1	2
1	2	0	0
2	0	1	-1

X_scaled - NumPy array

	0	1	2
0	0	-1.22474	1.33631
1	1.22474	0	-0.267261
2	-1.22474	1.22474	-1.06904

Scaled data has zero mean and unit variance:

```
In [4]: X_scaled.mean(axis=0)
Out[4]: array([0., 0., 0.])

In [5]: X_scaled.std(axis=0)
Out[5]: array([1., 1., 1.])
```

3.3.2. Scaling to a range

An alternative standardization is scaling features to lie between a given minimum and maximum value, often between zero and one, or so that the maximum absolute value of each feature is scaled to unit size. This can be achieved using [MinMaxScaler](#) or [MaxAbsScaler](#), respectively.

The screenshot shows the Spyder Python IDE interface. The editor window displays a Python script for scaling data using `MinMaxScaler`. The code defines a training set `X_train` and applies the scaler to it. Below the editor, two windows display the resulting NumPy arrays.

Code in preprocessing_2.py:

```
1 #-*- coding: utf-8 -*-
2 """
3 Created on Sat Feb 1 13:50:10 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7 from sklearn import preprocessing
8 import numpy as np
9
10 #scaling to a range [0,1]
11 X_train = np.array([[ 1., -1., 2.],
12                    [ 2., 0., 0.],
13                    [ 0., 1., -1.]])
14
15 min_max_scaler = preprocessing.MinMaxScaler()
16
17 X_train_minmax = min_max_scaler.fit_transform(X_train)
18
19
```

X_train - NumPy array:

	0	1	2
0	1	-1	2
1	2	0	0
2	0	1	-1

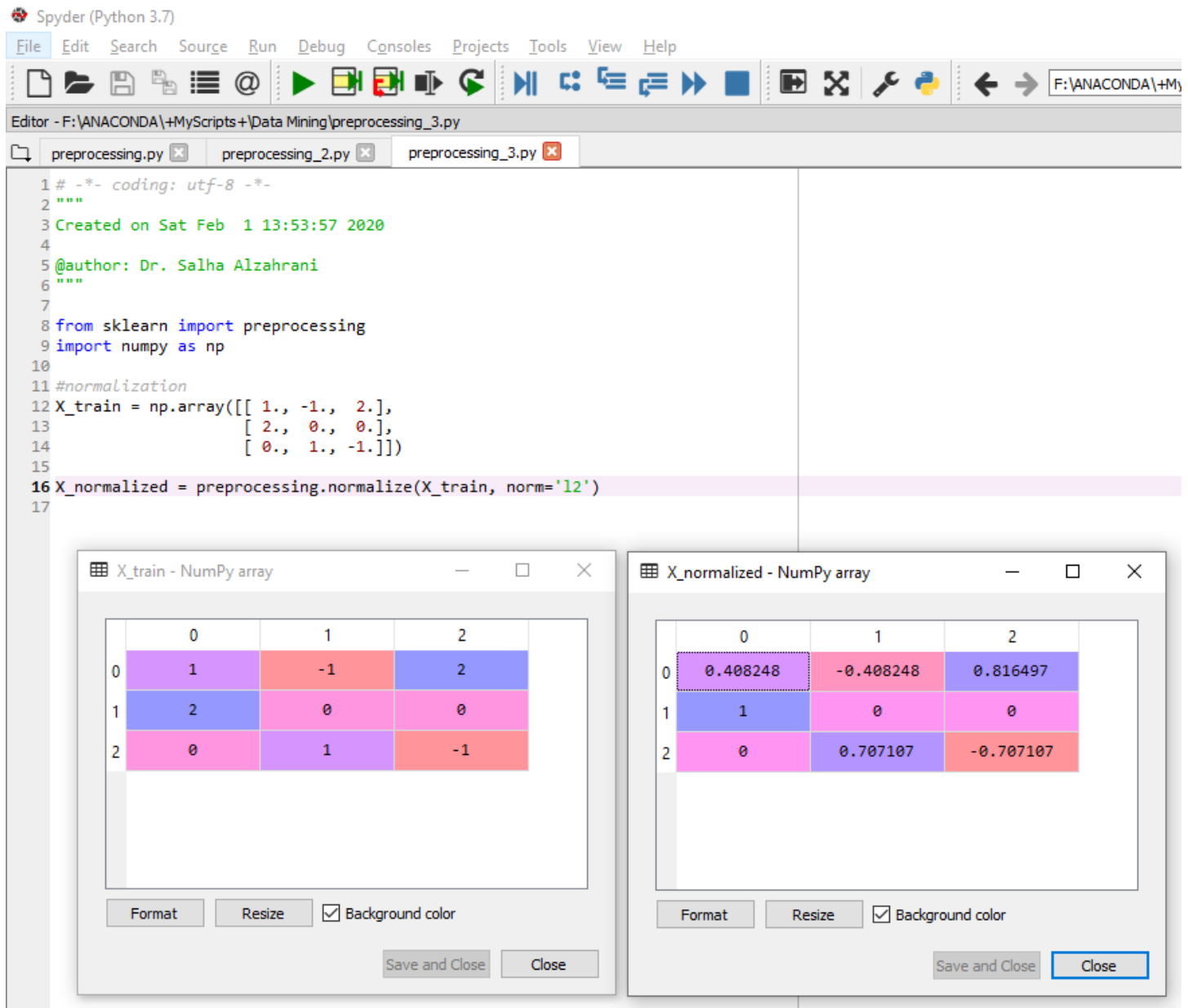
X_train_minmax - NumPy array:

	0	1	2
0	0.5	0	1
1	1	0.5	0.333333
2	0	1	0

3.3.3. Normalization

Normalization is the process of **scaling individual samples to have unit norm**. This process can be useful if you plan to use a quadratic form such as the dot-product or any other kernel to quantify the similarity of any pair of samples.

The function `normalize` provides a quick and easy way to perform this operation on a single array-like dataset, either using the l1 or l2 norms:



```
1 #-*- coding: utf-8 -*-
2 """
3 Created on Sat Feb 1 13:53:57 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7
8 from sklearn import preprocessing
9 import numpy as np
10
11 #normalization
12 X_train = np.array([[ 1., -1., 2.],
13                    [ 2., 0., 0.],
14                    [ 0., 1., -1.]])
15
16 X_normalized = preprocessing.normalize(X_train, norm='l2')
17
```

X_train - NumPy array

	0	1	2
0	1	-1	2
1	2	0	0
2	0	1	-1

X_normalized - NumPy array

	0	1	2
0	0.408248	-0.408248	0.816497
1	1	0	0
2	0	0.707107	-0.707107

Topic 4: Using SciKitLearn for Data Mining in Python

4.1 Loading an example dataset

scikit-learn comes with a few standard datasets, for instance the [iris](#) and [digits](#) datasets for classification and the [boston house prices dataset](#) for regression.

```
1 #-*- coding: utf-8 -*-
2 """
3 Created on Mon Feb 24 14:01:52 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7
8 from sklearn import datasets
9 iris = datasets.load_iris()
10
11 digits = datasets.load_digits()
```

iris - Dictionary (5 elements)

Key	Type	Size	Value
DESCR	str	1	Iris Plants Database =====
data	float64	(150, 4)	[[5.1 3.5 1.4 0.2] [4.9 3. 1.4 0.2]
feature_names	list	4	['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal ...
target	int32	(150,)	[[0] [0]
target_names	str320	(3,)	ndarray object of numpy module

Save and Close Close

data - NumPy array

	0	1	2	3
0	5.1	3.5	1.4	0.2
1	4.9	3	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4

Format Resize ☒ Background color

Save and Close Close

target - NumPy array

	0
0	0
1	0
2	0
3	0
4	0
5	0

Format Resize ☒ Background color

Save and Close Close


```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Feb 24 14:01:52 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7
8 from sklearn import datasets
9 iris = datasets.load_iris()
10
11 digits = datasets.load_digits()

```

digits - Dictionary (5 elements)

Key	Type	Size	Value
DESCR	str	1	Optical Recognition of Handwritten Digits Data ... ===== ...
data	float64	(1797, 64)	[[0. 0. 5. ... 0. 0. 0.] [0. 0. 0. ... 10. 0. 0.]
images	float64	(1797, 8, 8)	[[[0. 0. 5. ... 1. 0. 0.] [0. 0. 13. ... 15. 5. 0.]
target	int32	(1797,)	[[0] [1]
target_names	int32	(10,)	[0 1 2 3 4 5 6 7 8 9]

Save and Close

Close

data - NumPy array

	0	1	2	3	4	5
0	0	0	5	13	9	1
1	0	0	0	12	13	5
2	0	0	0	4	15	12
3	0	0	7	15	13	1
4	0	0	0	1	11	0

Format

Resize

☒ Background color

Save and Close

Close

target - NumPy array

	0
0	0
1	1
2	2
3	3
4	4
5	5

Format

Resize

4.2 Feature selection

The classes in the [sklearn.feature_selection](#) module can be used for feature selection/dimensionality reduction on sample sets, either to improve estimators' accuracy scores or to boost their performance on very high-dimensional datasets.

4.2.1. Removing features with low variance

[VarianceThreshold](#) is a simple baseline approach to feature selection. It removes all features whose variance doesn't meet some threshold. By default, it removes features that have the same value in all samples.

As an example, suppose that we have a dataset with boolean features, and we want to remove all features that are either one or zero (on or off) in probability (p) more than 80% of the samples, using this equation:

$$\text{Var}[X] = p(1 - p)$$

so we can select using the threshold $.8 * (1 - .8)$:

```
# -*- coding: utf-8 -*-
"""
Created on Mon Feb 24 13:47:21 2020

@author: Dr. Salha Alzahrani
"""

from sklearn.feature_selection import VarianceThreshold
X = [[0, 0, 1], [0, 1, 0], [1, 0, 0], [0, 1, 1], [0, 1, 0], [0, 1, 1]]
sel = VarianceThreshold(threshold=(.8 * (1 - .8)))
X_sel = sel.fit_transform(X)
```

X - List (6 elements)

Index	Type	Size	Value
0	list	3	[0, 0, 1]
1	list	3	[0, 1, 0]
2	list	3	[1, 0, 0]
3	list	3	[0, 1, 1]
4	list	3	[0, 1, 0]
5	list	3	[0, 1, 1]

Save and Close Close

X_sel - NumPy array

	0	1
0	0	1
1	1	0
2	0	0
3	1	1
4	1	0
5	1	1

Format Resize ☒ Background color

Save and Close Close

4.2.2. Univariate feature selection

Univariate feature selection works by selecting the best features based on univariate statistical tests. It can be seen as a preprocessing step to an estimator. Scikit-learn exposes feature selection routines as objects that implement the `transform` method:

- [`SelectKBest`](#) removes all but the highest k scoring features
- [`SelectPercentile`](#) removes all but a user-specified highest scoring percentage of features

For instance, we can perform a χ^2 test (`chi2`) to the samples to retrieve only the two best features as follows:

```
# -*- coding: utf-8 -*-
"""
Created on Mon Feb 24 13:57:35 2020

@author: Dr. Salha Alzahrani
"""

from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
X, y = load_iris(return_X_y=True)
X_new = SelectKBest(chi2, k=2).fit_transform(X, y)
```



Topic 5: Using SciKitLearn for Naïve Bayes Classification

5.1 Play Dataset

In this example, you can use the dummy dataset with three columns: weather, temperature, and play. The first two are features(weather, temperature) and the other is the label (i.e. class).

Whether	Temperature	Play
Sunny	Hot	No
Sunny	Hot	No
Overcast	Hot	Yes
Rainy	Mild	Yes
Rainy	Cool	Yes
Rainy	Cool	No
Overcast	Cool	Yes
Sunny	Mild	No
Sunny	Cool	Yes
Rainy	Mild	Yes
Sunny	Mild	Yes
Overcast	Mild	Yes
Overcast	Hot	Yes
Rainy	Mild	No

First, you need to convert these string labels into numbers. for example: 'Overcast', 'Rainy', 'Sunny' as 0, 1, 2. This is known as label encoding. Scikit-learn provides LabelEncoder library for encoding labels with a value between 0 and one less than the number of discrete classes.

Similarly, you can also encode temp and play columns.

Second, combine both the features (weather and temp) in a single variable (list of tuples).

Generating Model

Generate a model using naive bayes classifier in the following steps:

- Create naive bayes classifier
- Fit the dataset on classifier
- Perform prediction

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Feb 24 14:43:41 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7
8 # Assigning features and label variables
9 weather=['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny',
10 'Rainy','Sunny','Overcast','Overcast','Rainy']
11 temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mild','Hot','Mild']
12 play=['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']
13
14
15 # Import LabelEncoder
16 from sklearn import preprocessing
17 #creating LabelEncoder
18 le = preprocessing.LabelEncoder()
19 # Converting string labels into numbers.
20 weather_encoded=le.fit_transform(weather)
21 # Converting string labels into numbers
22 temp_encoded=le.fit_transform(temp)
23 label=le.fit_transform(play)
24 print("Weather:", weather_encoded)
25 print("Temp:", temp_encoded)
26 print("Play:", label)
27
28 #Combining weather and temp into single list of tuples
29 features=list(zip(weather_encoded,temp_encoded))
30 print("features:", features)
31
32 #####
33 #Import Gaussian Naive Bayes model
34 from sklearn.naive_bayes import GaussianNB
35
36 #Create a Gaussian Classifier
37 model = GaussianNB()
38
39 # Train the model using the training sets
40 model.fit(features,label)
41
42 #Predict Output
43 predicted= model.predict([[0,2]]) # 0:Overcast, 2:Mild
44 print("Predicted Value:", predicted)
45
46 #Predict Output
47 predicted= model.predict([[1,1]]) # 1:Rainy, 1:Hot
48 print("Predicted Value:", predicted)

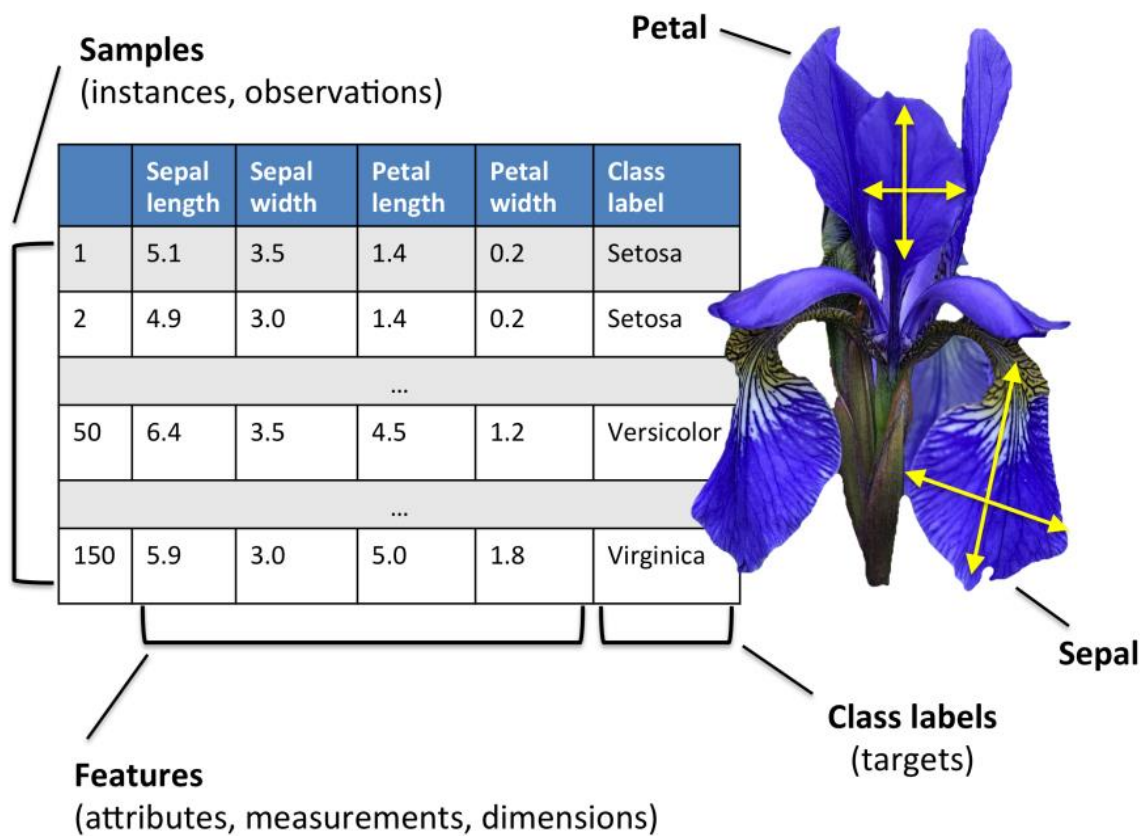
```

```

In [22]: runfile('F:/ANACONDA/MyScripts/Data Mining/naive_bayes_ex1.py', wdir='F:/ANACONDA/MyScripts/Data Mining')
Weather: [2 2 0 1 1 1 0 2 2 1 2 0 0 1]
Temp: [1 1 1 2 0 0 0 2 0 2 2 1 2]
Play: [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
features: [(2, 1), (2, 1), (0, 1), (1, 2), (1, 0), (1, 0), (0, 0), (2, 2), (2, 0), (1, 2), (2, 2), (0, 2), (0, 1), (1, 2)]
Predicted Value: [1]
Predicted Value: [1]

```

5.2 Irish Dataset



iris setosa



petal

sepal

iris versicolor



petal

sepal

iris virginica



petal

sepal

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Feb 25 04:50:04 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7
8 from sklearn.datasets import load_iris
9 from sklearn.model_selection import train_test_split
10 from sklearn.naive_bayes import GaussianNB
11
12 #Load features X, and class y
13 X, y = load_iris(return_X_y=True)
14
15 #Split into train and test
16 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
17
18 #create Naive Bayes Classifier
19 model = GaussianNB()
20
21 #Train the model using the training sets
22 model.fit(X_train,y_train)
23
24 #Make predictions for the test dataset
25 y_pred = model.predict(X_test)
26
27 print("Number of mislabeled points out of a total %d points : %d"
28       % (X_test.shape[0], (y_test != y_pred).sum()))

```

```

In [25]: runfile('F:/ANACONDA/+MyScripts+/Data Mining/naive_bayes_ex2.py', wdir='F:/ANACONDA/+MyScripts+/Data Mining')
Number of mislabeled points out of a total 75 points : 4

```

Note:

GaussianNB implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

The parameters σ_y and μ_y are estimated using maximum likelihood.

Topic 6: Using SciKitLearn for Nearest Neighbour Classification

6.1 Play Dataset

Using k=3

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Feb 25 05:11:18 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7
8
9 # Assigning features and label variables
10 weather=['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny',
11 'Rainy','Sunny','Overcast','Overcast','Rainy']
12 temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mild','Hot','Mild']
13 play=['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']
14
15
16 # Import LabelEncoder
17 from sklearn import preprocessing
18 #creating LabelEncoder
19 le = preprocessing.LabelEncoder()
20 # Converting string labels into numbers.
21 weather_encoded=le.fit_transform(weather)
22 # Converting string labels into numbers
23 temp_encoded=le.fit_transform(temp)
24 label=le.fit_transform(play)
25 print("Weather:", weather_encoded)
26 print("Temp:", temp_encoded)
27 print("Play:", label)
28
29 #Combining weather and temp into single list of tuples
30 features=list(zip(weather_encoded,temp_encoded))
31 print("features:", features)
32
33 #####
34 # import Nearest Neighbor Classifier
35 from sklearn.neighbors import KNeighborsClassifier
36
37 model = KNeighborsClassifier(n_neighbors=3)
38
39 # Train the model using the training sets
40 model.fit(features,label)
41
42 #Predict Output
43 predicted= model.predict([[0,2]]) # 0:Overcast, 2:Mild
44 print("Predicted Value:", predicted)
45
46 #Predict Output
47 predicted= model.predict([[1,1]]) # 1:Rainy, 1:Hot
48 print("Predicted Value:", predicted)

In [30]: runfile('F:/ANACONDA/MyScripts/Data Mining/nearest_neighbour_ex1.py', wdir='F:/ANACONDA/MyScripts/Data Mining')
Weather: [2 2 0 1 1 1 0 2 2 1 2 0 0 1]
Temp: [1 1 1 2 0 0 0 2 0 2 2 1 2]
Play: [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
features: [(2, 1), (2, 1), (0, 1), (1, 2), (1, 0), (1, 0), (0, 0), (2, 2), (2, 0), (1, 2), (2, 2), (0, 2), (0, 1), (1, 2)]
Predicted Value: [1]
Predicted Value: [0]
```


Using k=7

```
33 #####
34 # import Nearest Neighbor Classifier
35 from sklearn.neighbors import KNeighborsClassifier
36
37 model = KNeighborsClassifier(n_neighbors=7)
38
39 # Train the model using the training sets
40 model.fit(features,label)
41
42 #Predict Output
43 predicted= model.predict([[0,2]]) # 0:Overcast, 2:Mild
44 print("Predicted Value:", predicted)
45
46 #Predict Output
47 predicted= model.predict([[1,1]]) # 1:Rainy, 1:Hot
48 print("Predicted Value:", predicted)
49
```

```
In [31]: runfile('F:/ANACONDA/+MyScripts+/Data Mining/nearest_neighbour_ex1.py', wdir='F:/ANACONDA/+MyScripts+/Data Mining')
Weather: [2 2 0 1 1 1 0 2 2 1 2 0 0 1]
Temp: [1 1 1 2 0 0 0 2 0 2 2 2 1 2]
Play: [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
features: [(2, 1), (2, 1), (0, 1), (1, 2), (1, 0), (1, 0), (0, 0), (2, 2), (2, 0), (1, 2), (2, 2), (0, 2), (0, 1), (1, 2)]
Predicted Value: [1]
Predicted Value: [0]
```

6.2 Irish Dataset

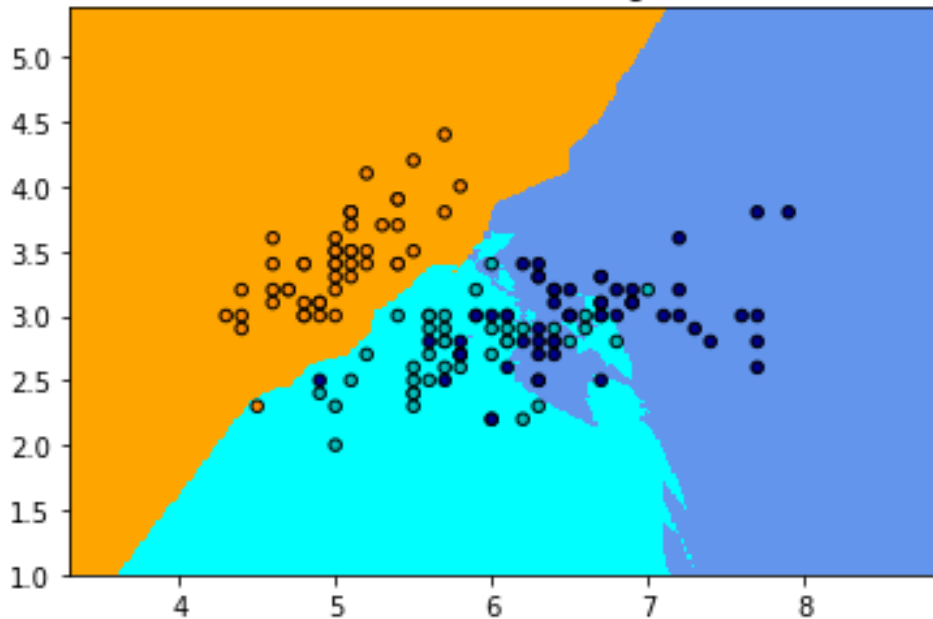
Sample usage of Nearest Neighbors classification for Irish Dataset. It will plot the decision boundaries for each class.

Using k=7

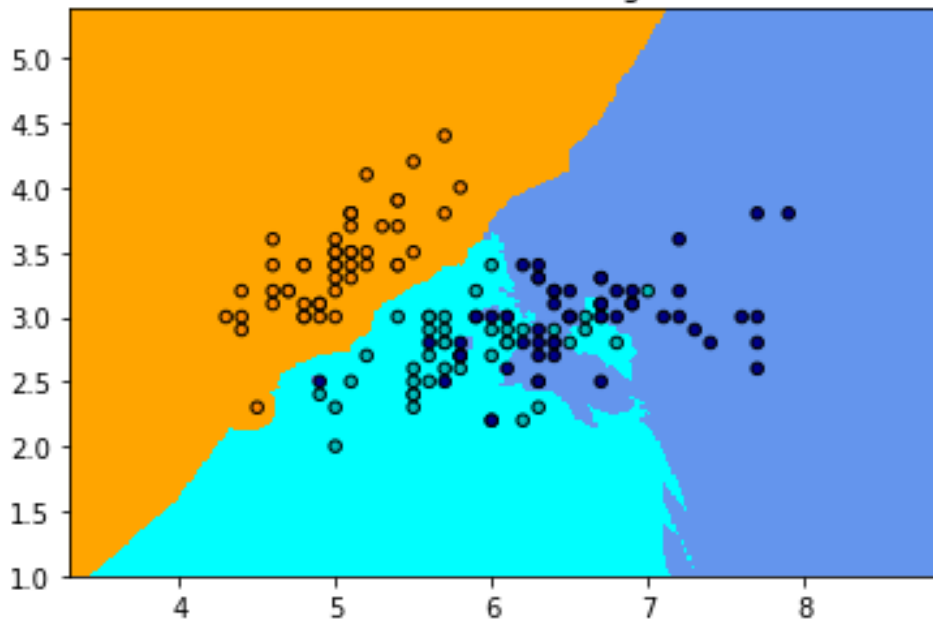
```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Feb 25 05:03:07 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7
8 import numpy as np
9 import matplotlib.pyplot as plt
10 from matplotlib.colors import ListedColormap
11 from sklearn import neighbors, datasets
12
13 n_neighbors = 7
14
15 # import some data to play with
16 iris = datasets.load_iris()
17
18 # we only take the first two features. We could avoid this ugly
19 # slicing by using a two-dim dataset
20 X = iris.data[:, :2]
21 y = iris.target
22
23 h = .02 # step size in the mesh
24
25 # Create color maps
26 cmap_light = ListedColormap(['orange', 'cyan', 'cornflowerblue'])
27 cmap_bold = ListedColormap(['darkorange', 'c', 'darkblue'])
28
29 for weights in ['uniform', 'distance']:
30     # we create an instance of Neighbours Classifier and fit the data.
31     clf = neighbors.KNeighborsClassifier(n_neighbors, weights=weights)
32     clf.fit(X, y)
33
34     # Plot the decision boundary. For that, we will assign a color to each
35     # point in the mesh [x_min, x_max]x[y_min, y_max].
36     x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
37     y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
38     xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
39                          np.arange(y_min, y_max, h))
40     Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
41
42     # Put the result into a color plot
43     Z = Z.reshape(xx.shape)
44     plt.figure()
45     plt.pcolormesh(xx, yy, Z, cmap=cmap_light)
46
47     # Plot also the training points
48     plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold,
49                edgecolor='k', s=20)
50     plt.xlim(xx.min(), xx.max())
51     plt.ylim(yy.min(), yy.max())
52     plt.title("3-Class classification (k = %i, weights = '%s')\n"
53              % (n_neighbors, weights))
54
55 plt.show()
```

```
In [27]: runfile('F:/ANACONDA/+MyScripts+/Data Mining/nearest_neighbour_ex1.py')
```

3-Class classification (k = 7, weights = 'uniform')



3-Class classification (k = 7, weights = 'distance')

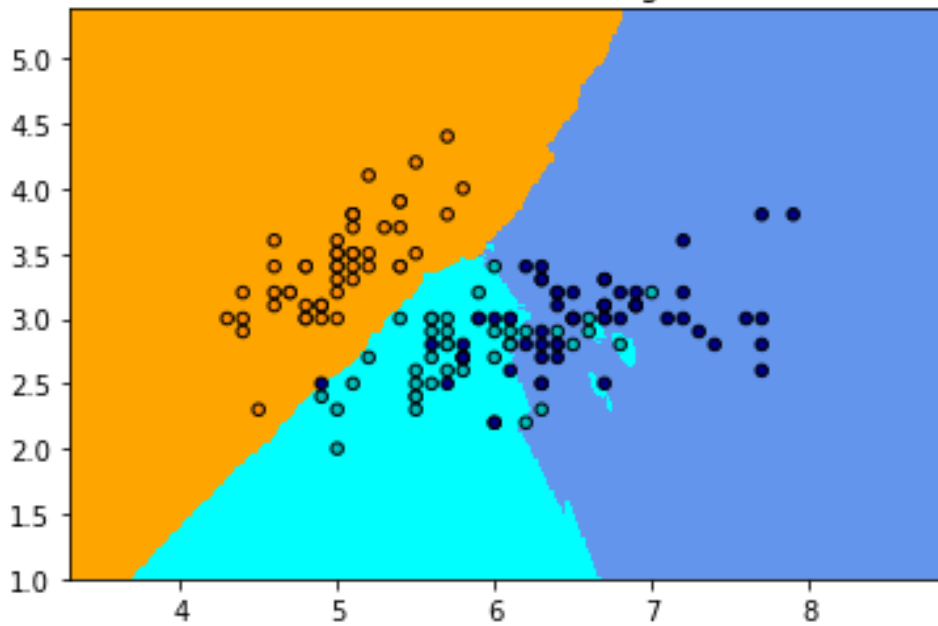


Using k=15

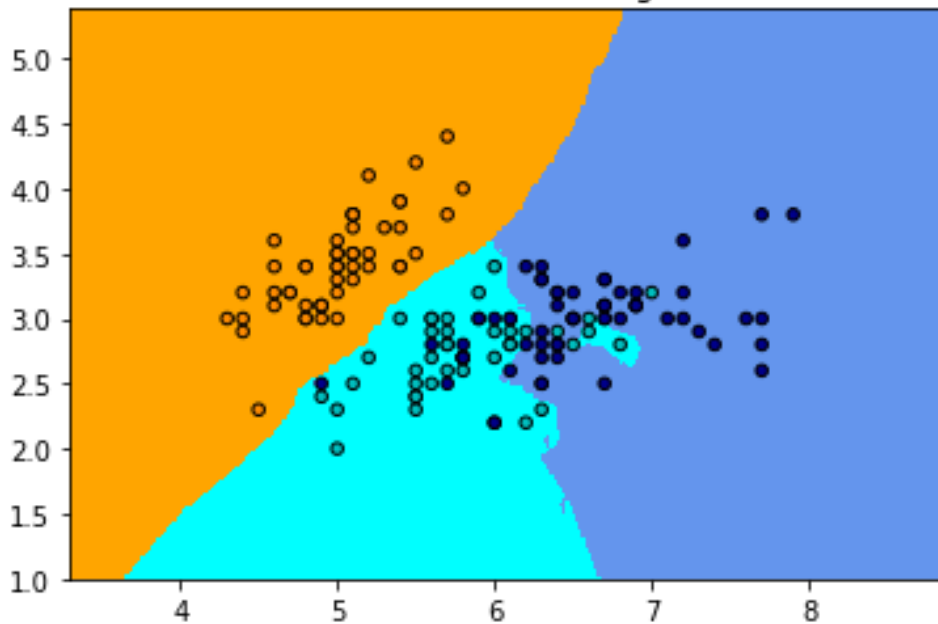
```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Feb 25 05:03:07 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7
8 import numpy as np
9 import matplotlib.pyplot as plt
10 from matplotlib.colors import ListedColormap
11 from sklearn import neighbors, datasets
12
13 n_neighbors = 15
14
15 # import some data to play with
16 iris = datasets.load_iris()
17
18 # we only take the first two features. We could avoid this ugly
19 # slicing by using a two-dim dataset
20 X = iris.data[:, :2]
21 y = iris.target
22
23 h = .02 # step size in the mesh
24
25 # Create color maps
26 cmap_light = ListedColormap(['orange', 'cyan', 'cornflowerblue'])
27 cmap_bold = ListedColormap(['darkorange', 'c', 'darkblue'])
28
29 for weights in ['uniform', 'distance']:
30     # we create an instance of Neighbours Classifier and fit the data.
31     clf = neighbors.KNeighborsClassifier(n_neighbors, weights=weights)
32     clf.fit(X, y)
33
34     # Plot the decision boundary. For that, we will assign a color to each
35     # point in the mesh [x_min, x_max]x[y_min, y_max].
36     x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
37     y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
38     xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
39                          np.arange(y_min, y_max, h))
40     Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
41
42     # Put the result into a color plot
43     Z = Z.reshape(xx.shape)
44     plt.figure()
45     plt.pcolormesh(xx, yy, Z, cmap=cmap_light)
46
47     # Plot also the training points
48     plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold,
49                edgecolor='k', s=20)
50     plt.xlim(xx.min(), xx.max())
51     plt.ylim(yy.min(), yy.max())
52     plt.title("3-Class classification (k = %i, weights = '%s')",
53              % (n_neighbors, weights))
54
55 plt.show()
```

```
In [28]: runfile('F:/ANACONDA/+MyScripts+/Data Mining/nearest_neighbour_ex1.py').
```

3-Class classification (k = 15, weights = 'uniform')



3-Class classification (k = 15, weights = 'distance')



6.4 Other Numerical Datasets

6.4.1. Example of numerical array

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Feb 25 05:21:15 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7
8 from sklearn import datasets
9 from sklearn import metrics
10 from sklearn.neighbors import KNeighborsClassifier
11 import numpy as np
12
13 # dataset
14 X = np.array([[ -1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
15 target = [0, 0, 0, 1, 1, 1]
16
17 # fit a k-nearest neighbor model to the data
18 K = 3
19 model = KNeighborsClassifier(n_neighbors = K)
20 model.fit(X, target)
21 print(model)
22
23 # make predictions
24 print( '(-2,-2) is class'),
25 print( model.predict([[-2,-2]]) )
26
27 print( '(1,3) is class'),
28 print( model.predict([[1,3]]) )
```

```
In [34]: runfile('F:/ANACONDA/+MyScripts+/Data Mining/nearest_neighbour_ex3.py', wdir='F:/ANACONDA/+MyScripts+/Data Mining')
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=1, n_neighbors=3, p=2,
                    weights='uniform')
(-2,-2) is class
[0]
(1,3) is class
[1]
```

6.4.2. Example of prices

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Feb 25 05:24:03 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7
8 from sklearn.neighbors import KNeighborsRegressor
9
10 # dataset (X=m^2, y=rental price)
11 X = [[40], [45], [60], [70]]
12 y = [1000, 1200, 2000, 2500]
13
14 # fit
15 neigh = KNeighborsRegressor(n_neighbors=2)
16 neigh.fit(X, y)
17
18 # predict
19 print('Monthly Rental Price for 65m^2 in $'),
20 print(neigh.predict([[65]]))
```

```
In [35]: runfile('F:/ANACONDA/MyScripts/Data Mining/nearest_neighbour_ex4.py', wdir='F:/ANACONDA/MyScripts/Data Mining')
Monthly Rental Price for 65m^2 in $
[2250.]
```

Topic 7: Using SciKitLearn for Decision Trees Classification

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Some advantages of decision trees are:

- Simple to understand and to interpret. Trees can be visualised.
- Requires little data preparation.
- Able to handle both numerical and categorical data.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

The disadvantages of decision trees include:

- Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting.
- The problem of learning an optimal decision tree is known to be NP-complete.
- Decision tree learners create biased trees if some classes dominate. **7.1 Irish Dataset**

7.1 Decision Tree Classifier for a Simple Numerical Dataset

[DecisionTreeClassifier](#) is a class capable of performing multi-class classification on a dataset. As with other classifiers, [DecisionTreeClassifier](#) takes as input two arrays:

- an array X, sparse or dense, of size [n_samples, n_features] holding the training samples, and
- an array Y of integer values, size [n_samples], holding the class labels for the training samples:

Note that in the numerical example below we have two classes [0,1], so we call this binary classification.

decision_tree_ex1.py

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Fri Mar 20 12:40:25 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7 #import decition tree from sklearn
8 from sklearn import tree
9
10 #Suppose that your data is very simple
11 X = [[0, 0], [1, 1]] #features
12 Y = [0, 1]          #labels or target
13
14 #Build your classifier
15 classifier = tree.DecisionTreeClassifier()
16 #Train your classifier
17 classifier = classifier.fit(X, Y)
18 #Finally use your classifier to predict unclassified / unseen instances
19 print(classifier.predict([[0., 0.5]]))
20 print(classifier.predict([[1., 1.5]]))
21
22

```

X - List (2 elements)

Index	Type	Size	Value
0	list	2	[0, 0]
1	list	2	[1, 1]

Save and Close
Close

Y - List (2 elements)

Index	Type	Size	Value
0	int	1	0
1	int	1	1

Save and Close
Close

Output:

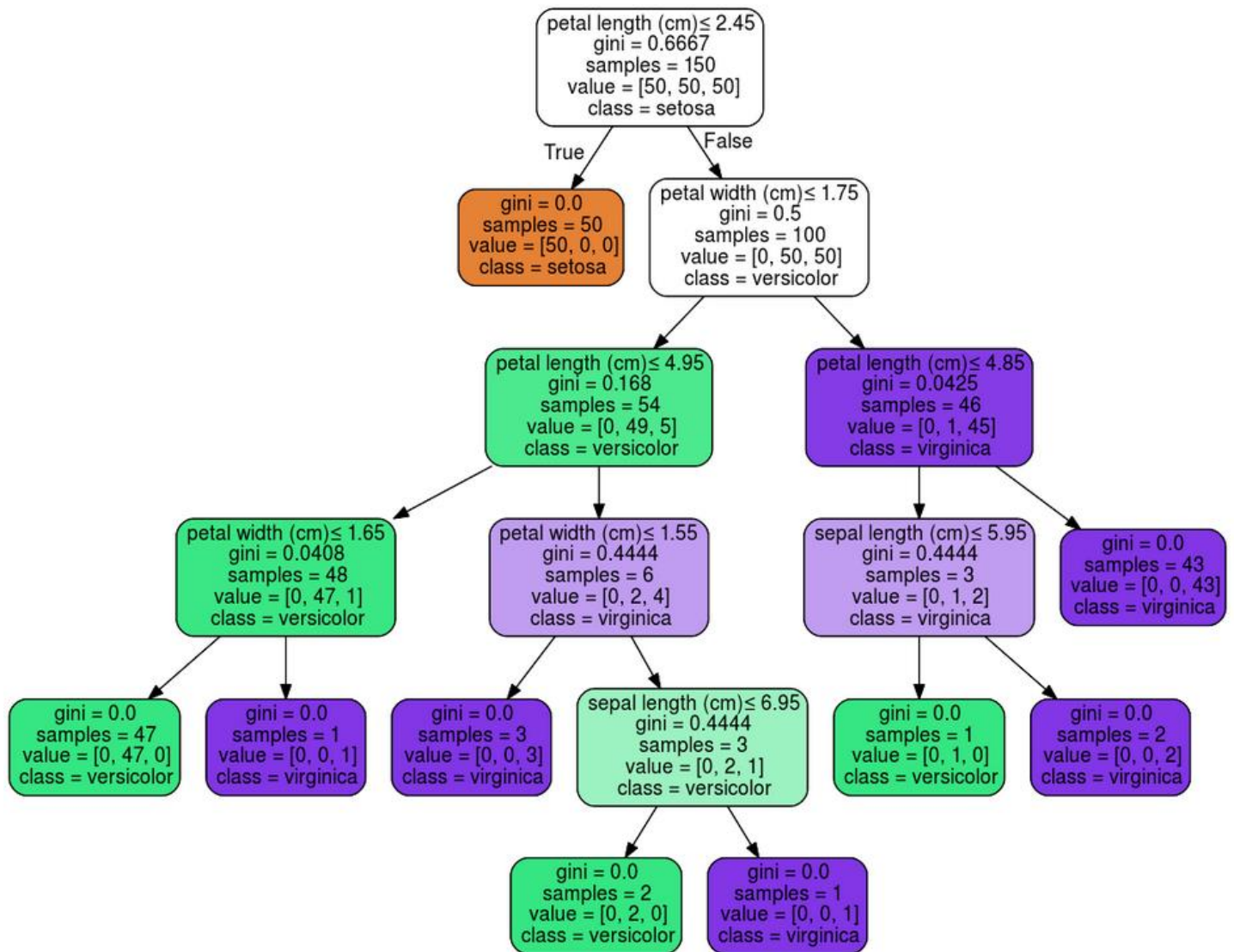
```

In [43]: runfile('F:/ANACONDA/MyScripts/Data Mining/decision_tree_ex1.py', wdir='F:/ANACONDA/MyScripts/Data Mining')
[0]
[1]

```

7.2 Decision Tree Classifier for Irish Dataset

[DecisionTreeClassifier](#) is capable for multiclass (where the labels are [0, ..., K-1]) classification. Example of this is classifying Irish data.



decision_tree_ex2.py

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Fri Mar 20 12:54:21 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7 #Load dataset
8 from sklearn.datasets import load_iris
9 #import decision tree from sklearn
10 from sklearn import tree
11
12 #Load features X and targets Y
13 X, Y = load_iris(return_X_y=True)
14
15 #Build and train your classifier
16 classifier = tree.DecisionTreeClassifier()
17 classifier = classifier.fit(X, Y)
18
19 #Predict unclassified instance
20 print(classifier.predict([[5, 5, 5, 2]]))
21
22

```

X - NumPy array

	0	1	2	3
46	5.1	3.8	1.6	0.2
47	4.6	3.2	1.4	0.2
48	5.3	3.7	1.5	0.2
49	5	3.3	1.4	0.2
50	7	3.2	4.7	1.4
51	6.4	3.2	4.5	1.5
52	6.9	3.1	4.9	1.5

Y - NumPy array

	0
46	0
47	0
48	0
49	0
50	1
51	1
52	1

Output:

```

In [44]: runfile('F:/ANACONDA/+MyScripts+/Data Mining/decision_tree_ex2.py', wdir='F:/ANACONDA/+MyScripts+/Data Mining')
[2]

```


Training dataset:

X_train - NumPy array					y_train - NumPy array				
	0	1	2	3		0			
0	0.444444	0.416667	0.534483	0.583333		1			
1	0.416667	0.25	0.5	0.458333		1			
2	0.694444	0.416667	0.758621	0.833333		2			
3	0.111111	0.5	0.0344828	0.0416667		0			
4	0.722222	0.458333	0.689655	0.916667		2			
5	0.194444	0.625	0.0862069	0.208333		0			
6	0.305556	0.708333	0.0689655	0.0416667		0			
7	0.194444	0	0.413793	0.375		1			
8	0.611111	0.416667	0.758621	0.708333		2			
9	0.666667	0.541667	0.793103	1		2			
10	0.472222	0.0833333	0.672414	0.583333		2			
11	0.666667	0.208333	0.810345	0.708333		2			
12	0.361111	0.208333	0.482759	0.416667		1			

Test dataset:

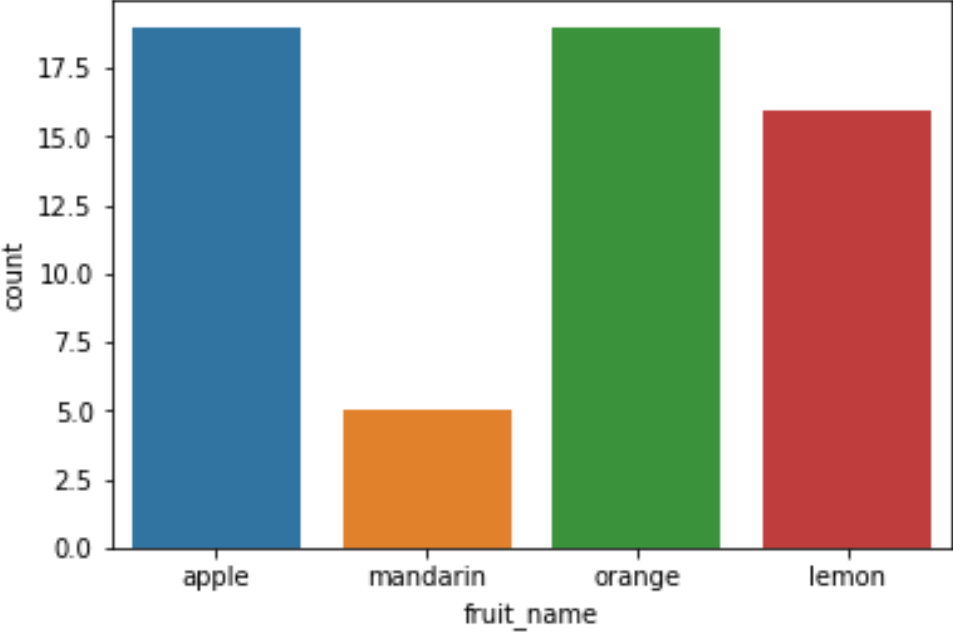
X_test - NumPy array					y_test - NumPy array				
	0	1	2	3		0			
0	0.416667	0.333333	0.689655	0.958333		2			
1	0.472222	0.0833333	0.5	0.375		1			
2	0.333333	0.916667	0.0517241	0.0416667		0			
3	0.833333	0.375	0.896552	0.708333		2			
4	0.194444	0.583333	0.0689655	0.0416667		0			
5	0.555556	0.541667	0.844828	1		2			
6	0.194444	0.625	0.0344828	0.0833333		0			
7	0.666667	0.458333	0.62069	0.583333		1			
8	0.694444	0.333333	0.637931	0.541667		1			
9	0.5	0.333333	0.5	0.5		1			
10	0.5	0.25	0.775862	0.541667		2			
11	0.583333	0.5	0.586207	0.583333		1			
12	0.5	0.333333	0.62069	0.458333		1			

8.2 Comparison between Naïve Bayes, Nearest Neighbour and Decision Tree Classifiers

```
classfires_comparison.py* x
1 # -*- coding: utf-8 -*-
2 """
3 Created on Fri Mar 20 13:58:40 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7
8 import pandas as pd
9 import matplotlib.pyplot as plt
10 fruits = pd.read_table('fruit_data_with_colors.txt')
11 fruits.head()
12 print(fruits.shape)
13 print(fruits['fruit_name'].unique())
14 print(fruits.groupby('fruit_name').size())
15
16 #Visulaization
17 import seaborn as sns
18 sns.countplot(fruits['fruit_name'],label="Count")
19 plt.show()
20
21 #Create Training and Test Sets and Apply Scaling
22 from sklearn.model_selection import train_test_split
23 X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
24 #Preprocessing
25 from sklearn.preprocessing import MinMaxScaler
26 scaler = MinMaxScaler()
27 X_train = scaler.fit_transform(X_train)
28 X_test = scaler.transform(X_test)
29
30 #Gaussian Naive Bayes
31 from sklearn.naive_bayes import GaussianNB
32 gnb = GaussianNB()
33 gnb.fit(X_train, y_train)
34 print('Accuracy of GNB classifier on training set: {:.2f}'
35       .format(gnb.score(X_train, y_train)))
36 print('Accuracy of GNB classifier on test set: {:.2f}'
37       .format(gnb.score(X_test, y_test)))
38
39 #K-Nearest Neighbors
40 from sklearn.neighbors import KNeighborsClassifier
41 knn = KNeighborsClassifier()
42 knn.fit(X_train, y_train)
43 print('Accuracy of K-NN classifier on training set: {:.2f}'
44       .format(knn.score(X_train, y_train)))
45 print('Accuracy of K-NN classifier on test set: {:.2f}'
46       .format(knn.score(X_test, y_test)))
47
48 #Decision Tree
49 from sklearn.tree import DecisionTreeClassifier
50 clf = DecisionTreeClassifier().fit(X_train, y_train)
51 print('Accuracy of Decision Tree classifier on training set: {:.2f}'
52       .format(clf.score(X_train, y_train)))
53 print('Accuracy of Decision Tree classifier on test set: {:.2f}'
54       .format(clf.score(X_test, y_test)))
55
```

Output:

```
In [59]: runfile('F:/ANACONDA/+MyScripts+/Data Mining/classifires_comparison.py',
(59, 7)
['apple' 'mandarin' 'orange' 'lemon']
fruit_name
apple      19
lemon      16
mandarin    5
orange     19
dtype: int64
```



fruit_name	count
apple	19
mandarin	5
orange	19
lemon	16

```
Accuracy of GNB classifier on training set: 0.95
Accuracy of GNB classifier on test set: 1.00
Accuracy of K-NN classifier on training set: 0.96
Accuracy of K-NN classifier on test set: 0.97
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.97
```


Topic 9: Using SciKitLearn for Clustering

Clustering of unlabeled data can be performed with the module [sklearn.cluster](#).

9.1. Overview of clustering methods

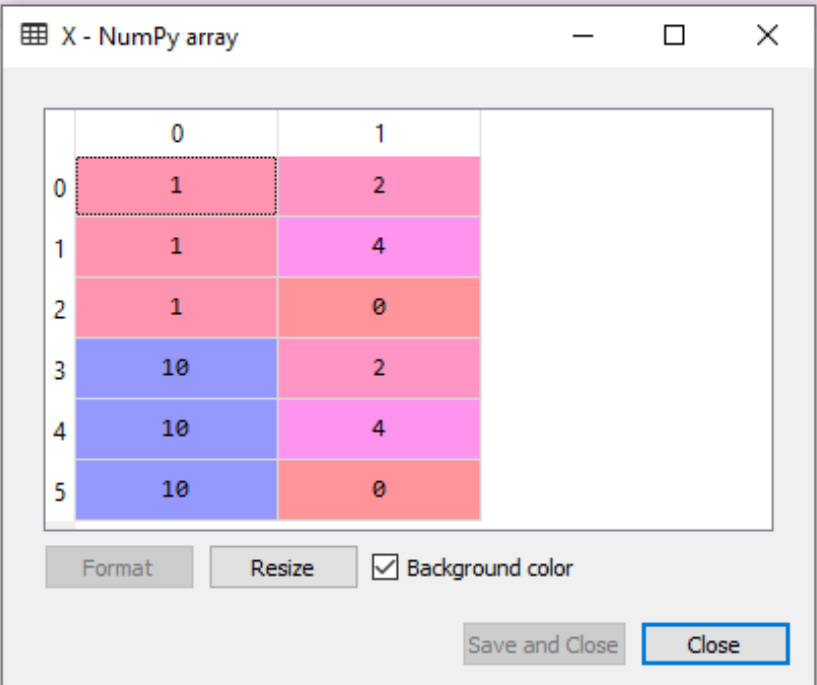
There are many clustering algorithms provided in scikit-learn. We will focus only on one of them: k-Means.

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large <code>n_samples</code> , medium <code>n_clusters</code> with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with <code>n_samples</code>	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with <code>n_samples</code>	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium <code>n_samples</code> , small <code>n_clusters</code>	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large <code>n_samples</code> and <code>n_clusters</code>	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large <code>n_samples</code> and <code>n_clusters</code>	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large <code>n_samples</code> , medium <code>n_clusters</code>	Non-flat geometry, uneven cluster sizes	Distances between nearest points
OPTICS	minimum cluster membership	Very large <code>n_samples</code> , large <code>n_clusters</code>	Non-flat geometry, uneven cluster sizes, variable cluster density	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large <code>n_clusters</code> and <code>n_samples</code>	Large dataset, outlier removal, data reduction.	Euclidean distance between points

9.2. K-Means Clustering of Numerical Dataset

The KMeans algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares. This algorithm requires the number of clusters to be specified.

```
clustering_ex1.py x
1 # -*- coding: utf-8 -*-
2 """
3 Created on Fri Mar 20 14:49:14 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7 #import K-Means Clustering algorithm from SciKit-Learn
8 from sklearn.cluster import KMeans
9 #Numbers package
10 import numpy as np
11
12 #Define the dataset as an array of points
13 X = np.array([[1, 2], [1, 4], [1, 0],[10, 2], [10, 4], [10, 0]])
14
15 #Build and Training your kmeans clustering
16 kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
17
18 #Results
19 #Labels of data points after clustering
20 print("Clusters' Labels: ")
21 print(kmeans.labels_)
22
23 #Predict the cluster of unclassified/unseen instance
24 print("Predict cluster of (0,0) and (12,3): ")
25 print(kmeans.predict([[0, 0], [12, 3]]))
26
27 #Show the clustering centers
28 print("Clusters' Centers: ")
29 print(kmeans.cluster_centers_)
30
```



	0	1
0	1	2
1	1	4
2	1	0
3	10	2
4	10	4
5	10	0

Output:

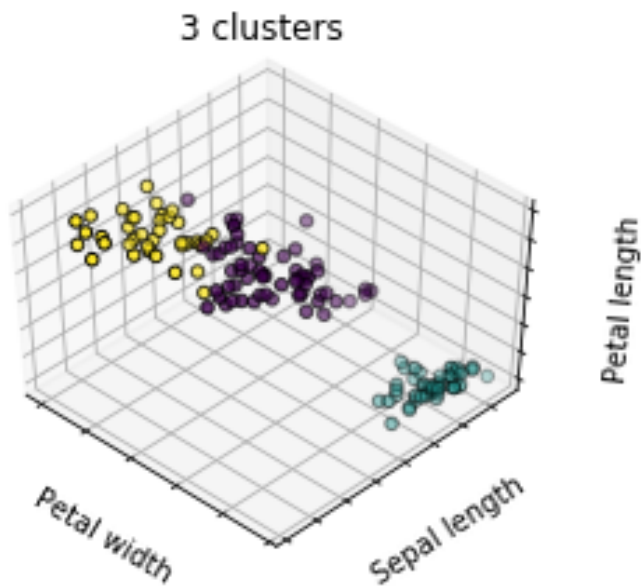
```
In [64]: runfile('F:/ANACONDA/+MyScripts+/Data Mining/clustering_ex1.py'.
Clusters' Labels:
[1 1 1 0 0 0]
Precit cluster of (0,0) and (12,3):
[1 0]
Clusters' Centers:
[[10.  2.]
 [ 1.  2.]]
```

9.2. K-Means Clustering of Irish Dataset and Visualization

```
clustering_ex2.py
1 # -*- coding: utf-8 -*-
2 """
3 Created on Fri Mar 20 14:59:41 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7 #Libs for Visualziation
8 import numpy as np
9 import matplotlib.pyplot as plt
10 # Though the following import is not directly being used, it is required
11 # for 3D projection to work
12 from mpl_toolkits.mplot3d import Axes3D
13 np.random.seed(5)
14
15 #Libs for Clustering
16 from sklearn.cluster import KMeans
17 from sklearn import datasets
18
19 #Load Dataset
20 iris = datasets.load_iris()
21 X = iris.data
22 y = iris.target
23
24 #Build and Train your kmeans clustering
25 estimator = kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
26 #Labels of data points after clsutering
27 labels = estimator.labels_
28
29 #This part ofr 3D visulzation of resulted clusters
30 fignum = 1
31 title = '3 clusters'
32 fig = plt.figure(fignum, figsize=(4, 3))
33
34 ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)
35 ax.scatter(X[:, 3], X[:, 0], X[:, 2], c=labels.astype(np.float), edgecolor='k')
36 ax.w_xaxis.set_ticklabels([])
37 ax.w_yaxis.set_ticklabels([])
38 ax.w_zaxis.set_ticklabels([])
39 ax.set_xlabel('Petal width')
40 ax.set_ylabel('Sepal length')
41 ax.set_zlabel('Petal length')
42 ax.set_title(title)
43 ax.dist = 12
44
45 fig.show()
```

Output:

```
In [73]: runfile('F:/ANACONDA/+MyScripts+/Data Mining/clustering_ex2.py',  
F:\ANACONDA\lib\site-packages\matplotlib\figure.py:457: UserWarning: matp  
cannot show the figure  
"matplotlib is currently using a non-GUI backend, "
```



9.3. K-Means Clustering of Irish Dataset and Visualization using 3 clusters, 8 clusters, and 3 with random/bad initialization

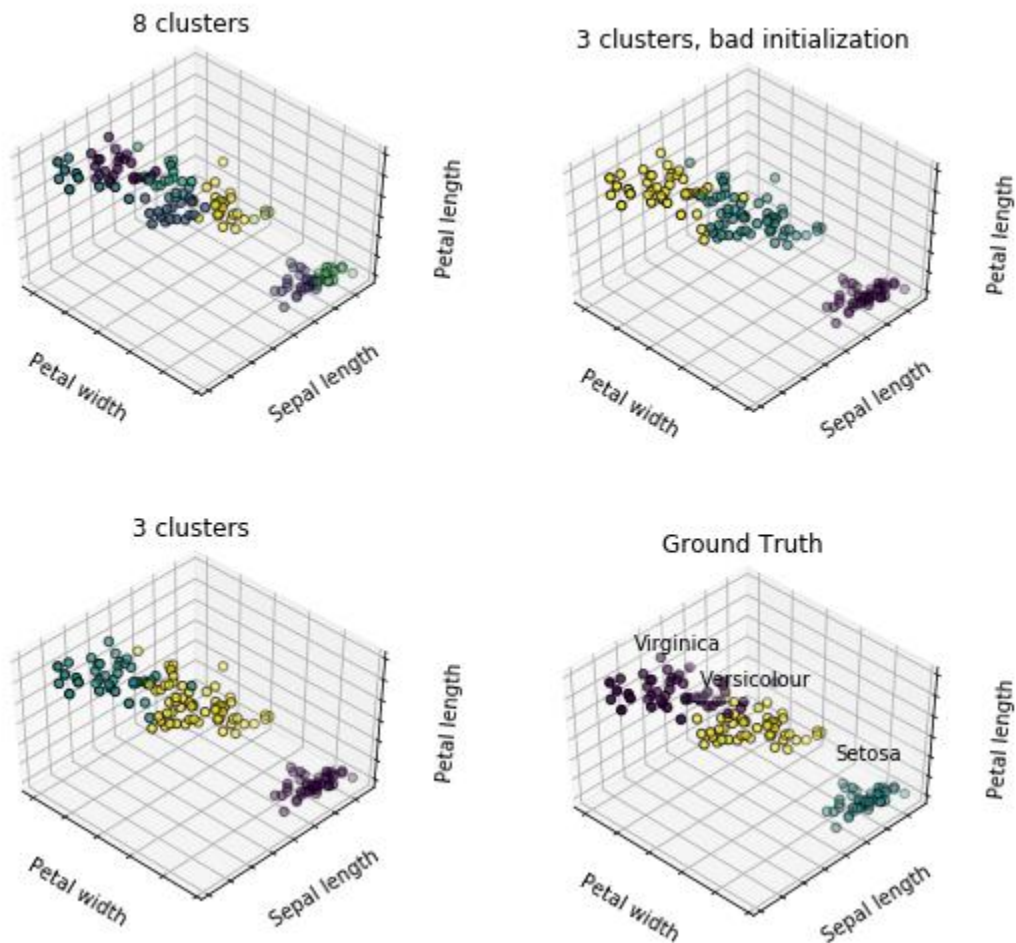
```
clustering_ex3.py
1 # -*- coding: utf-8 -*-
2 """
3 Created on Fri Mar 20 14:58:06 2020
4
5 @author: Dr. Salha Alzahrani
6 """
7 import numpy as np
8 import matplotlib.pyplot as plt
9 from mpl_toolkits.mplot3d import Axes3D
10 np.random.seed(5)
11
12 from sklearn.cluster import KMeans
13 from sklearn import datasets
14 iris = datasets.load_iris()
15 X = iris.data
16 y = iris.target
17
18 estimators = [('k_means_iris_8', KMeans(n_clusters=8)),
19               ('k_means_iris_3', KMeans(n_clusters=3)),
20               ('k_means_iris_bad_init', KMeans(n_clusters=3, n_init=1, init='random'))]
21 fignum = 1
22 titles = ['8 clusters', '3 clusters', '3 clusters, bad initialization']
23 for name, est in estimators:
24     fig = plt.figure(fignum, figsize=(4, 3))
25     ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)
26     est.fit(X)
27     labels = est.labels_
28     ax.scatter(X[:, 3], X[:, 0], X[:, 2], c=labels.astype(np.float), edgecolor='k')
29     ax.w_xaxis.set_ticklabels([])
30     ax.w_yaxis.set_ticklabels([])
31     ax.w_zaxis.set_ticklabels([])
32     ax.set_xlabel('Petal width')
33     ax.set_ylabel('Sepal length')
34     ax.set_zlabel('Petal length')
35     ax.set_title(titles[fignum - 1])
36     ax.dist = 12
37     fignum = fignum + 1
```

```

38
39 # Plot the ground truth
40 fig = plt.figure(figsize=(4, 3))
41 ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)
42
43 for name, label in [('Setosa', 0),
44                    ('Versicolour', 1),
45                    ('Virginica', 2)]:
46     ax.text3D(X[y == label, 3].mean(),
47              X[y == label, 0].mean(),
48              X[y == label, 2].mean() + 2, name,
49              horizontalalignment='center',
50              bbox=dict(alpha=.2, edgecolor='w', facecolor='w'))
51 # Reorder the labels to have colors matching the cluster results
52 y = np.choose(y, [1, 2, 0]).astype(np.float)
53 ax.scatter(X[:, 3], X[:, 0], X[:, 2], c=y, edgecolor='k')
54 ax.w_xaxis.set_ticklabels([])
55 ax.w_yaxis.set_ticklabels([])
56 ax.w_zaxis.set_ticklabels([])
57 ax.set_xlabel('Petal width')
58 ax.set_ylabel('Sepal length')
59 ax.set_zlabel('Petal length')
60 ax.set_title('Ground Truth')
61 ax.dist = 12
62
63 fig.show()

```

Output:



References:

<https://scikit-learn.org/>

<https://scikit-learn.org/>

<https://www.datacamp.com>