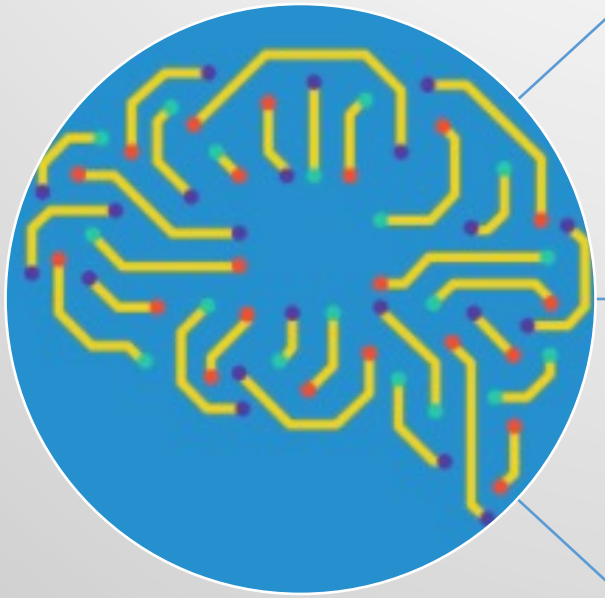
Natural Language Processing

Dr. Karen Mazidi

Part Five:
Machine Learning



Topics

- NumPy
- pandas
- sklearn
- seaborn

Quizzes

- Q: ML Basics
- Q: ML Libraries

Homework

- Paper summary

Python ML libraries

- NumPy – numerical array processing
- Pandas – data manipulation
- ScikitLearn – machine learning algorithms and data manipulation
- Seaborn – plotting

Installing

- Install with pip/pip3, or just use Google colab
- Upgrade pip:

```
python -m pip install --upgrade pip
```

Code 18.0.1 — pip. Using pip package manager.

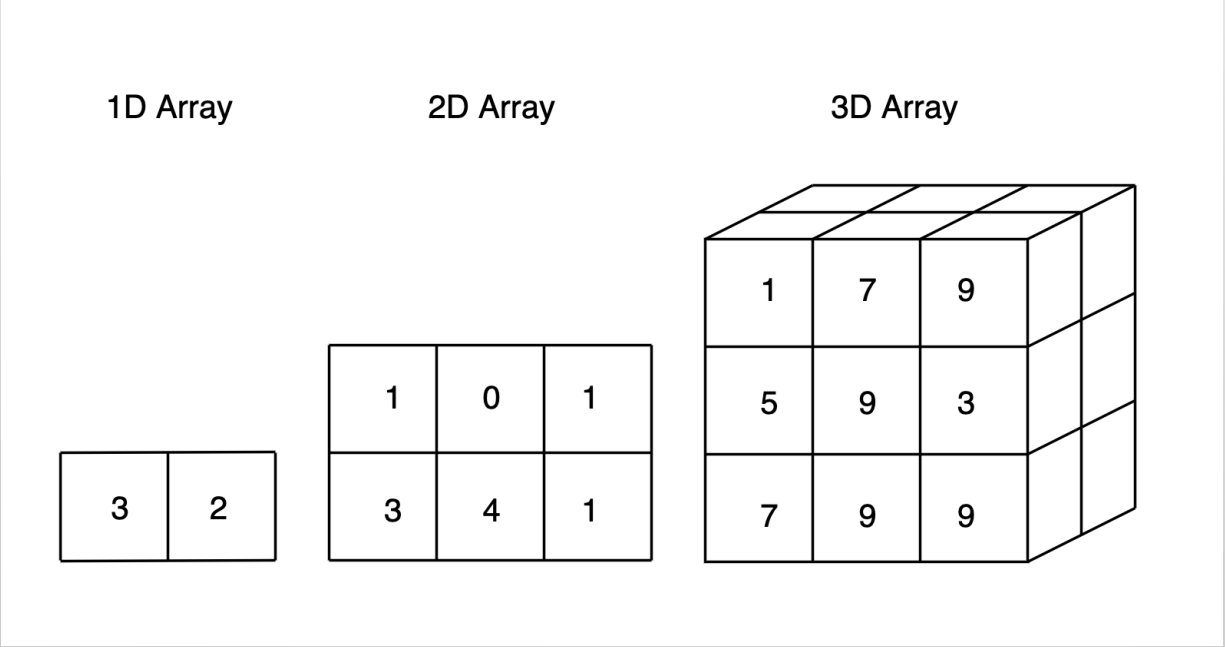
```
pip install x             # install package x
pip install x==2.01       # install x version 2.01
pip install x>=2.01       # install at least x version 2.01
pip install --upgrade x   # update package x
pip uninstall x            # uninstall package x
pip --version             # find your version
```

SciPy.org

- https://www.scipy.org/
- An organization for development of:

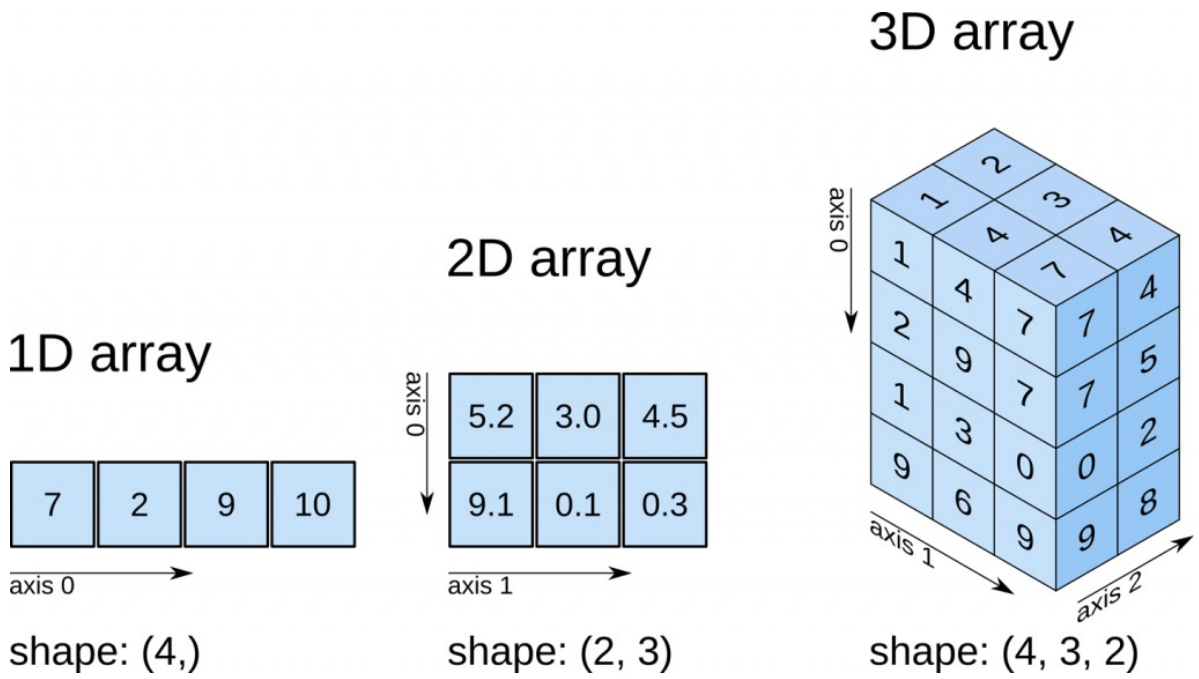
NumPy

- Efficient multi-dimensional arrays
- All elements are of the same type
- More efficient in storage and computation than Python lists



NumPy

- Efficient multi-dimensional arrays
- Rank: number of dimensions (axes)
- Shape: tuple (n, n, ...)



NumPy

- Build a numpy array from a Python list
- Index the same as a Python list

Code 18.1.1 — numpy. Create a 1d numpy array.

```
import numpy as np
a = np.array([1,2,3,4,5], float)

print(a[2])
3.0
```

NumPy

- Create a 2D array from a list of lists:

Code 18.1.2 — NumPy. Create a 2d numpy array.

```
b = np.array([[1,2,3], [4,5,6]], int)

b[1,1]
5
```

- Create an Nd array:

```
>>> np.arange(10, 30, 5)
array([10, 15, 20, 25])
>>> np.arange(0, 2, 0.3)  # it accepts float arguments
array([0. , 0.3, 0.6, 0.9, 1.2, 1.5, 1.8])
```

NumPy

- Shape and reshape

Code 18.1.3 — NumPy. Shape and rank.

```
c = np.array(range(10), float)
print("c originally: ", c)
c = c.reshape(5, 2)

print("c with the new shape: \n", c)
print("The new shape is: ", c.shape)
print("Length = ", len(c))
```

```
c originally:  [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9.]
c with the new shape:
 [[ 0.  1.]
 [ 2.  3.]
 [ 4.  5.]
 [ 6.  7.]
 [ 8.  9.]]
The new shape is:  (5, 2)
Length =  5
```

NumPy

- zero-filled arrays

Code 18.1.4 — NumPy. Create and initialize arrays.

```
c.fill(0) # c is overwritten with zeros
d = np.zeros((2,3)) # d is a 2x3 array of zeros
```

- there is also np.ones() and np.empty()

NumPy array operations

- Element-by-element: d += 5 # add 5 to each element
- or entire array:

Code 18.1.5 — NumPy. Operations on arrays.

```
np.sum(array1)  # sum all elements
np.mean(array1) # find the average of all elements
range = np.max(array1) - np.min(array1) # find the range
f = np.array(range(10), float).reshape(5,2)  # not working ???
print(f)
[[ 0.  1.]
 [ 2.  3.]
 [ 4.  5.]
 [ 6.  7.]
 [ 8.  9.]]
print(f.mean(axis=0)) # find mean of columns
[ 4.,  5.]
```

NumPy random numbers

- Set seed for reproducible results

```
np.random.seed(17)
rand_array = np.random.rand(2,3)
print(rand_array)
[[ 0.6375209   0.57560289  0.03906292]
 [ 0.3578136   0.94568319  0.06004468]]
```

NumPy Boolean selection

- The array:

```
[[ 0.6375209   0.57560289  0.03906292]
 [ 0.3578136   0.94568319  0.06004468]]
```

```
# boolean indexing
print(rand_array[rand_array>.2])
[ 0.6375209   0.57560289  0.3578136   0.94568319]
```

NumPy sort

- sorts on rows by default

```
# sort
print(np.sort(rand_array))
[[ 0.03906292  0.57560289  0.6375209 ]
 [ 0.06004468  0.3578136   0.94568319]]
```

NumPy arrays

- Have these attributes:
  - ndarray.ndim – number of axes
  - ndarray.shape – tuple of integers
  - ndarray.size – total number of elements
  - ndarray.dtype – int, float or more specific: int16, etc.
  - ndarray.itemsize – size in bytes of each element, ex:
    - Float64 = 64/8 = 8 bytes item size

Questions

- Create a 1D NumPy array, type double, length 30 using numbers 1 through 30
- Multiply each element by 5
- Square each element
- Sum the array
- Find the min and max
- Zero out the array

pandas

- Pandas series – one-dimensional data
- Pandas data frame – two-dimensional data
- Read in csv file:

Code 18.2.1 — Pandas. Read in Data.

```
import pandas as pd
df = pd.read_csv('Heart.csv', index_col='ID')
print(df.head())
```

Accessing elements

ID	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG
1	63	1	typical	145	233	1	2
2	67	1	asymptomatic	160	286	0	2
3	67	1	asymptomatic	120	229	0	2
4	37	1	nonanginal	130	250	0	0
5	41	0	nontypical	130	204	0	2

5 rows × 14 columns Open in new tab

- Element [1] from Age column

Code 18.2.2 — Pandas. Access data elements.

```
print(df['Age'][1])
print(df.Age[1])
```

```
63
63
```

- Notice ID [1] refers to the first row

Accessing elements

ID	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG
1	63	1	typical	145	233	1	2
2	67	1	asymptomatic	160	286	0	2
3	67	1	asymptomatic	120	229	0	2
4	37	1	nonanginal	130	250	0	0
5	41	0	nontypical	130	204	0	2

5 rows × 14 columns Open in new tab

- Two ways:
  - .loc method uses indices
  - .iloc method uses index positions
- Both of these lines below print 63

Code 18.2.3 — Pandas. Accessors loc and iloc.

```
print(df.loc[1, 'Age'])
print(df.iloc[0, 0])
```

pandas slicing

- When a single column is selected, it will be a Series, not a data frame
- To force it to be a data frame, surround the selection with double brackets

Code 18.2.5 — Pandas. Subsetting and Slicing.

```
df_not = df['Sex']     # not a data frame
df_new = df[['Sex']] # is a data frame
```

pandas select columns

- Specify selection in list of column names:

```
df_new = df[['Sex','Age']]
df_new.head()
```

ID	Sex	Age
1	1	63
2	1	67
3	1	67
4	1	37
5	0	41

5 rows × 2 columns Open in new tab

pandas slice

```
df['Age'][:5]   # this is a series not a data frame
```

```
ID
1    63
2    67
3    67
4    37
5    41
Name: Age, dtype: int64
```

```
df[['Age']][:5]  # this is a data frame because of the double brackets [[ ]]
```

ID	Age
1	63
2	67
3	67
4	37
5	41

Question

- From the Heart data frame:
  - Output the first 5 rows, all columns
  - Output the last 5 rows, only columns Age and Sex
  - Find the maximum Age and minimum Age
  - Select all rows with Age > mean Age
  - Select all rows with Sex==0

sklearn

- Sci-kit learn is an open source ML library since 2010 (dev. began 2007)
- Some built-in data sets
- Numerous functions for machine learning algorithms, data preprocessing, metrics, and more
- Next slide: look at iris built-in data set
- A “bunch” is a dictionary with key-value pairs

Code 18.3.1 — Scikit-Learn. Data Exploration.

```
from sklearn import datasets
iris = datasets.load_iris()
print(iris.data[:5])  # first 5 rows of data
[[ 5.1  3.5  1.4  0.2]
 [ 4.9  3.   1.4  0.2]
 [ 4.7  3.2  1.3  0.2]
 [ 4.6  3.1  1.5  0.2]
 [ 5.   3.6  1.4  0.2]]

print(iris.target[:5]) # first 5 labels
[0 0 0 0 0]

print('iris shape is ', iris.data.shape)  # get the shape of the data
iris shape is  (150, 4)

print(type(iris))
<class 'sklearn.utils.Bunch'>

print(iris.keys())
dict_keys(['data', 'feature_names', 'target', 'DESCR', 'target_names'])
```

```
print(iris.target)
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]

print(iris.target_names)
['setosa' 'versicolor' 'virginica']
```

sklearn example

- Data is already separated into features (.data) and target

Code 18.3.2 — Scikit-Learn. Set Up Data.

```
X = iris.data
y = iris.target

import pandas as pd
df = pd.DataFrame(X, columns=iris.feature_names)
df.head()
```

sklearn kNN

Code 18.3.3 — Scikit-Learn. kNN.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cross_validation import train_test_split  # outdated
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=7)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
knn.score(X_test, y_test)
0.9555555555555556
```

Seaborn

- Many plotting packages exist for Python
- Seaborn is easy to use, creates nice graphs
- Convert iris to data frame first:

```
# load iris
iris = datasets.load_iris()
X = iris.data
y = iris.target

# convert to data frames
df = pd.DataFrame(X, columns=iris.feature_names)
df_y = pd.DataFrame(y, columns=["species"])
```

Plot a quantitative array

- kde=True draws the curve, rug=Tree ticks below

Code 18.4.1 — Seaborn. Distribution Plot

```
sb.distplot(df["petal length (cm)"], kde=True, rug=True)
```

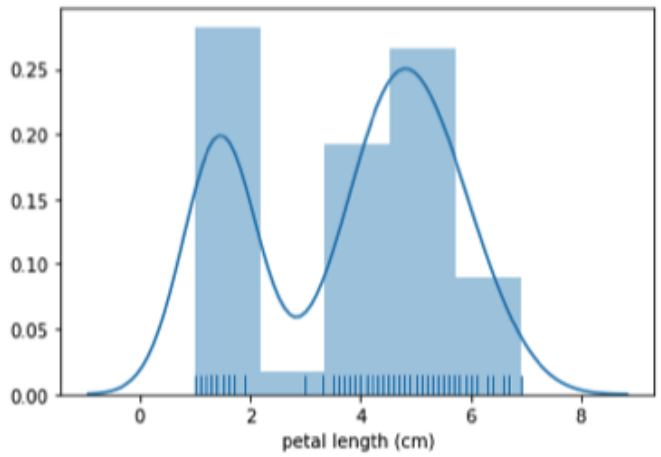


Figure 18.1: Distribution Plot

Plot 2 quantitative arrays

- Using hue and style

Code 18.4.2 — Seaborn. Relation Plot

```
sb.relplot(x="petal length (cm)", y="petal width (cm)",
       data=df, hue=df_y.species, style=df_y.species)
```

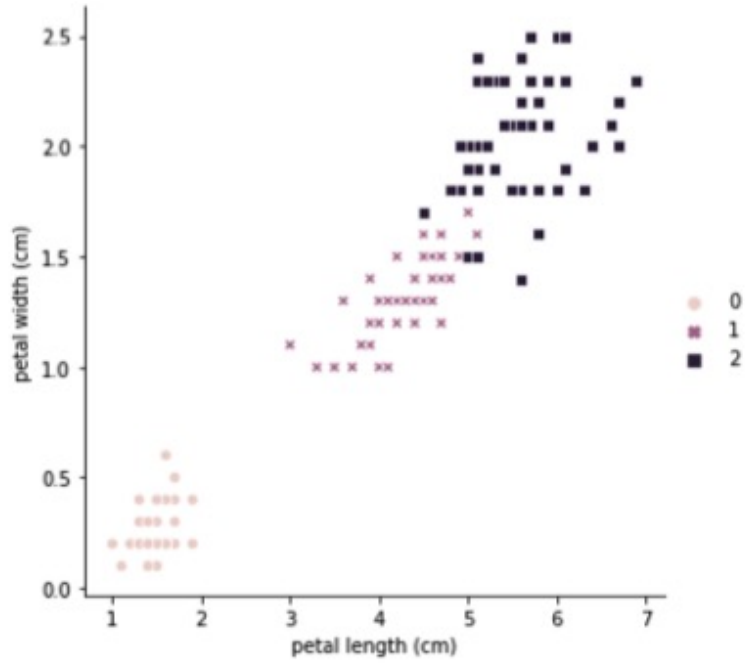


Figure 18.2: Relation Plot

Categorical plot

Code 18.4.3 — Seaborn. Category Plot

```
sb.catplot(x="species", kind="count", data=df_y)
```

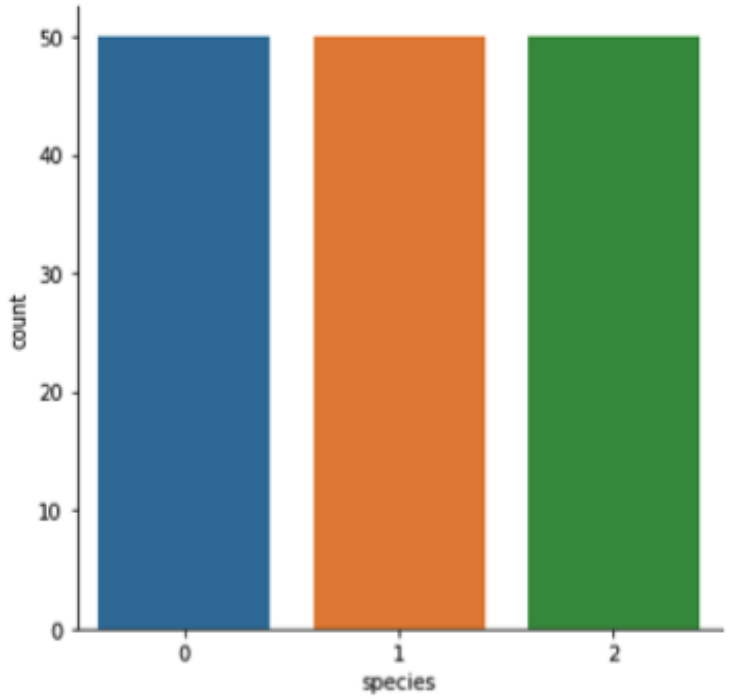


Figure 18.3: Category Plot

X categorical, y quantitative

- Combine data, target for plot first

Code 18.4.4 — Seaborn.

```
df_temp = pd.concat([df.reset_index(drop=True),
      df_y.reset_index(drop=True)], axis=1)
sb.catplot(x="species", y="petal length (cm)", data=df_temp)
```

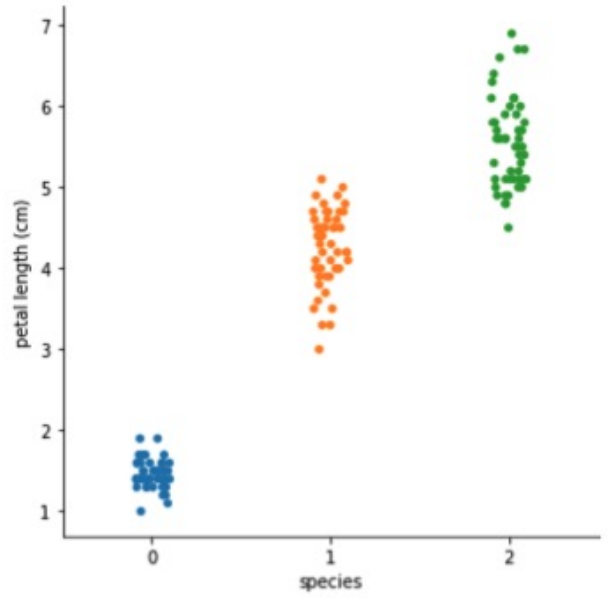


Figure 18.4: Category and Petal Length Plot

Code Examples

- GitHub Part 5 Chapter 18

Essential points to note

- Using the libraries becomes easier with practice
- See the Cheat Sheets in eLearning

To Do

- Quiz on ML Libraries

Next class

Getting started with ML