AI Games course

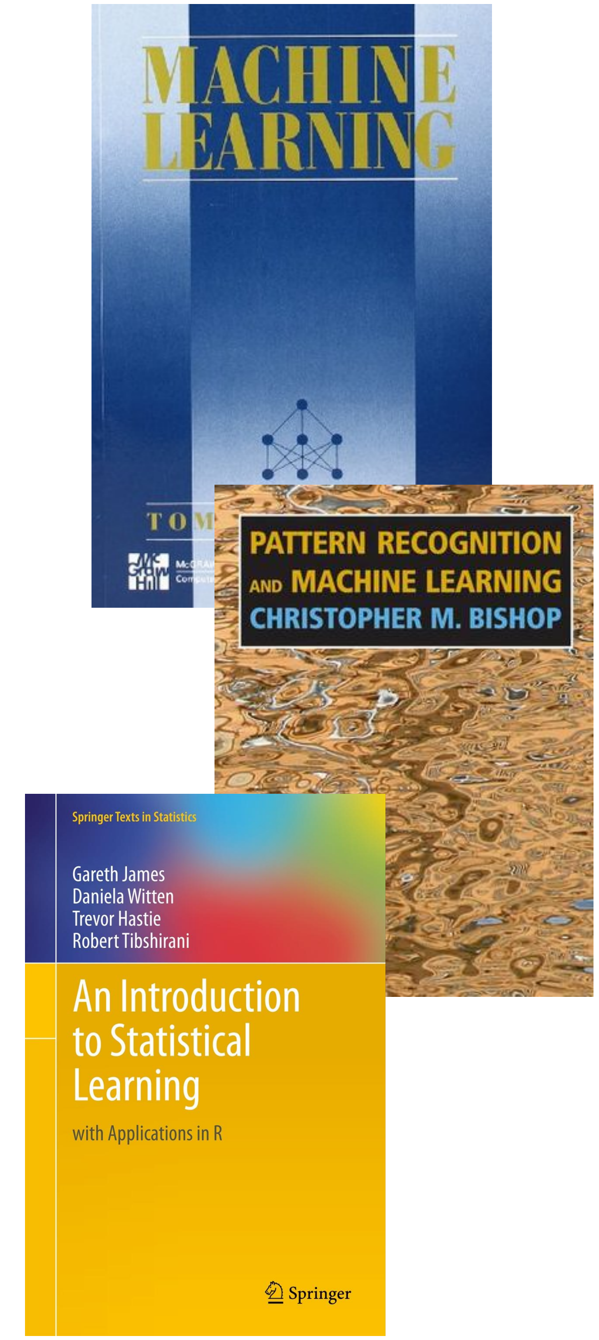
Certificate 2, session 1

Introduction to Machine Learning

Part 1. A few words about learning

What is learning?

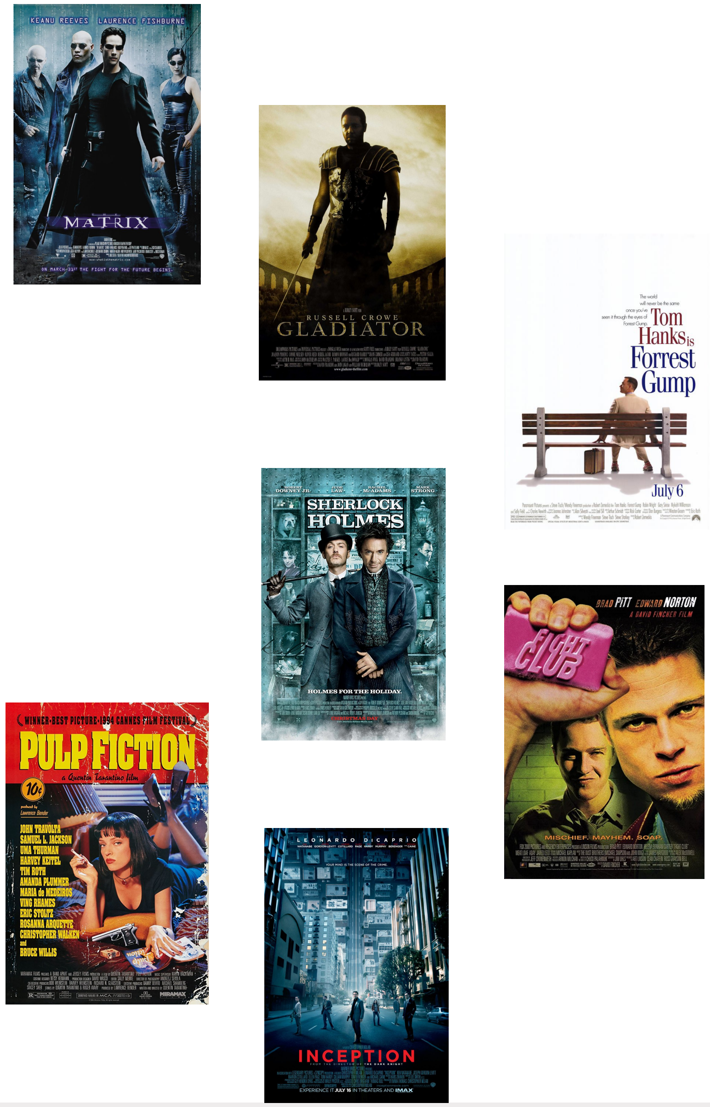
- **Mitchell**: learning is *improving with experience* at some task *T* according to a performance measure M;
- systems are not *explicitly programmed* to carry out *T*; they are programmed how to learn to carry out *T*;
- learning usually means *learning from data.*



Example:
choosing a film for a user

IMDb

- Do you like *The Matrix*? — Yes
- Do you like *Gladiator*? — No
- Do you like *Forest Gump*? — Yes
- Do you like *Sherlock Holmes*? — No
- Do you like *Fight Club*? — Yes
- Do you like *Pulp Fiction*? — Yes!
- Do you like *Inception*? — Yes!!



AIgaming.com

Supervised learning

- collect a set of *data instances*

    emails, pictures, user profiles, tweets etc.

- represent them in a machine-readable format, as *vectors of features*

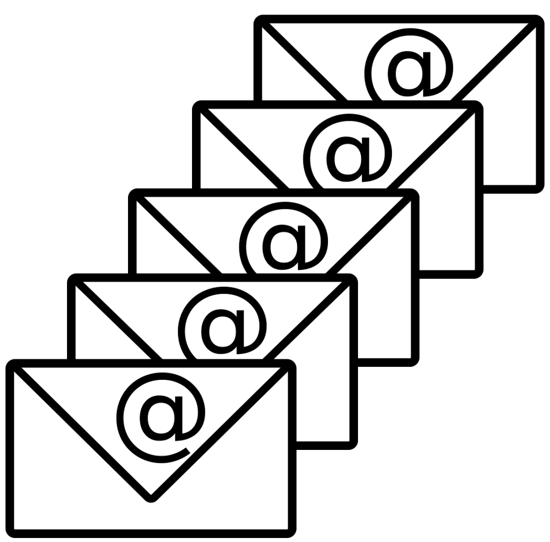
    [1, 0, 0, 0.85, -3, 4, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 275]

- provide *correct labels* for every data instance

    “spam” - “not spam”, age, “solvent” - “insolvent”, topic etc.

- train a *model* to assign correct labels to instances (potentially unseen)

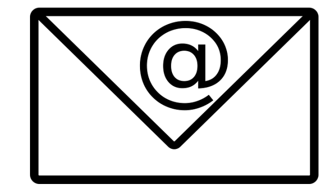
Supervised learning



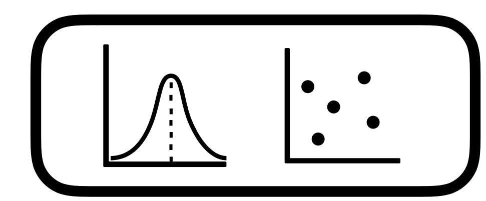
data instances

[0,1,1,0,1,0,**spam**]
[1,1,0,1,1,1,**not-spam**]
[0,1,1,0,0,0,**spam**]
[0,1,1,1,1,1,**not-spam**]
[0,0,0,1,1,0,**not-spam**]

feature vectors of instances



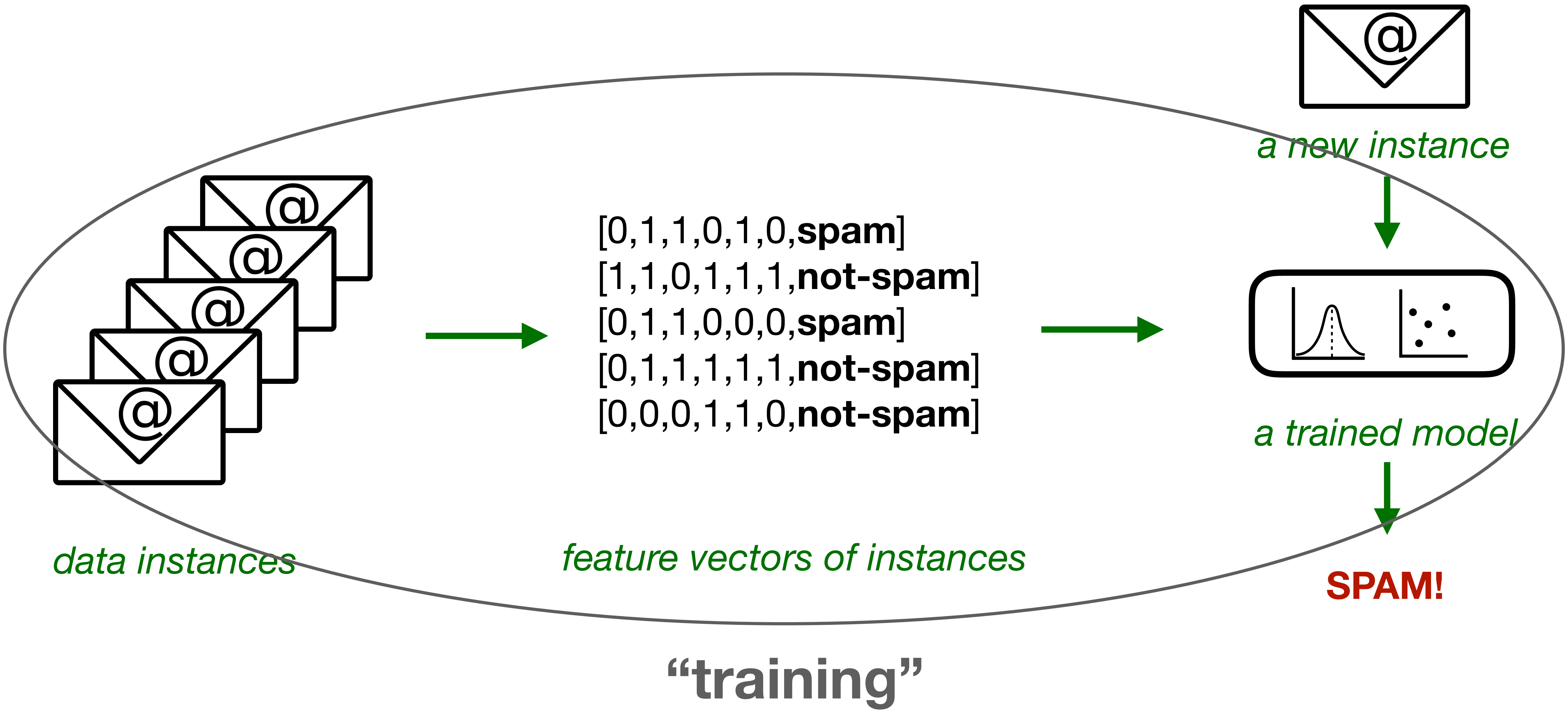
a new instance



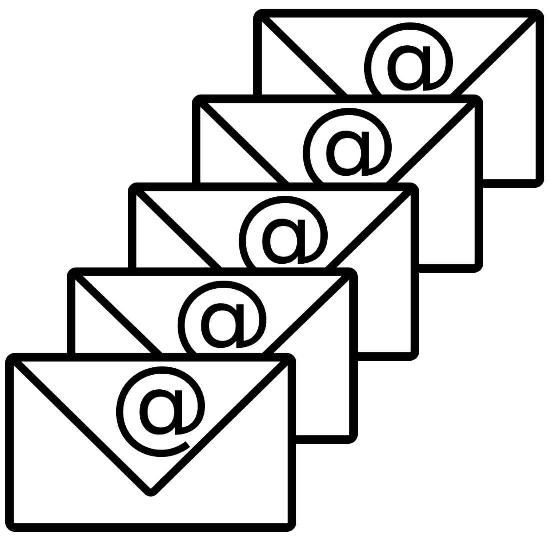
a trained model

SPAM!

Supervised learning



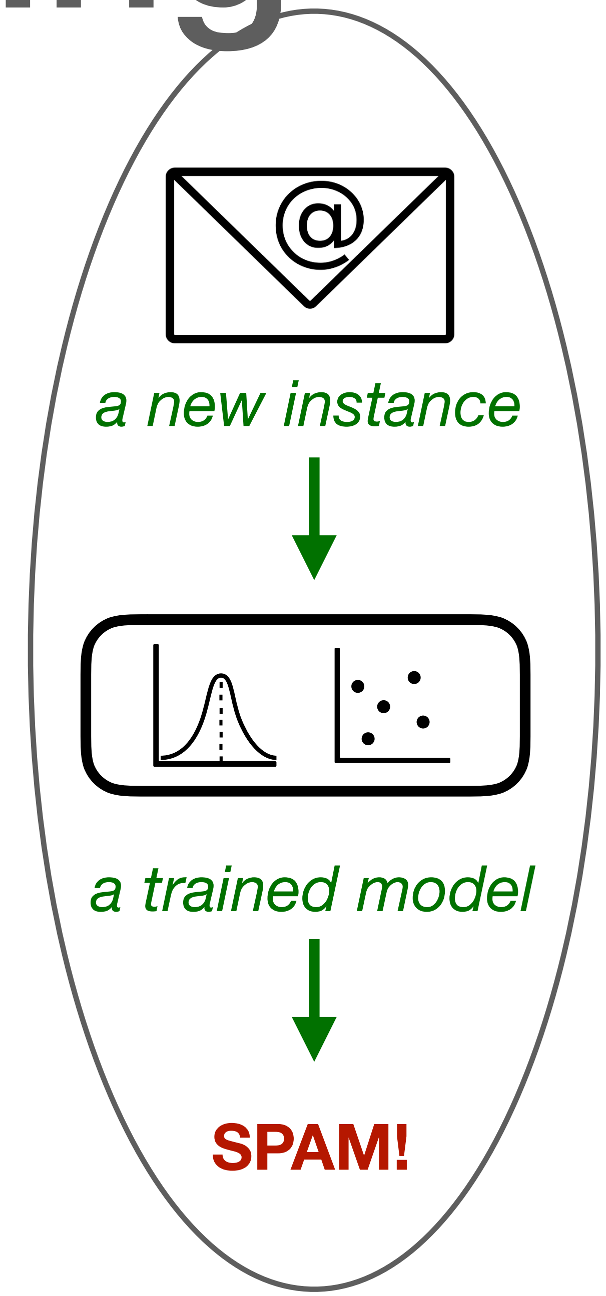
Supervised learning



data instances

[0,1,1,0,1,0,**spam**]
[1,1,0,1,1,1,**not-spam**]
[0,1,1,0,0,0,**spam**]
[0,1,1,1,1,1,**not-spam**]
[0,0,0,1,1,0,**not-spam**]

feature vectors of instances



“testing”

Types of learning-1

- **Supervised learning**: we know correct labels

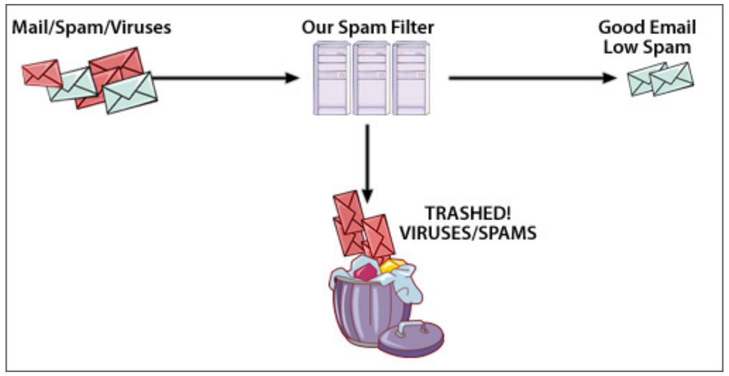
    Ex: an email is either “spam” or “not spam”

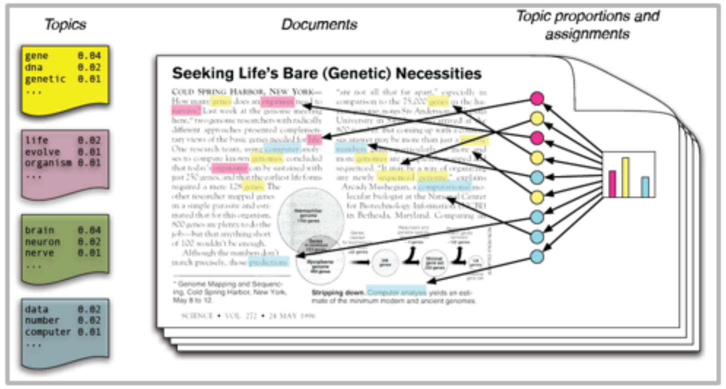
- **Unsupervised learning**: no labels given

    Ex: a set of articles you need to group by topics, but you do not know their topic labels

- **Semi-supervised learning**: we know labels for a fraction of data (usually small)

    Ex: same set of articles, but for some of them you do know the topic label





Types of learning-2

- **Classification**: labels are categories

  Ex1: an email is either “spam” or “not spam”

  Ex2: bank clients have a known credit rating **category**

- **Regression**: labels are real numbers

  Ex: every bank client has a known numeric credit rating **score**

- **Clustering**: instances are divided into groups (number and type of groups unknown)

  Ex: new patients have medical profiles and need to be grouped wrt health issues

Part 2. Classification example

Bankruptcy prediction

- analytics data on 250 companies:
  http://archive.ics.uci.edu/ml/datasets/Qualitative_Bankruptcy

- based on the data, we need to decide whether or not a company goes bankrupt,

- i.e., *classify* every instance as *bankrupt* or *not bankrupt.*

BANKRUPTCY features

- for each company there is information on:
  - **industrial risk**: degree of competition, growth of market demand etc.
  - **management risk**: management stability, HR management, business planning etc.
  - **financial flexibility**: direct and indirect financing
  - **credibility**: credit history, relations with financial institutions etc.
  - **competitiveness**: market position, differentiated strategies etc.
  - **operating risk**: production efficiency, effectiveness of sale network etc.
- all attributes can have values *positive (1), average (0), negative(-1)*
- every company is labeled as *bankrupt (B)* or *not bankrupt (NB)*

Bankruptcy dataset

- such a complicated qualitative analysis is captured in a simple, concise, machine-readable form:

P,P,A,A,A,P,**NB**
N,N,A,A,A,N,**NB**
P,N,N,N,N,N,**B**
P,P,P,P,P,P,**NB**
N,N,N,A,N,P,**B**
A,A,P,P,P,A,**NB**
P,P,A,P,P,P,**NB**
P,N,N,N,A,A,**B**
…

every line represents information about one company

this company has positive credibility

this company went bankrupt

- download bankruptcy_dataset.txt

Step 1: preprocessing

```python
import numpy as np
import re


# parsing input file into an array
file = open('bankruptcy_dataset.txt', 'r')
input = file.read()

input = input.replace('NB', '1')
input = input.replace('P', '2')
input = input.replace('A', '1')
input = input.replace('N', '0')
input = input.replace('B', '2')

instance_strings = input.splitlines()
print(len(instance_strings))  # 250 instances

np.random.shuffle(instance_strings) # important for train-test

instances = []
for s in instance_strings:
    instance = re.split(r",", s)
    instances.append(instance)
print(len(instances))
```

Step 1: creating a dataset

```
# creating a dataset
data = []
targets = []

for i in instances:
    instance_data = i[:6]
    data.append(instance_data)
    target = i[6]
    targets.append(target)

print(len(targets))
print(len(data))

target_names = ['bankrupt', ‘not-bankrupt']

dataset = {
    u'data': data,
    u'targets':targets,
    u'target_names':target_names
}
```

Step 3: training a model

```
dataset_X = dataset.get(‘data')
dataset_Y = dataset.get('targets')
print(np.unique(dataset_Y))


dataset_X_train = dataset_X[:225] # 90% for training
dataset_Y_train = dataset_Y[:225]
dataset_X_test  = dataset_X[225:] # 10% for testing
dataset_Y_test  = dataset_Y[225:]


# train a model
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()

model.fit(dataset_X_train, dataset_Y_train)
```

Step 4: Testing the model

```python
#test
predictions = model.predict(dataset_X_test)

print(predictions)
print(dataset_Y_test)
```

Output:

 # predictions
[1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 1, 1, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 1, 2, 2, 2, 1, 2, 1, 1]
 # answers
[1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 1, 1, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 1, 2, 2, 2, 1, 2, 1, 1]

Playing with the model

Try modifying some of the parameters of the pipeline and see what happens:

- change the train-test ratio (*80-20*, *70-30* etc.)
- remove some of the features
- change the learning algorithm:
  - logistic regression: http://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  - Support Vector Machines: http://scikit-learn.org/stable/modules/svm.html
  - Decision Problems: http://scikit-learn.org/stable/modules/tree.html