

Μηχανική Μάθηση – Machine Learning (ML)

Εισαγωγή

Τι είναι μηχανική μάθηση;

Ποιες είναι οι βασικές 2 κατηγορίες μηχανικής μάθησης

Βασικά παραδείγματα μηχανικής μάθησης

Πως «δουλεύει» η μηχανική μάθηση

Εισαγωγή – Τί είναι μηχανική μάθηση

Είναι η εξαγωγή γνώσης από δεδομένα μέσω μιας σχεδόν αυτόματης διεργασίας;

- **Γνώση από δεδομένα:** Ξεκινά με κάποια ερώτηση που μπορεί να απαντηθεί από τα δεδομένα
- **Αυτοματοποιημένη εξαγωγή:** Ένας Η/Υ μπορεί να δώσει αυτή την δυνατότητα
- **Ημι-αυτοματοποιημένη εξαγωγή:** Χρειάζονται έξυπνες αποφάσεις από ανθρώπινη παρέμβαση

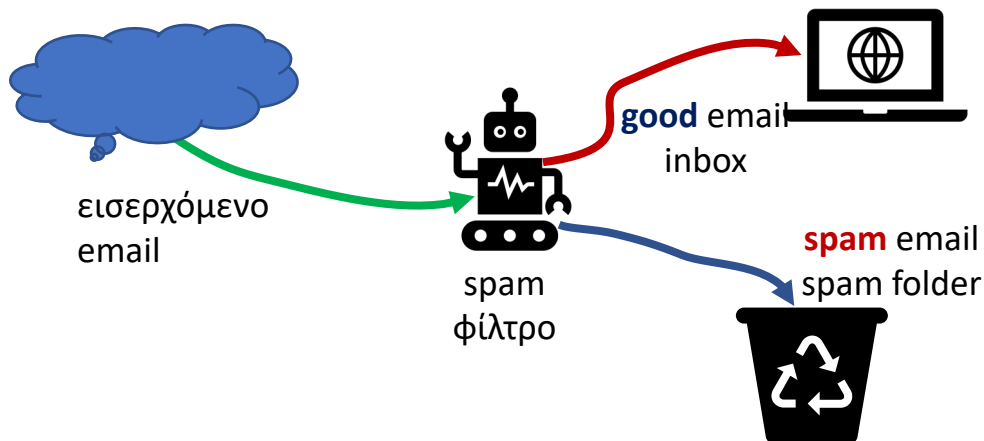
Εισαγωγή – Βασικές κατηγορίες μηχανικής μάθησης

Υπάρχουν δύο βασικές κατηγορίες μηχανικής μάθησης:

- **Μάθηση υπό επίβλεψη (supervised learning) ή Μοντελοποίηση πρόβλεψης (predictive modeling)**

Προσπάθεια να κάνουμε προβλέψεις χρησιμοποιώντας δεδομένα

Παράδειγμα:



Επειδή υπάρχει ένα συγκεκριμένο αποτέλεσμα που προσπαθούμε να βρούμε η διεργασία αυτή αποτελεί – supervised learning

Εισαγωγή – Βασικές κατηγορίες μηχανικής μάθησης

➤ Μάθηση χωρίς επίβλεψη – unsupervised learning

Διεργασία εξαγωγής χαρακτηριστικής δομής από τα δεδομένα ή η διεργασία μάθησης για την καλύτερη αναπαράσταση των δεδομένων.

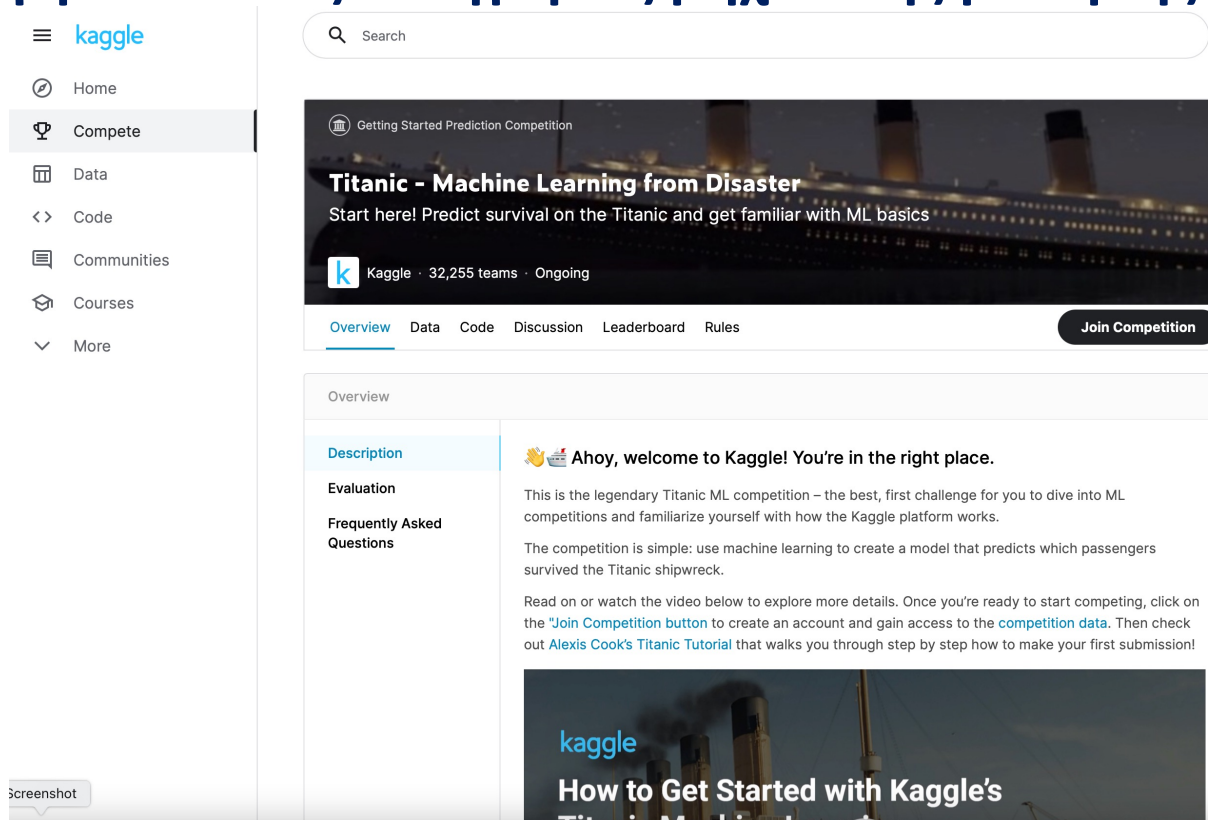
Παράδειγμα:

Έστω τα δεδομένα είναι η αγοραστική συμπεριφορά καταναλωτών σε ένα supermarket. Η διεργασία της μάθησης χωρίς επίβλεψη, θα ήταν να κατατάξω τους καταναλωτές σε ομάδες (clusters) που εμφανίζουν ανάλογη συμπεριφορά

Για παράδειγμα, οι φοιτητές, οικογένειες με μικρά παιδιά, ή ηλικιωμένοι έχουν ιδιαίτερη αγοραστική συμπεριφορά παρόμοια με άλλα άτομα των ομάδων τους και αρκετά διαφορετικές από τη συμπεριφορά που εμφανίζουν μέλη των άλλων ομάδων.

Το παραπάνω αποτελεί παράδειγμα unsupervised learning διεργασίας, γιατί δεν υπάρχει σωστή ή λάθος απάντηση σχετικά με το πόσες δομές υπάρχουν στα δεδομένα, ποια άτομα ανήκουν σε κάθε δομή ή ακόμα πως θα περιγράψουμε κάθε δομή

Εισαγωγή – Βασικές κατηγορίες μηχανικής μάθησης



The screenshot shows the Kaggle website interface. On the left is a navigation menu with links to Home, Compete, Data, Code, Communities, Courses, and More. The main content area displays the 'Titanic - Machine Learning from Disaster' competition. It includes a search bar at the top, a banner image of the Titanic ship, and a section titled 'Overview' with tabs for Description, Data, Code, Discussion, Leaderboard, and Rules. The 'Description' tab is active, showing a welcome message and details about the competition. A 'Join Competition' button is visible in the top right corner of the competition page.

Titanic - Machine Learning from Disaster
Start here! Predict survival on the Titanic and get familiar with ML basics

Kaggle · 32,255 teams · Ongoing

[Overview](#) [Data](#) [Code](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Join Competition](#)

Overview

Description

Evaluation

Frequently Asked Questions

Ahoy, welcome to Kaggle! You're in the right place.

This is the legendary Titanic ML competition – the best, first challenge for you to dive into ML competitions and familiarize yourself with how the Kaggle platform works.

The competition is simple: use machine learning to create a model that predicts which passengers survived the Titanic shipwreck.

Read on or watch the video below to explore more details. Once you're ready to start competing, click on the [Join Competition button](#) to create an account and gain access to the [competition data](#). Then check out [Alexis Cook's Titanic Tutorial](#) that walks you through step by step how to make your first submission!

How to Get Started with Kaggle's Titanic Machine Learning

Από την Kaggle που είναι μια πλατφόρμα για ML διαγωνισμούς.
Ένας από τους διαγωνισμούς είναι αυτός για τον Τιτανικό του Τιτανικού.

Ο σκοπός είναι να βρεθούν ποιοι επιβάτες επέζησαν το ναυάγιο

Η ερώτηση: Είναι supervised ή unsupervised learning?

Είναι supervised learning γιατί ο σκοπός είναι να προβλέψουμε κάποιο συγκεκριμένο αποτέλεσμα: η επιβίωση κάθε επιβάτη

Μηχανική μάθηση - Πως δουλεύει

Στα επόμενα θα επικεντρωθούμε στην περίπτωση της supervised learning

Πώς ακριβώς ωστόσο μπορούμε να χρησιμοποιήσουμε την supervised learning?

Πρώτο βήμα:

- Εκπαίδευση ενός μοντέλου μηχανικής μάθησης, χρησιμοποιώντας δεδομένα με ετικέτες (**labeled data**)

Labeled data, είναι τα δεδομένα που έχουν χαρακτηριστεί με το αποτέλεσμα της ML π.χ. στην περίπτωση του παραδείγματος των εισερχόμενων e-mails, κάθε μήνυμα μπορεί να χαρακτηριστεί ως κανονικό ή spam

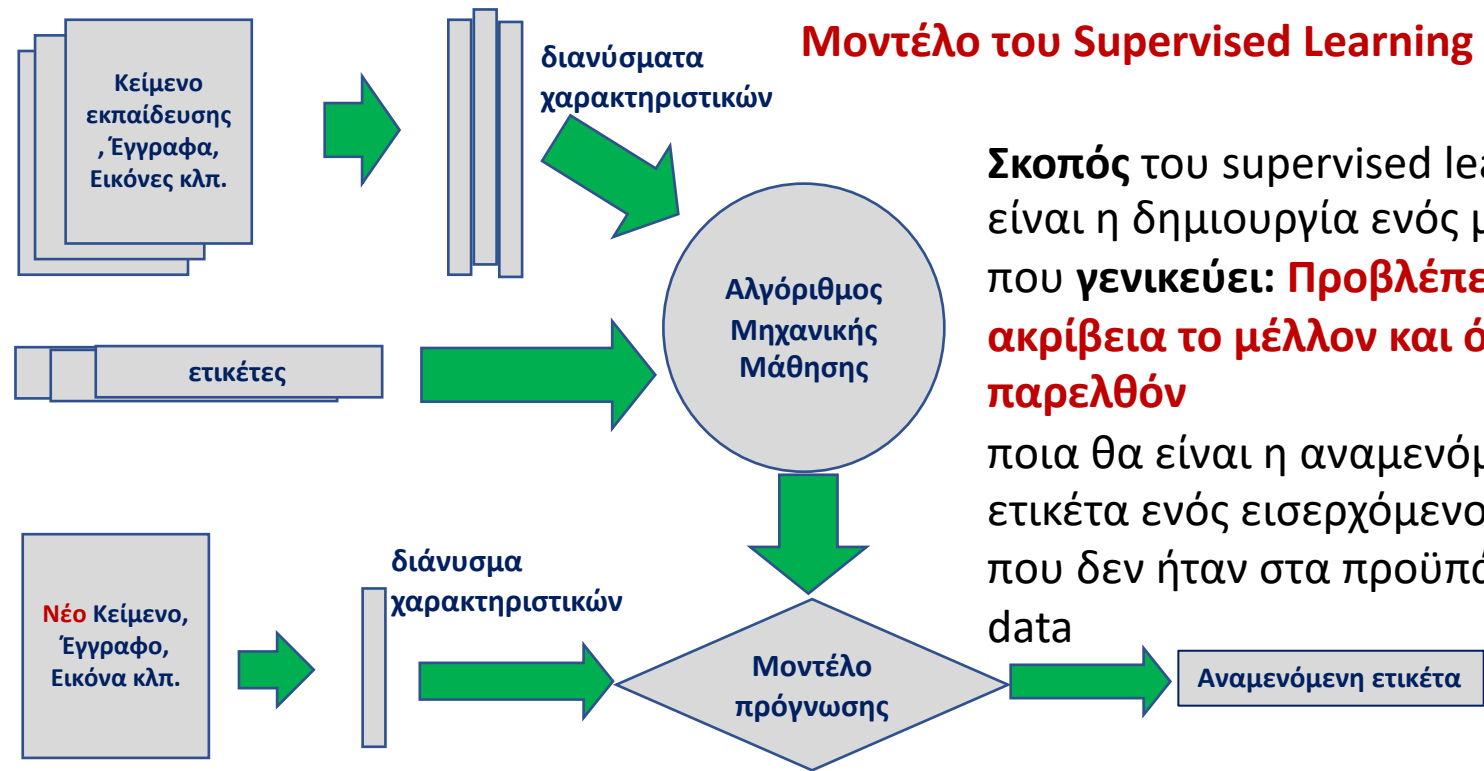
- Αυτό αποτελεί τη λεγόμενη εκπαίδευση του μοντέλου (**model training**) γιατί το μοντέλο μαθαίνει τη σχέση μεταξύ των χαρακτηριστικών των δεδομένων και το επιθυμητό αποτέλεσμα.

Τα χαρακτηριστικά των δεδομένων (όπως συμβαίνει στο παράδειγμα του spam filter) μπορεί να περιέχουν τον τίτλο του email, τον αριθμό των συνδέσμων που περιέχονται στο email, το μήκος του email κλπ.

Μηχανική μάθηση - Πως δουλεύει

Δεύτερο βήμα:

- Πραγματοποίηση πρόβλεψης σε νέα δεδομένα όπου δεν υπάρχουν ετικέτες για τα χαρακτηριστικά των δεδομένων. Δηλαδή δεν ξέρουμε το πραγματικό αποτέλεσμα. Θέλουμε χρησιμοποιώντας το εκπαιδευμένο μοντέλο να κάνουμε ακριβή πρόβλεψη για το αποτέλεσμα άγνωστων δεδομένων. Στην περίπτωση του παραδείγματος των emails, θέλουμε να προβλέψουμε αν το εισερχόμενο email είναι spam ή κανονικό χωρίς να υπάρξει ανθρώπινη παρέμβαση



Μηχανική μάθηση – Τα συστατικά

Τα βασικά συστατικά ενός αλγόριθμου μηχανικής μάθησης είναι:

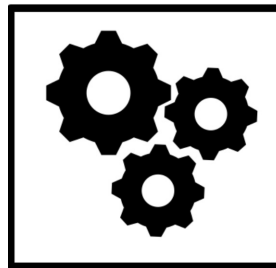
Αναπαράσταση
δεδομένων

01100
10110
11110

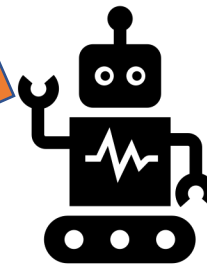
Αλγόριθμος
βελτιστοποίησης



Ρυθμιζόμενο
μοντέλο



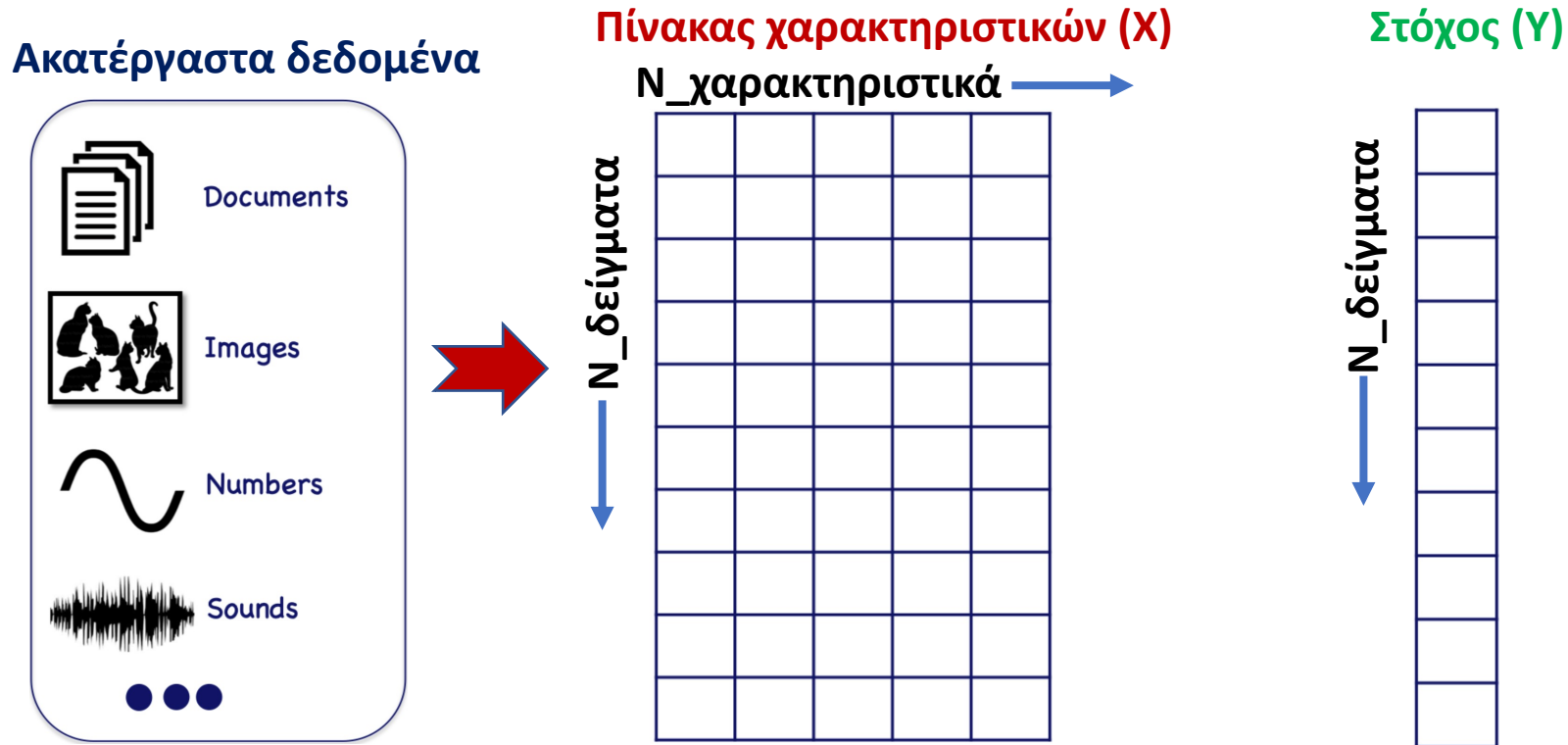
Μοντέλο
εκπαίδευσης



Συστατικά ML - Αναπαράσταση δεδομένων

Τα δεδομένα μπορεί να υπάρχουν σε διάφορες μορφές: έγγραφα, εικόνες, αριθμητικά δεδομένα, ήχοι, σήματα κλπ.

Θα πρέπει να δοθούν σε μορφή που είναι αναγνώσιμη από τους υπολογιστές ως ακολούθως



Αντιπροσωπεύουμε τα data σε έναν πίνακα, όπου κάθε γραμμή είναι ένα δείγμα των data

Οι στήλες αντιπροσωπεύουν διαφορετικά χαρακτηριστικά (**features**) των δειγμάτων, π.χ.

χρώμα, υφή, μέγεθος, μυρωδιά, σχήμα. Ο πίνακας αυτός ονομάζεται **πίνακας χαρακτηριστικών**

Ο στόχος (**target**) πίνακας, αποθηκεύει το αποτέλεσμα. Είναι η ετικέτα

Συστατικά ML: Ρυθμιζόμενο μοντέλο & Αλγόριθμος Βελτιστοποίησης

Τη στιγμή που έχουμε προετοιμάσει τα εισαγώμενα και εξαγώμενα data, χρειαζόμαστε ένα μοντέλο που μπορεί να εκπαιδευτεί από τα data.

Συνήθως οι αλγόριθμοι έχουν πολλούς παραμέτρους που μπορούν να ρυθμιστούν με τέτοιο τρόπο ώστε το μοντέλο να είναι ιδιαίτερα αποδοτικό για τις ανάγκες των data που του εμφανίζουν. Υπάρχουν πολλά μοντέλα, όπως νευρωνικά δίκτυα, τυχαία δάση κλπ.

Ο **αλγόριθμος βελτιστοποίησης** είναι το πλέον σημαντικό τμήμα ενός αλγόριθμου μηχανικής μάθησης.

Πως γίνεται η βελτιστοποίηση του αλγορίθμου? Ορίζεται μια αντικειμενική συνάρτηση (**objective function**) και χρησιμοποιούνται αλγόριθμοι βελτιστοποίησης για να την ελαχιστοποιήσουμε ή μεγιστοποιήσουμε.

Η objective function μπορεί να υπολογιστεί από τη διαφορά/σφάλμα, μεταξύ της εκτιμώμενης και του στόχου (πραγματικής τιμής). Επομένως όσο μικρότερο το σφάλμα, τόσο καλύτερη η συνάρτηση.

Υπάρχουν διάφοροι αλγόριθμοι βελτιστοποίησης όπως ελαχιστοποίηση ελαχίστων τετραγώνων

Εκπαιδευμένο μοντέλο: Από τη στιγμή που έχει προσδιοριστεί το ρυθμιζόμενο μοντέλο, είναι σε θέση να κάνει προβλέψεις και κατηγοροποιήσεις. Το μοντέλο αυτό μπορεί να χρησιμοποιηθεί σε νέα δεδομένα για να πετύχουμε το σκοπό μας.

Κατηγοροποίηση - Classification

Η κατηγοροποίηση είναι ένα συνηθισμένο πρόβλημα στην καθημερινή ζωή.

Για παράδειγμα, θέλουμε να κατηγοροποιήσουμε διάφορα προϊόντα ως καλής ή/και κακής ποιότητας, εισερχόμενα e-mails ως κανονικά ή spam, ταινίες ή βιβλία ως ενδιαφέρουσες κλπ.

Δυο βασικοί παράγοντες ανάγουν το πρόβλημα σε πρόβλημα κατηγοροποίησης:

(α) Το πρόβλημα έχει συγκεκριμένη απάντηση (ετικέτα) και

(β) Το αποτέλεσμα που θέλουμε είναι το ίδιο κατηγοροποιημένο, όπως ΝΑΙ/ΟΧΙ ή κάποια διαφορετική κατηγορία .

Παράδειγμα: κατηγοροποίηση δεδομένων με μήλα και πορτοκάλια – **binary classification**

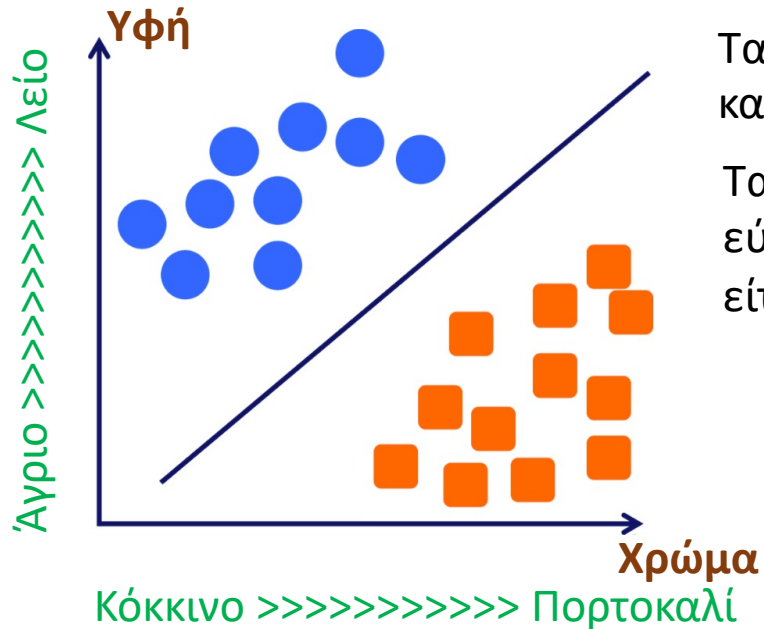


0: πορτοκάλι

1: μήλο

Κατηγοροποίηση - Classification

Από τη στιγμή που έχουμε 5 χαρακτηριστικά για τα δεδομένα είναι δύσκολο να οπτικοποιήσουμε τα χαρακτηριστικά μεταξύ τους. Θα μπορούσαμε να κοιτάξουμε 2 από τα χαρακτηριστικά και να δούμε αν υπάρχει κάποια συσχέτιση μεταξύ τους



Τα μήλα (blue) και τα πορτοκάλια (πορτοκαλί) καταλαμβάνουν διαφορετικούς χώρους στη φωτογραφία.

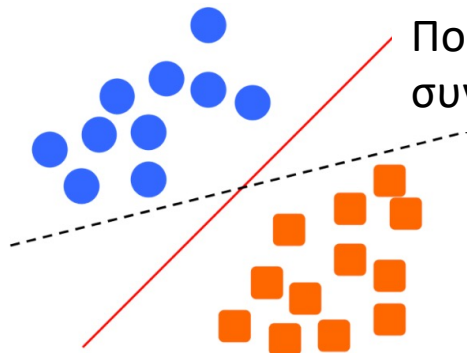
Το πρόβλημα της κατηγοροποίησης είναι πρόβλημα εύρεσης ενός συνόρου απόφασης (boundary decision), είτε ευθεία γραμμή ή καμπύλη.

Μηχανή με υποστηρικτικά διανύσματα μηχανής – **Support vector machine**

Ο αλγόριθμος support vector machine, είναι ένας ιδιαίτερα απλός και διαισθητικός αλγόριθμος ως προς το πως λαμβάνουμε μια απόφαση.

Με βάση το παρακάτω σχήμα, ποια συνοριακή γραμμή είναι καλύτερη? Η μαύρη διακεκομμένη ή η συμπαγής πορτοκαλί.

Με βάση το παρακάτω σχήμα, ποια συνοριακή γραμμή είναι καλύτερη?

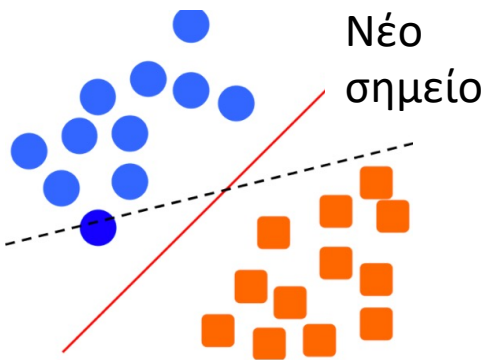


Ποια είναι η καλύτερη συνοριακή γραμμή?

Ο περισσότερος κόσμος θα επιλέξει την πορτοκαλί γραμμή γιατί είναι στο μέσο του διαστήματος μεταξύ των δύο δειγμάτων

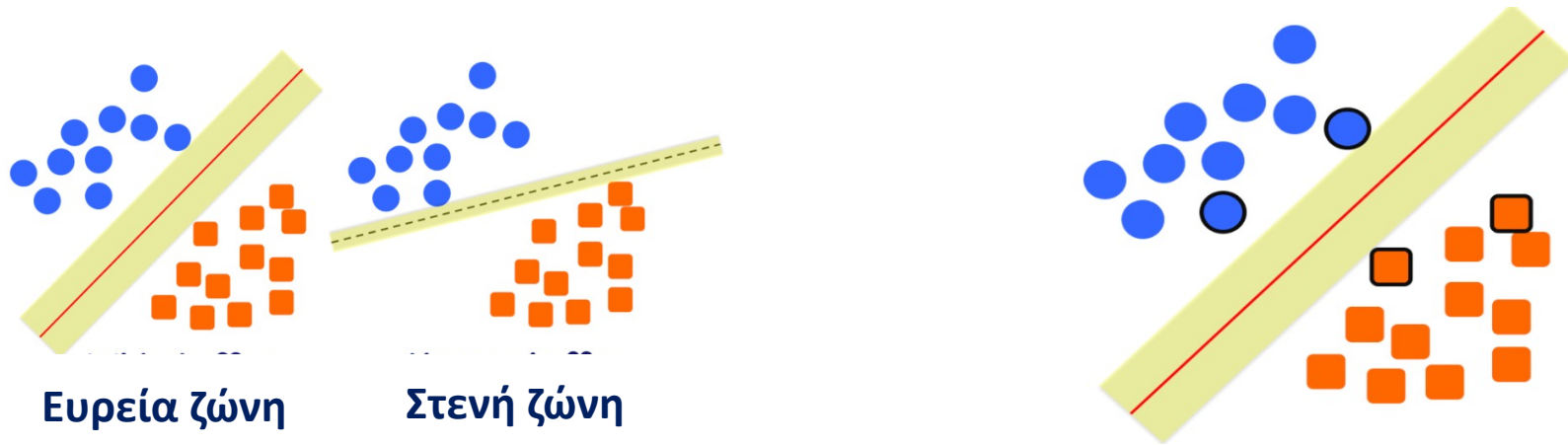
Αν είχαμε ένα νέο σημείο (blue), τότε η μαύρη διακεκομμένη γραμμή θα κάνει την λάθος απόφαση. Επομένως τα μοντέλα που έχουν το μέσο της απόστασης μεταξύ των δύο κλάσεων έχουν καλύτερη προσέγγιση στην απόφαση.

Η προσέγγιση πρέπει να μετατραπεί ώστε να αναγνωρίζεται από τον Η/Υ. Αυτός είναι σχεδιασμός του SVM αλγόριθμου. Πρώτα κατασκευάζει μια ζώνη από τη συνοριακή γραμμή στα πλησιέστερα στη γραμμή στοιχεία της κάθε κλάσης. Αυτά είναι τα υποστηρικτικά διανύσματα



Support vector machine

Το πρόβλημα του SVM αλγόριθμου καταλήγει στο να βρεθεί η μέγιστη ζώνη που ικανοποιεί την παραπάνω συνθήκη, δεδομένων όλων των διανυσμάτων υποστήριξης.



Η μαύρη διακεκομμένη γραμμή έχει πολύ στενή ζώνη, ενώ η κόκκινη διακεκομμένη έχει αρκετά ευρεία ζώνη. Με βάση την προηγούμενη συζήτηση, θα πρέπει να επιλέξουμε την κόκκινη ζώνη

Αν κάνουμε το γράφημα των διανυσμάτων υποστήριξης θα πάρουμε το σχήμα

Μηχανική μάθηση – Εύλογα ερωτήματα

- Πως επιλέγουμε τα χαρακτηριστικά των δεδομένων στα οποία θα στηριχθεί το μοντέλο;
- Πως επιλέγουμε ποιο μοντέλο θα χρησιμοποιήσουμε;
- Πως βελτιστοποιούμε το μοντέλο αυτό ώστε να έχουμε τα καλύτερα αποτελέσματα;
- Πως είμαι σίγουρος ότι δημιουργώ ένα γενικό μοντέλο το οποίο θα γενικεύσει τα αποτελέσματα της εκπαίδευσής του σε νέα δεδομένα;
- Μπορούμε να εκτιμήσουμε το πόσο καλά θα συμπεριφερθεί το μοντέλο πρόγνωσης σε νέα δεδομένα;

Οι ερωτήσεις αυτές περιπλέκουν την κατάσταση και δείχνουν πόσο σύνθετο είναι το πρόβλημα της μηχανικής μάθησης

Εφαρμογή ML με την βιβλιοθήκη **scikit-learn**

Θα συζητήσουμε πως μπορούμε να εφαρμόσουμε ML με την βιβλιοθήκη **scikit-learn**

Μπορείτε να την εγκαταστήσετε στο centos του laptop ή στο account σας στους Η/Υ του εργαστηρίου: **python3 -m pip install --user -U scikit-learn**

scikit-learn είναι ένα πολύ καλό λογισμικό για χρήση σε προβλήματα με ML γιατί:

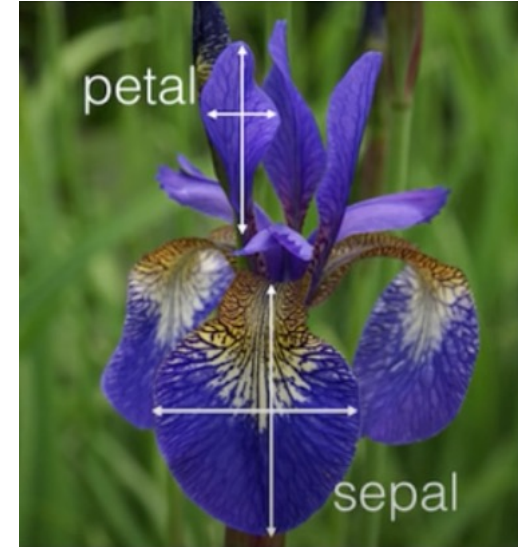
- Προσφέρεται σα λογισμικό διασύνδεσης με πολλά μοντέλα μηχανικής μάθησης
- Προσφέρει επίσης πολλές επιλογές για κάθε μοντέλο, οι οποίες μπορούν να ρυθμιστούν ανάλογα με το πρόβλημα. Ωστόσο υπάρχουν και αρχικοποιημένες τιμές που επιτρέπουν στο χρήστη να ξεκινήσει γρήγορα με την εφαρμογή που επιθυμεί
- Πολύ καλός και αναλυτικός οδηγός χρήσης
- Περιέχει πολλές μεθόδους για επιλογή και εκτίμηση της απόδοσης διαφόρων μοντέλων, αλλά και προετοιμαδία των δεδομένων

Παράδειγμα ML με scikit-learn: Το λουλούδι ίριδα (iris)

Το 1936 ο E. Andersen συνέλεξε δεδομένα για το λουλούδι ίριδα

Η συλλογή περιείχε:

- 50 διαφορετικά δείγματα από 3 είδη του λουλουδιού (συνολικά 150 δείγματα)
- Μετρήσεις του μήκους και πλάτους του πέταλου και του σέπαλου του λουλουδιού. 4 μετρήσεις/δείγμα



ίριδα
setosa



ίριδα
virginica



ίριδα
versicolor

Παράδειγμα ML με scikit-learn: Το λουλούδι ίριδα (iris)

```
>>> import numpy as np
>>> import itertools
>>> import matplotlib.pyplot as plt
>>> from sklearn import svm, datasets
>>> from sklearn.metrics import classification_report
>>>
>>> iris=datasets.load_iris()
>>> print(iris.feature_names)
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
>>> print(iris.data[:10])
[[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]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]]
```

- Τα δεδομένα είναι πάντοτε ένας 2-Δ πίνακας (**N_δείγματα**, **N_χαρακτηριστικά**) παρόλο που τα αρχικά δεδομένα μπορεί να είχαν διαφορετική μορφή
- Οι εντολές **print(iris.target_names)** και **print(set(iris.target))** τυπώνουν τα ονόματα των χαρακτηριστικών (target/features) ['setosa', 'versicolor', 'virginica'] των δεδομένων και το νούμερο της αναπαράστασής τους χρησιμοποιώντας {0,1,2}

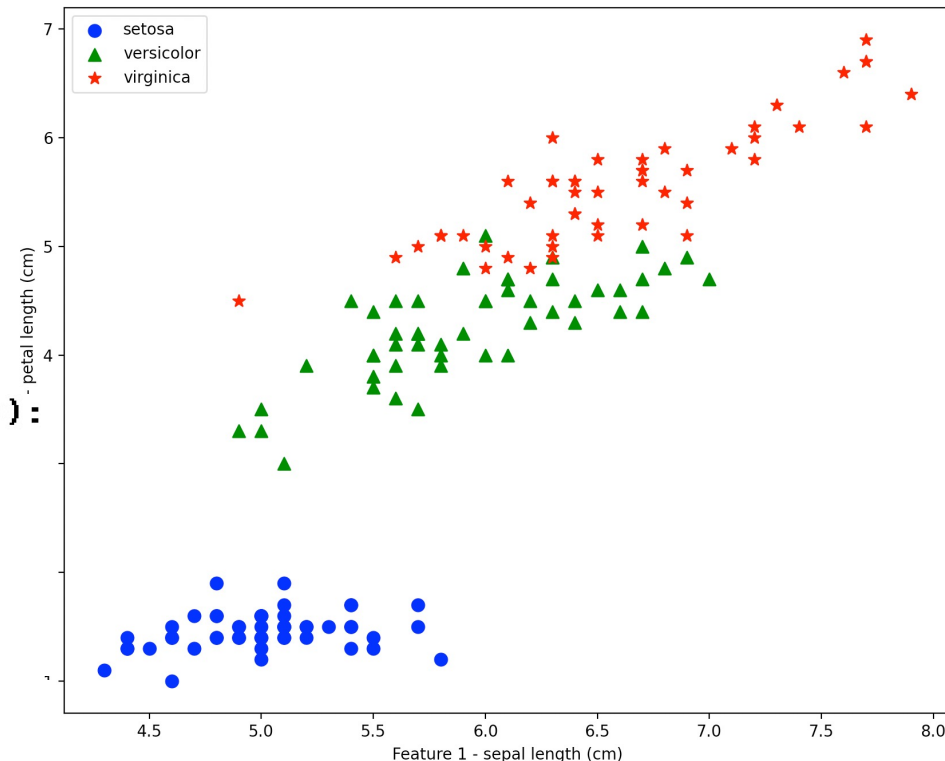
Πίνακας χαρακτηριστικών X και πίνακας target Y

- Για την προετοιμασία των δύο πινάκων εκτελούμε τα ακόλουθα:

```
>>> X = iris.data[:,[0,2]]                                # Χρήση 2 χαρακτηριστικών των δεδομένων
>>> Y = iris.target
>>> target_names = iris.target_names
>>> feature_names = iris.feature_names
>>> n_class = len(set(Y))
>>> print("We have %d classes in the data"%(n_class))
We have 3 classes in the data
```

- Μπορούμε να κάνουμε το γράφημα των δύο χαρακτηριστικών που επιλέξαμε για τις 3 κλάσεις δεδομένων.

```
colors = ['b','g','r']
symbols = ['o','^','*']
plt.figure(figsize=(10,8))
for i,c,s in (zip(range(n_class),colors,symbols)):
    ix = Y == i
    plt.scatter(X[:,0][ix], X[:,1][ix],\
                color = c, marker = s, s=60, \
                label = target_names[i])
plt.legend(loc=2,scatterpoints = 1)
plt.xlabel('Feature 1 - ' + feature_names[0])
plt.ylabel('Feature 2 - ' + feature_names[2])
plt.show()
```



- Παρατηρούμε ότι ακόμα και με τα 2 αυτά χαρακτηριστικά υπάρχει διαχωρισμός των κλάσεων.

SVM in scikit-learn

Η χρήση της εφαρμογής για SVM στο scikit-learn είναι απλή για τους περισσότερους αλγόριθμους

Εφαρμόζονται τα ακόλουθα βήματα:

- Αρχικοποίηση του μοντέλου:
- Εκπαίδευση του μοντέλου χρησιμοποιώντας μια συνάρτηση προσαρμογής
- Πρόβλεψη σε νέα δεδομένα χρησιμοποιώντας την συνάρτηση πρόβλεψης

```
>>> #Initialize the SVM classifier
>>> clf = svm.SVC(kernel='linear')
>>>
>>> #Train the classifier with data
>>> clf.fit(X,Y)
SVC(kernel='linear')    # Print της συνάρτησης προσαρμογής
```

Υπάρχουν πολλές παράμετροι στη μοντέλο που χρειάζεται να ρυθμιστούν (`help(clf)`)

```
SVC(*, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True,
probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False,
max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None)
```

Για το SVM μοντέλο, οι σημαντικότεροι παράγοντες είναι ο παράγοντας C και gamma

Οι παράμετροι πρέπει να εξετάζονται και να ρυθμίζονται ώστε ανάλογα με τα data η απόδοση του μοντέλου να είναι καλή

SVM in scikit-learn

Χρειάζεται να χωρίσουμε τα data σε δύο μέρη: το μέρος που χρησιμοποιείται για εκπαίδευση και αυτό για τον έλεγχο των αποτελεσμάτων. Το τμήμα αυτό δεν χρησιμοποιείται στην εκπαίδευση του αλγόριθμου.

Στην προκειμένη περίπτωση ελέγχουμε τα αποτελέσματα στο τμήμα που χρησιμοποιήσαμε στο training

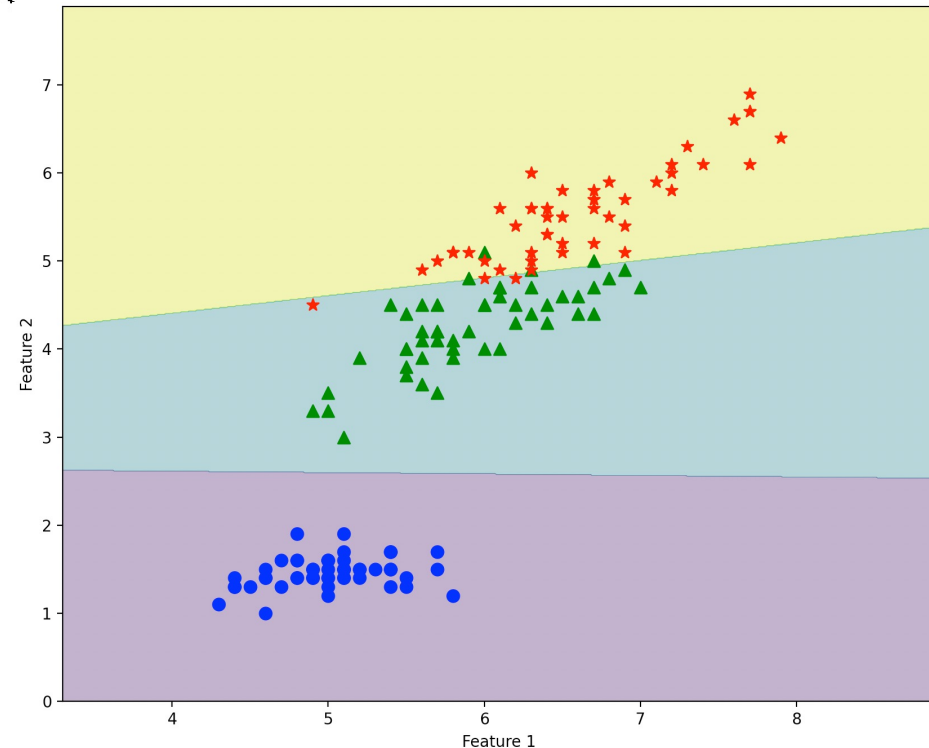
```
>>> clf.predict(X)
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

Μπορούμε να κάνουμε το γράφημα του συνόρου της απόφασης για το μοντέλο.

Γράφημα του συνόρου της απόφασης

```
>>> def plot_decision_boundary(X,Y,clf,title=None):
...     x_min,x_max = X[:,0].min()-1, X[:,0].max()+1
...     y_min,y_max = X[:,1].min()-1, X[:,1].max()+1
...     xx,yy =np.meshgrid(np.arange(x_min, x_max, 0.01),
...                         np.arange(y_min, y_max, 0.01))
...     Z = clf.predict(np.c_[xx.ravel(),yy.ravel()])
...     Z = Z.reshape(xx.shape)
...     plt.figure(figsize=(10,8))
...     plt.contourf(xx,yy,Z,alpha=0.4)
...     for i,c,s in (zip(range(n_class),colors,symbols)):
...         ix=Y==i
...         plt.scatter(X[:,0][ix], X[:,1][ix], \
...                     color=c, marker=s, s=60, \
...                     label = target_names[i])
...     if title is None:
...         plt.title(title)
...     plt.xlabel('Feature 1')
...     plt.ylabel('Feature 2')
...     plt.show()
>>> plot_decision_boundary(X,Y,clf)
```

Καλός διαχωρισμός μεταξύ των 3 κλάσεων



Παλινδρόμηση - Regression

Regression αποτελεί μια ομάδα αλγορίθμων που χρησιμοποιείται σε supervised learning όταν το αποτέλεσμα είναι ποσότητα αριθμών και όχι κατηγοροποιημένα data.

Μια απλή μορφή regression είναι η μέθοδος χ^2 που χρησιμοποιείται στο εργαστήριο και όπου χρησιμοποιούνται τα data και συγκρίνονται με θεωρητικό μοντέλο.

Στη μηχανική μάθηση ωστόσο η κατάσταση είναι πιο ρευστή και μπορούμε να προσαρμόσουμε οποιαδήποτε συναρτήσεις δεδομένων χωρίς να ξέρουμε την πραγματική αναλυτική μορφή υπεύθυνη για τα δεδομένα

Υπάρχουν διάφοροι τρόποι για να το κάνουμε αυτό, τυχαίο δάσος (random forest), τεχνικά νευρωνικά δίκτυα (artificial neural networks), supported vector machine κλπ

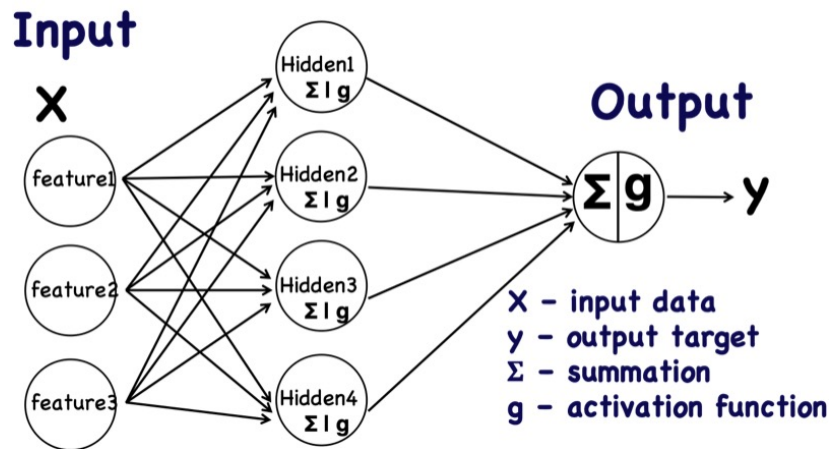
Νευρωνικά Δίκτυα

Η μέθοδος των νευρωνικών δικτύων αναπτύχθηκε για να προσομοιώση τον τρόπο που λειτουργεί ο ανθρώπινος εγκέφαλος.

Η παρακάτω εικόνα δείχνει μια συνηθισμένη δομή ενός νευρωνικού δικτύου.

Αποτελείται από 3 στρώματα, το στρώμα εισόδου (**input layer**), το κρυμμένο στρώμα (**hidden layer**) και το στρώμα εξόδου (**output layer**). Μπορούμε να έχουμε πολλά ενδιάμεσα στρώματα

Κάθε στρώμα αποτελείται από νευρώνες (οι κύκλοι στο σχήμα)



Στο συγκεκριμένο παράδειγμα, το στρώμα εισόδου έχει 3 νευρώνες, έναν για κάθε χαρακτηριστικό των data

Το ενδιάμεσο στρώμα αποτελείται από πολλούς νευρώνες και αποτελούν την βασική μονάδα επεξεργασίας του δικτύου

Κάθε νευρώνας εκτελεί δύο διαδικασίες (α) αθροίζει την πληροφορία που του εισάχθηκε από το προηγούμενο στρώμα και (β) περνά την συνολική πληροφορία σε μια συνάρτηση δραστηριοποίησης (**activation function**) για να αποφασιστεί αν δώσει κάποιο σήμα

Το στρώμα εξόδου αποτελείται από νευρώνες, για κατηγοροποίηση το output είναι ένας αριθμός μεταξύ 0 και 1, ενώ για προβλήματα regression μπορεί να είναι οποιαδήποτε αριθμός

Νευρωνικά Δίκτυα

Τα βέλη που συνδέουν τους νευρώνες από στρώμα σε στρώμα ονομάζονται σύνδεσμοι (**links**) και είναι υπεύθυνοι για να περάσει η πληροφορία από στρώμα σε στρώμα

Σε κάθε σύνδεσμο αντιστοιχεί ένας αριθμός (βάρος) που αποτελεί κάποια από τις ρυθμιζόμενες παραμέτρους του μοντέλου.

Όταν η πληροφορία εισάγεται από το στρώμα εισόδου, μέσω των συνδέσμων στο κρυμμένο στρώμα για επεξεργασία και κατόπιν στο στρώμα εξόδου για τη δημιουργία του αποτελέσματος, η διεργασία ονομάζεται διάδοση προς τα εμπρός (**forward propagation**)

Η εκπαίδευση του νευρωνικού δικτύου είναι να ρυθμιστούν τα βάρη που αντιστοιχούν στους συνδέσμους

Συνήθως αυτό γίνεται χρησιμοποιώντας μια συνάρτηση η οποία υπολογίζει τα σφάλματα μεταξύ των εκτιμήσεων του μοντέλου και των πραγματικών στόχων

Μπορούμε να υπολογίσουμε τα σφάλματα αρχικά σε μια forward propagation και να επιστρέψουμε το σφάλμα πίσω στο στρώμα εισόδου

Ανάλογα με τις τιμές των βαρών σε κάθε σύνδεσμο και τη συνεισφορά του στο συνολικό σφάλμα, μπορούμε να ρυθμίσουμε τη τιμή του βάρους ώστε την επόμενη φορά που θα περάσει πληροφορία να μειωθεί η συνεισφορά στο σφάλμα. Αυτό λέγεται **backpropagation**

Μπορούμε να περάσουμε την πληροφορία προς τα εμπρός ή προς τα πίσω πολλές φορές σε μια διεργασία που ονομάζεται εποχή (**epoch**) ανανεώνοντας τις τιμές των βαρών των συνδέσμων ώστε να ελαττωθεί το σφάλμα και να υπάρχει σταθερότητα.

Υπάρχουν διάφοροι αλγόριθμοι που κάνουν την αλλαγή των βαρών

Νευρωνικά Δίκτυα - Παράδειγμα

Θα δημιουργήσουμε ένα δείγμα δεδομένων το οποίο αποτελείται από 2 ημιτονοειδή κύματα και προσθέτουμε επίσης κάποιο θόρυβο.

```
#!/usr/bin/python3
```

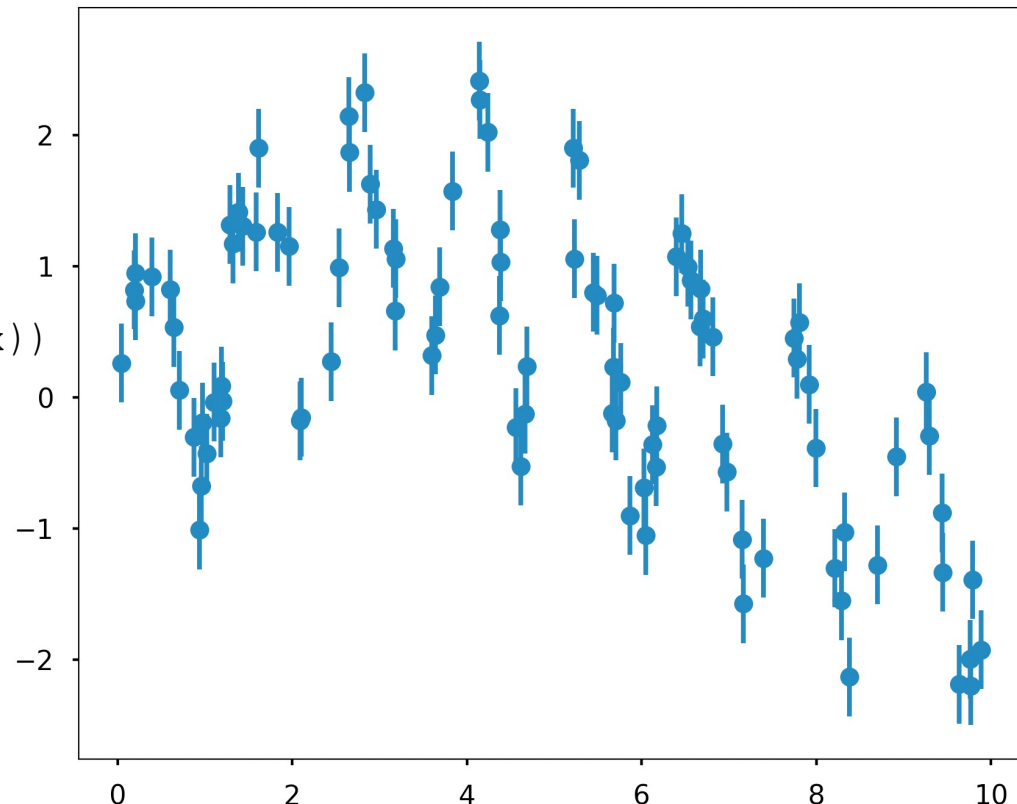
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
```

```
np.random.seed(12345)
```

```
x = 10 * np.random.rand(100)
```

```
def model(x,sigma=0.3):
    fast_osci = np.sin(5*x)
    slow_osci = np.sin(0.5*x)
    noise=sigma*np.random.randn(len(x))
    return slow_osci+fast_osci+noise
```

```
plt.figure(figsize=(10,8))
y = model(x)
plt.errorbar(x,y, 0.3, fmt='o')
plt.show()
```



Νευρωνικά Δίκτυα - Παράδειγμα

Μπορούμε να χρησιμοποιήσουμε το νευρωνικό δίκτυο για να προσαρμόσουμε τα data.

Κάθε σημείο στο x θα αποτελεί χαρακτηριστικό εισαγωγής του ANN, και το αποτέλεσμα θα είναι μια ακολουθία y .

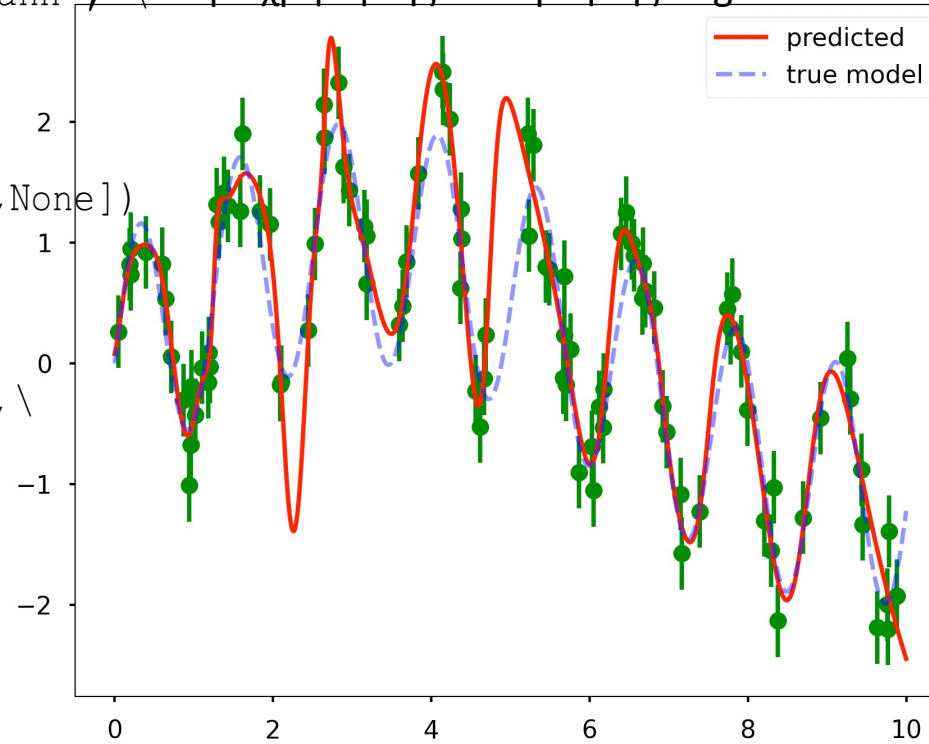
Στη βιβλιοθήκη scikit-learn το μοντέλο ANN για regression είναι το **MLPRegressor**, που σημαίνει **Multi Layer Perceptron Regressor**.

```
from sklearn.neural_network import MLPRegressor
mlp = MLPRegressor(hidden_layer_sizes=(200,200,200), \
                    max_iter=2000, solver='lbfgs', \
                    alpha=0.01, activation='tanh', \
                    random_state = 8)

xfit = np.linspace(0,10,1000)
ytrue = model(xfit,0.3)
yfit = mlp.fit(x[:, None], y).predict(xfit[:,None])

plt.figure(figsize=(10,8))
plt.errorbar(x, y, 0.3, fmt='o')
plt.plot(xfit, yfit, '-r', label='predicted', \
        zorder = 10)
plt.plot(xfit, ytrue, '-k', alpha=0.5, \
        label = 'true model', zorder = 10)
plt.legend()
plt.show()
```

Χρήση 3 κρυφών στρωμάτων με 200 νευρώνες το κάθε στρώμα. 2000 επαναλήψεις για ελαχιστοποίηση με χρήση της συνάρτησης lbfgs



Συστοιχίες (**clustering**)

Η δημιουργία συστοιχιών αποτελεί μια ομάδα αλγορίθμων που χρησιμοποιούνται κυρίως σε unsupervised learning. Είναι επομένως χρήσιμοι όταν δεν έχουμε ετικέτες για τα data και οι αλγόριθμοι θα προσπαθήσουν να βρουν τις διατάξεις (**patterns**) της εσωτερικής δομής ή ομοιότητες ανάμεσα στα data για να ομαδοποιηθούν.

Από τη στιγμή που δεν υπάρχουν ετικέτες (πραγματική απάντηση) δεν μπορούμε να χρησιμοποιήσουμε την πληροφορία αυτή για να περιορίσουμε το πρόβλημα.

Υπάρχουν ωστόσο άλλοι τρόποι που μπορούμε να ακολουθήσουμε για να λύσουμε το συγκεκριμένο είδος προβλήματος. Ένας ιδιαίτερα δημοφιλής αλγόριθμος είναι αυτός του **K-μέσου (K-means)**.

Αλγόριθμος K-μέσου (K-means)

Είναι από τους δημοφιλέστερους αλγόριθμους γιατί την απλότητα και αποτελεσματικότητά του

Η ιδέα του αλγόριθμου στηρίζεται στο γεγονός ότι αν έχεις 2 σημεία τα οποία είναι κοντά το ένα στο άλλο σε σχέση με τα υπόλοιπα σημεία, τότε θα είναι παρόμοια.

Στηρίζεται επομένως στην απόσταση των δύο σημείων. Για πρόβλημα δύο διαστάσεων (μπορεί να είναι και περισσότερες διαστάσεις) ο K-means αλγόριθμος χρησιμοποιεί τον Ευκλείδειο ορισμό της απόστασης για να υπολογίσει την απόσταση δύο σημείων 1 και 2:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

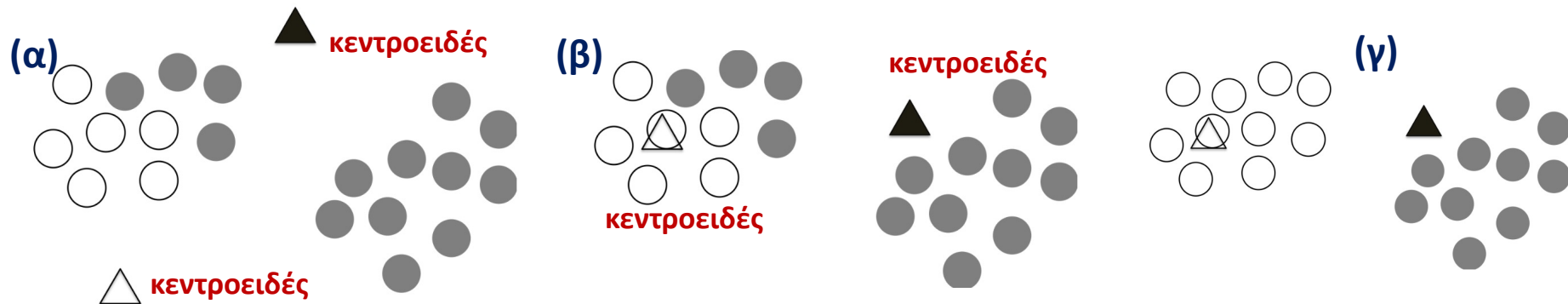
Το K- στο όνομα του αλγόριθμου σημαίνει ότι θα υπάρχουν K-συστοιχίες (ομαδοποιήσεις). K-means είναι ένας επαναληπτικός αλγόριθμος ο οποίος ανανεώνει τη θέση των κεντροειδών των συστοιχιών έως ότου βρεθεί η καλύτερη επιλογή.

Εύρεση των κεντροειδών

Αρχικά επιλέγουμε τυχαία K-σημεία ως τα κεντροειδή K-συστοιχιών που μπορεί να εμφανίζονται στα data.

Δεν γνωρίζουμε ποια σημεία ανήκουν σε ποιο cluster. Θεωρούμε στην περίπτωση εδώ ότι τα data περιγράφονται από 2 clusters με τυχαίες θέσεις για τα κεντροειδή τους.

Στο δεύτερο βήμα αρχίζουμε να αναθέτουμε σημεία των δεδομένων στους clusters με βάση τη μικρότερη απόσταση του κάθε σημείου από τα κεντροειδή. Το κάνουμε αυτό για όλα σημείων των data όπως φαίνεται στο σχήμα (α).



Με βάση την ανάθεση σημείων στους 2 clusters από το δεύτερο βήμα, μπορούμε να βρούμε το νέο κεντροειδές του κάθε cluster. Το σχήμα (β) πιο πάνω δείχνει τα νέα κεντροειδή

Με τα νέα κεντροειδή επαναλαμβάνουμε την διαδικασία ανάθεσης σημείων και επαναπροσδιορίζουμε τα κεντροειδή όπως φαίνεται στο σχήμα (γ).

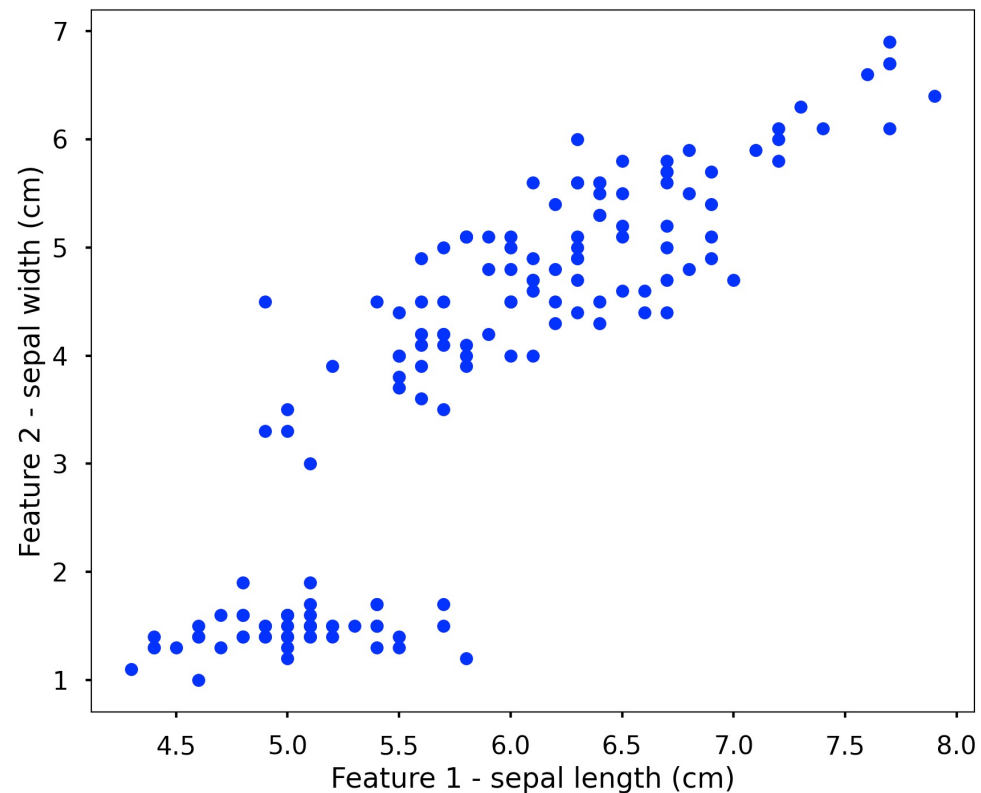
Η διαδικασία επαναλαμβάνεται έως ότου δεν υπάρχουν αλλαγές στα κεντροειδή των clusters και δεν υπάρχουν αλλαγές στα σημεία που ανήκουν στους clusters

Εφαρμογή του αλγορίθμου K-means στην python

Χρησιμοποιούμε τα δεδομένα για τα λουλούδια ίριδα, και αυτή τη φορά αγνοούμε τις ετικέτες οπότε το πρόβλημα ανάγεται σε πρόβλημα unsupervised learning.

Κάνουμε τα γραφήματα των δεδομένων ως να μην υπάρχουν ετικέτες

```
#!/usr/bin/python3
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
plt.style.use('seaborn-poster')
#import ta data tou louloudiou iris
iris = datasets.load_iris()
#estw uparxoun duo xaraktiristika kai
#etsi mporoume na ta optikopoiisoume
#Xrisimopoioume ta miki twn petals kai sepals
X = iris.data[:, [0,2]]
Y = iris.target
target_names = iris.target_names
feature_names = iris.feature_names
# oi klaseis
n_class = len(set(Y))
# Anaparastasi twn data
plt.figure(figsize=(10,8))
plt.scatter(X[:, 0], X[:, 1], \
            color = 'b', marker='o',s=60)
plt.xlabel('Feature 1 - ' + feature_names[0])
plt.ylabel('Feature 2 - ' + feature_names[1])
plt.show()
```

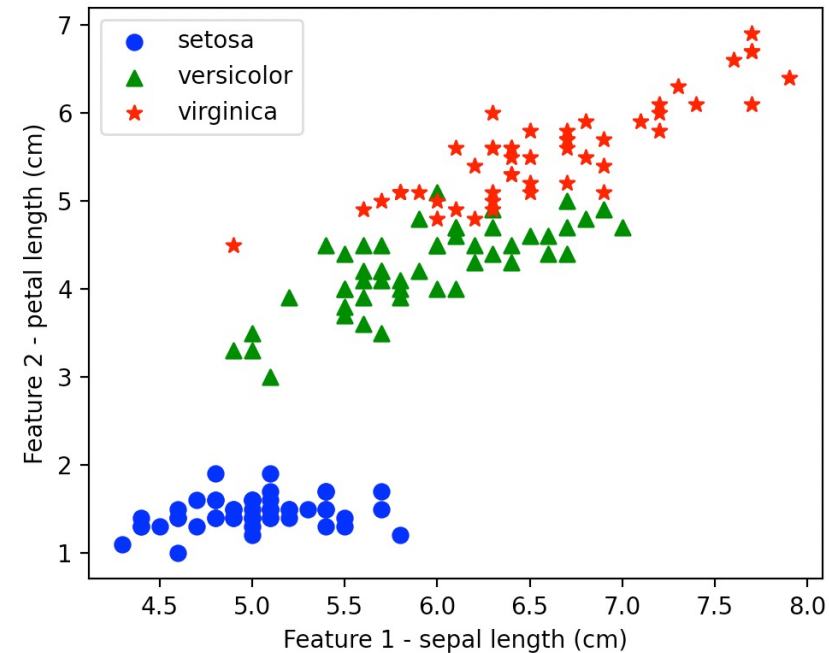
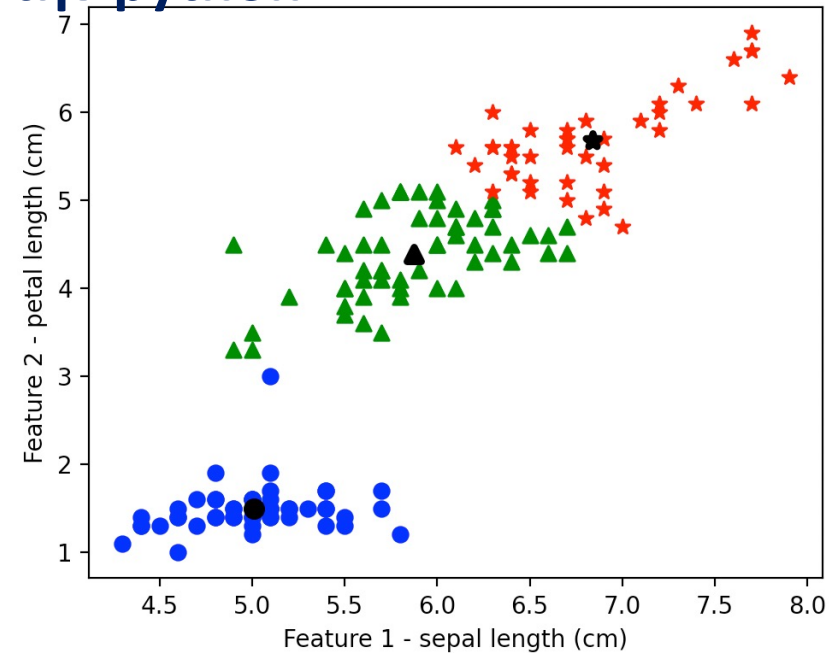


Εφαρμογή του αλγορίθμου K-means στην python

```

from sklearn.cluster import KMeans
kmean = KMeans(n_clusters=3, random_state=0)
kmean.fit(X)
colors=['b','g','r']
symbols=['o','^','*']
fig = plt.figure(figsize=(8,5))
ax = fig.add_subplot(121)
ax2 = fig.add_subplot(122)
for i,c,s in (zip(range(n_class),colors,symbols)):
    #Ta clusters pou vrethikan apothikeyontai sta labels
    ix = kmean.labels_ == i
    ax.scatter(X[:,0][ix], X[:,1][ix],\
               color=c, marker=s, s=40,\
               label=target_names[i])
    #Ta kentroeidi apothikeyontai sta cluster_centers
    loc = kmean.cluster_centers_[i]
    ax.scatter(loc[0], loc[1], color='k',\
               marker=s,linewidth=3)
    ix = Y == i
    ax2.scatter(X[:,0][ix], X[:,1][ix],\
               color=c, marker=s, s=40,\
               label=target_names[i])
plt.legend(loc=2,scatterpoints=1)
ax.set_xlabel('Feature 1 - ' + feature_names[0])
ax.set_ylabel('Feature 2 - ' + feature_names[2])
ax2.set_xlabel('Feature 1 - ' + feature_names[0])
ax2.set_ylabel('Feature 2 - ' + feature_names[2])
plt.tight_layout()
plt.show()

```



Εφαρμογή του αλγορίθμου K-means στην python

Από τα γραφήματα της προηγούμενης σελίδας, βλέπουμε ότι τα αποτελέσματα είναι αρκετά καλά και είναι αρκετά παρόμοια με τις πραγματικές κλάσεις που υπάρχουν στα δεδομένα.

Θυμηθείτε ότι τα αποτελέσματα αυτά, τα πήραμε αγνοώντας τις ετικέτες και βασιζόμενοι στην ομοιότητα μεταξύ των δεδομένων.

Μπορούμε να προβλέψουμε νέα σημεία από δεδομένα, σε ποιους clusters ανήκουν χρησιμοποιώντας την συνάρτηση **predict**

Για παράδειγμα το ακόλουθο τμήμα κώδικα προβλέπει τον cluster στον οποίο ανήκουν δύο νέα σημεία

```
new_points = np.array([[5, 2], [6, 5]])  
kmean.predict(new_points)
```

Output: array([0, 1], dtype=int32)

Επομένως στα clusters 0 και 1