# Data loading

**import pandas as pd**

iris\_filename = 'datasets-uci-iris.csv'

iris = **pd.read\_csv(**iris\_filename, sep=',', decimal='.', header=None, names= ['sepal\_length', 'sepal\_width', 'petal\_length', 'petal\_width', 'target'])

iris.head()

iris.tail()

iris.columns

Y = iris['target']

iris.shape

dataset.**fillna**(50) # fill NaN values .mean(), .median()

bad\_dataset = pd.read\_csv('a\_loading\_example\_2.csv', **error\_bad\_lines=False**)

iris\_chunks = pd.read\_csv(iris\_filename, header=None, names=['C1', 'C2', 'C3', 'C4', 'C5'], **chunksize=10**)

for chunk in iris\_chunks:

print ('Shape:', chunk.shape)

print (chunk,'\n')

iris\_iterator = pd.read\_csv(iris\_filename, header=None, names=['C1', 'C2', 'C3', 'C4', 'C5'], **iterator=True**)

my\_own\_dataset = **pd.DataFrame**({'Col1': range(5), 'Col2': [1.0]\*5, 'Col3': 1.0, 'Col4': 'Hello World!'})

my\_own\_dataset.**dtypes**

**import csv**

with open(iris\_filename, 'rt') as data\_stream:

# 'rt' mode

for n, row in enumerate(**csv.DictReader**(data\_stream,

fieldnames = ['sepal\_length', 'sepal\_width','petal\_length', 'petal\_width', 'target'],

dialect='excel')):

with open(iris\_filename, 'rt') as data\_stream:

for n, row in enumerate(**csv.reader**(data\_stream, dialect='excel')):

# Data preprocessing

mask\_feature = iris['sepal\_length'] > 6.0

iris['target'].**unique()**

grouped\_targets\_var = iris.**groupby(['target']).var()**

iris.**sort\_index(by='sepal\_length')**.head()

iris.**apply(np.count\_nonzero, axis=1)**.head() # by row

iris.apply(np.count\_nonzero, **axis=0**) #by column(feature)

iris.**applymap**(lambda el:len(str(el))).head()

# Data selection

**import pandas as pd**

dataset = pd.read\_csv('a\_selection\_example\_1.csv')

dataset = pd.read\_csv('a\_selection\_example\_1.csv', **index\_col=0**)#column as index

dataset['val3'][104]

dataset.loc[104, 'val3']

dataset.iloc[4, 2]

dataset[['val3', 'val2']][0:2]

dataset.loc[range(100, 102), ['val3', 'val2']]

dataset.iloc[range(2), [2,1]]

# Categorical data

**import pandas as pd**

categorical\_feature = pd.**Series**(['sunny', 'cloudy', 'snowy','rainy', 'foggy'])

mapping = pd.**get\_dummies**(categorical\_feature)

**from sklearn.preprocessing import OneHotEncoder**

**from sklearn.preprocessing import LabelEncoder**

le = **LabelEncoder**() #maps text to a 0-to-N integer

ohe = **OneHotEncoder**()

levels = ['sunny', 'cloudy', 'snowy', 'rainy', 'foggy']

fit\_levs = le.**fit\_transform**(levels)

ohe.**fit**([[fit\_levs[0]], [fit\_levs[1]], [fit\_levs[2]], [fit\_levs[3]],[fit\_levs[4]]])

# Text data

**from sklearn.datasets import fetch\_20newsgroups**

categories = ['sci.med', 'sci.space']

twenty\_sci\_news = fetch\_20newsgroups(categories=categories)

print(twenty\_sci\_news.**data[0]**)

**from sklearn.feature\_extraction.text import CountVectorizer**

count\_vect = **CountVectorizer()**

word\_count = count\_vect.**fit\_transform**(twenty\_sci\_news.data)

word\_count.**shape**

word\_list = count\_vect.**get\_feature\_names()**

for n in word\_count[0]**.indices:**

print ('Word "%s" appears %i times' % (word\_list[n], word\_count[0, n]))

**from sklearn.feature\_extraction.text import TfidfVectorizer**

tf\_vect = **TfidfVectorizer**(use\_idf=False, **norm='l1'**)

word\_freq = tf\_vect.**fit\_transform**(twenty\_sci\_news.data)

word\_list = tf\_vect.**get\_feature\_names()**

for n in word\_freq[0].**indices**:

print ('Word "%s" has frequency %0.3f' % (word\_list[n], word\_freq[0, n]))

count\_vect\_1\_grams = **CountVectorizer(ngram\_range=(1, 1),stop\_words=[], min\_df=1)**

word\_count = count\_vect\_1\_grams.**fit\_transform(documents)**

word\_list = count\_vect\_1\_grams.**get\_feature\_names()**

count\_vect\_1\_grams = CountVectorizer(**ngram\_range=(2, 2))**

word\_count = count\_vect\_1\_grams.**fit\_transform(documents)**

word\_list = count\_vect\_1\_grams.**get\_feature\_names()**

**from sklearn.feature\_extraction.text import HashingVectorizer**

hash\_vect = **HashingVectorizer**(n\_features=1000)

word\_hashed = hash\_vect.**fit\_transform(twenty\_sci\_news.data)**

word\_hashed.shape

# Scraping Web – Beautiful Soup

**import urllib.request**

**from bs4 import BeautifulSoup**

url = 'https://en.wikipedia.org/wiki/William\_Shakespeare'

request = **urllib.request.Request**(url)

response = **urllib.request.urlopen**(request)

soup = **BeautifulSoup**(response, 'html.parser')

section = soup.**find\_all**(id='mw-normal-catlinks')[0]

for catlink in section.**find\_all**("a")[1:]:

print(catlink.get("title"), "->", catlink.get("href"))

# Data processing with NumPy

– Dataframes are made of one-dimensional NumPy arrays → Faster by column/slower by rows