

Τεχνικές Προβλέψεων

Machine Learning



Robotics



Computer vision



Image recognition



Virtual agents

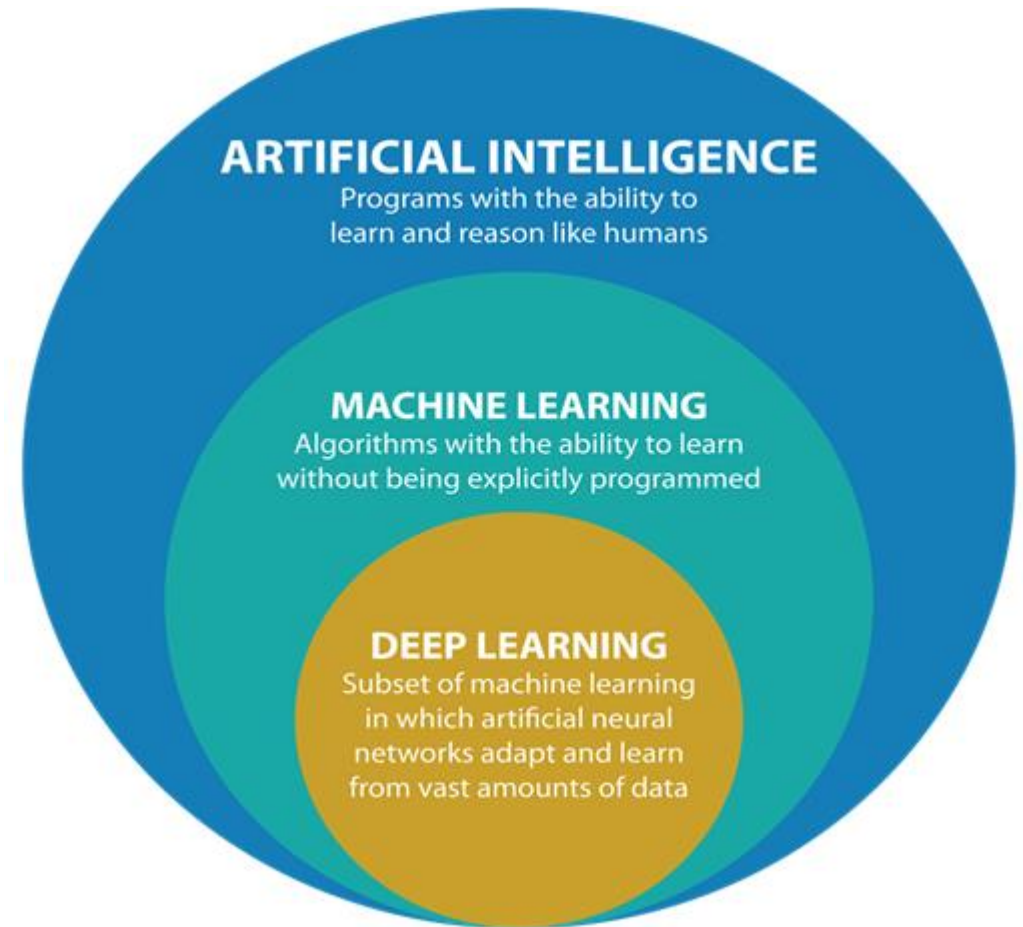


Autonomous vehicles



Games

- ✓ **Τεχνητή Νοημοσύνη (Artificial Intelligence - AI)** είναι η επιστήμη που έχει σαν στόχο την ανάπτυξη ευφυών συστημάτων τα οποία θα είναι ικανά να σκέφτονται και να ενεργούν σαν τον άνθρωπο.
- ✓ Η **Μηχανική Μάθηση (Machine Learning - ML)** είναι ένα πεδίο της AI και ασχολείται με την ανάπτυξη μεθόδων που παρέχουν την δυνατότητα σε συστήματα να μαθαίνουν αυτόνομα, βάσει των διαθέσιμων δεδομένων.
- ✓ Η **Βαθιά Μάθηση (Deep Learning - DL)** είναι ένα πεδίο της μηχανικής μάθησης που χρησιμοποιεί σύνθετα Νευρωνικά Δίκτυα (Neural Networks - NN) για να προσομοιώσει την δομή και λειτουργία του εγκεφάλου.



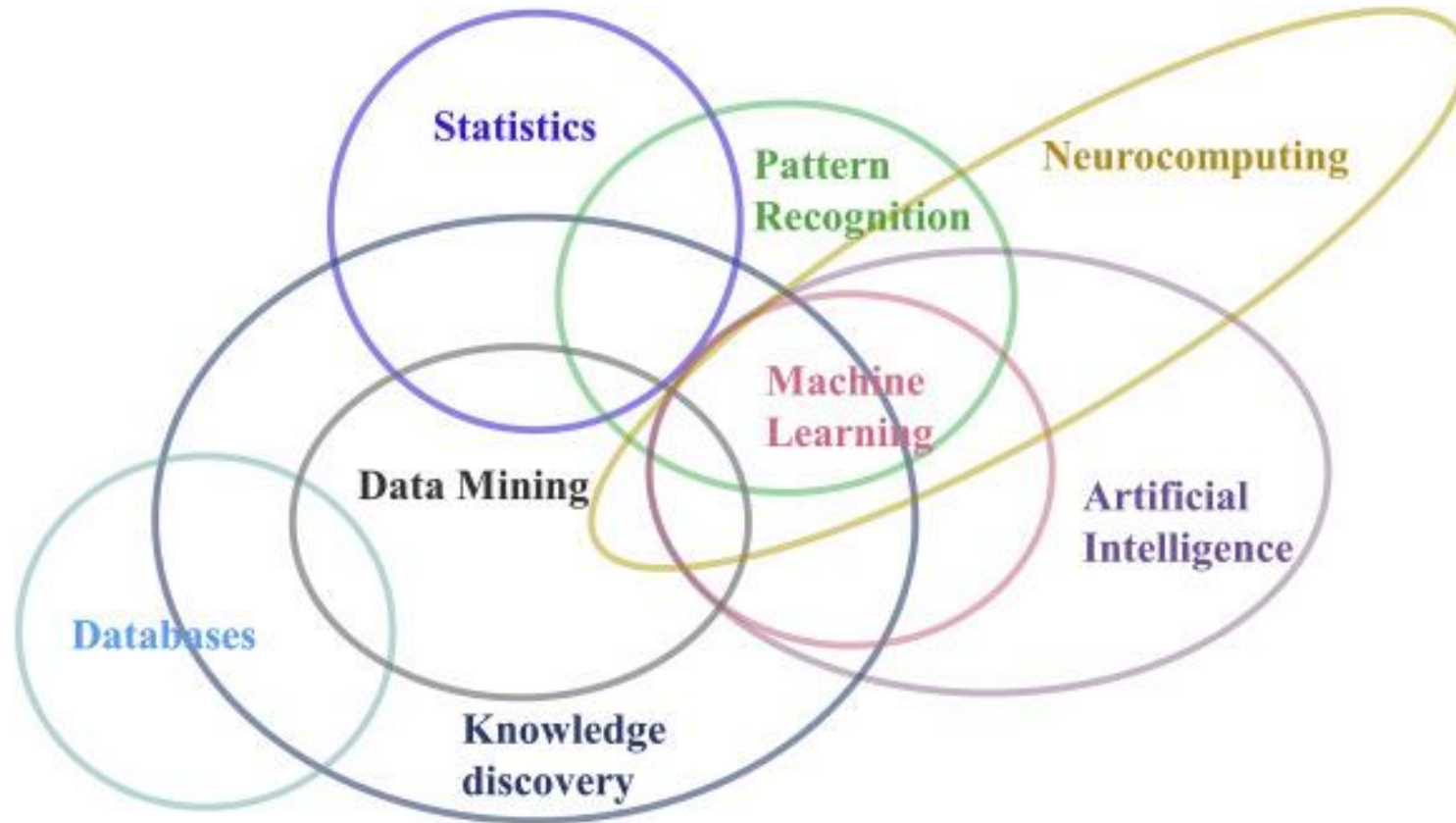
Πλεονεκτήματα & Μειονεκτήματα

- + Αναβάθμιση του ρόλου που έχουν τα διαθέσιμα δεδομένα
 - + Αναγνώριση νέων προτύπων και συσχετίσεων
 - + Αυτοματοποίηση διαδικασιών
-
- Υψηλές απαιτήσεις σε δεδομένα
 - Σχετικά υψηλές απαιτήσεις σε υπολογιστικούς πόρους
 - Δυσκολία στην ερμηνεία των αποτελεσμάτων (Black Boxes)

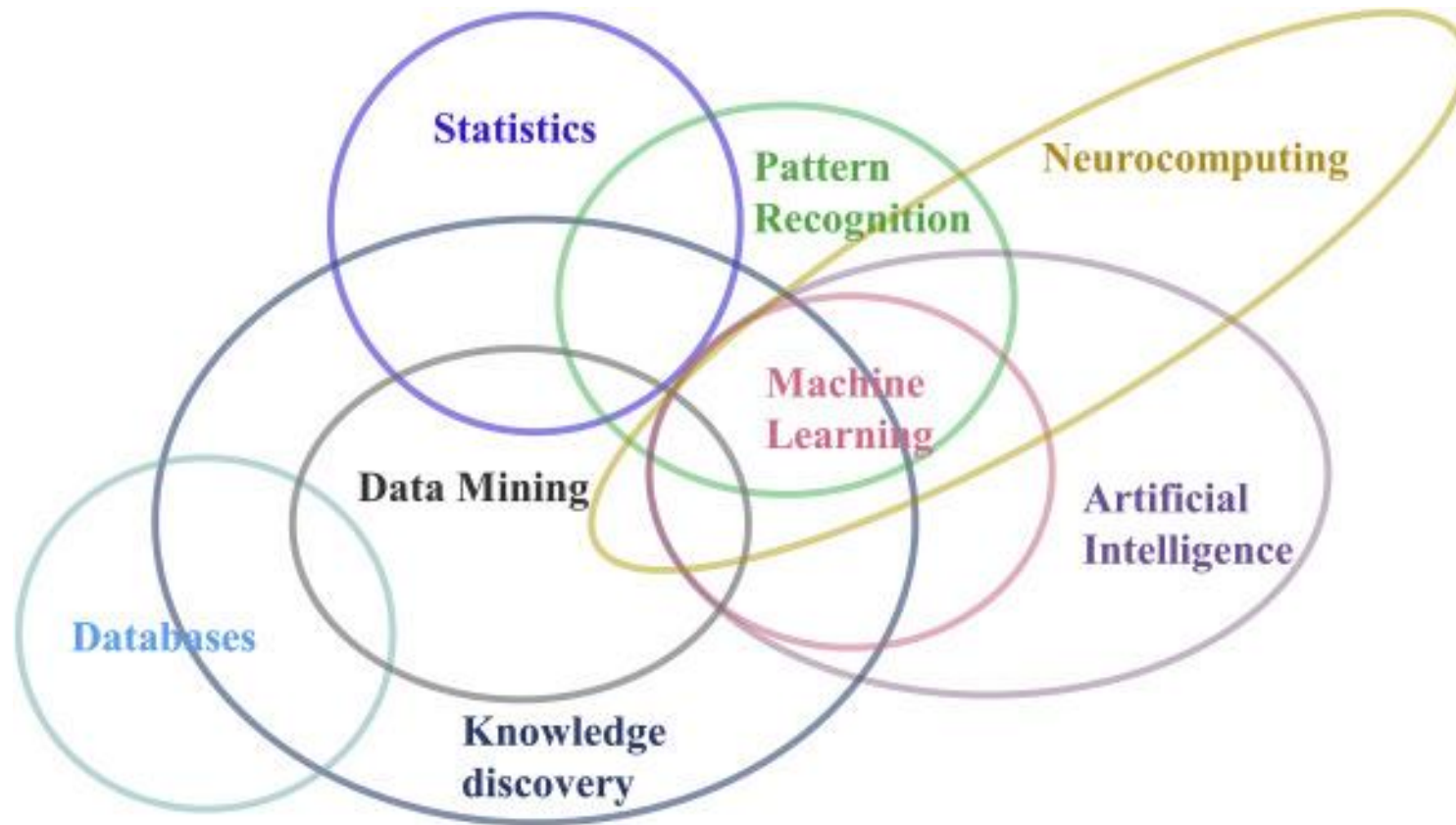


"Does your car have any idea why my car pulled it over?"

Data Science

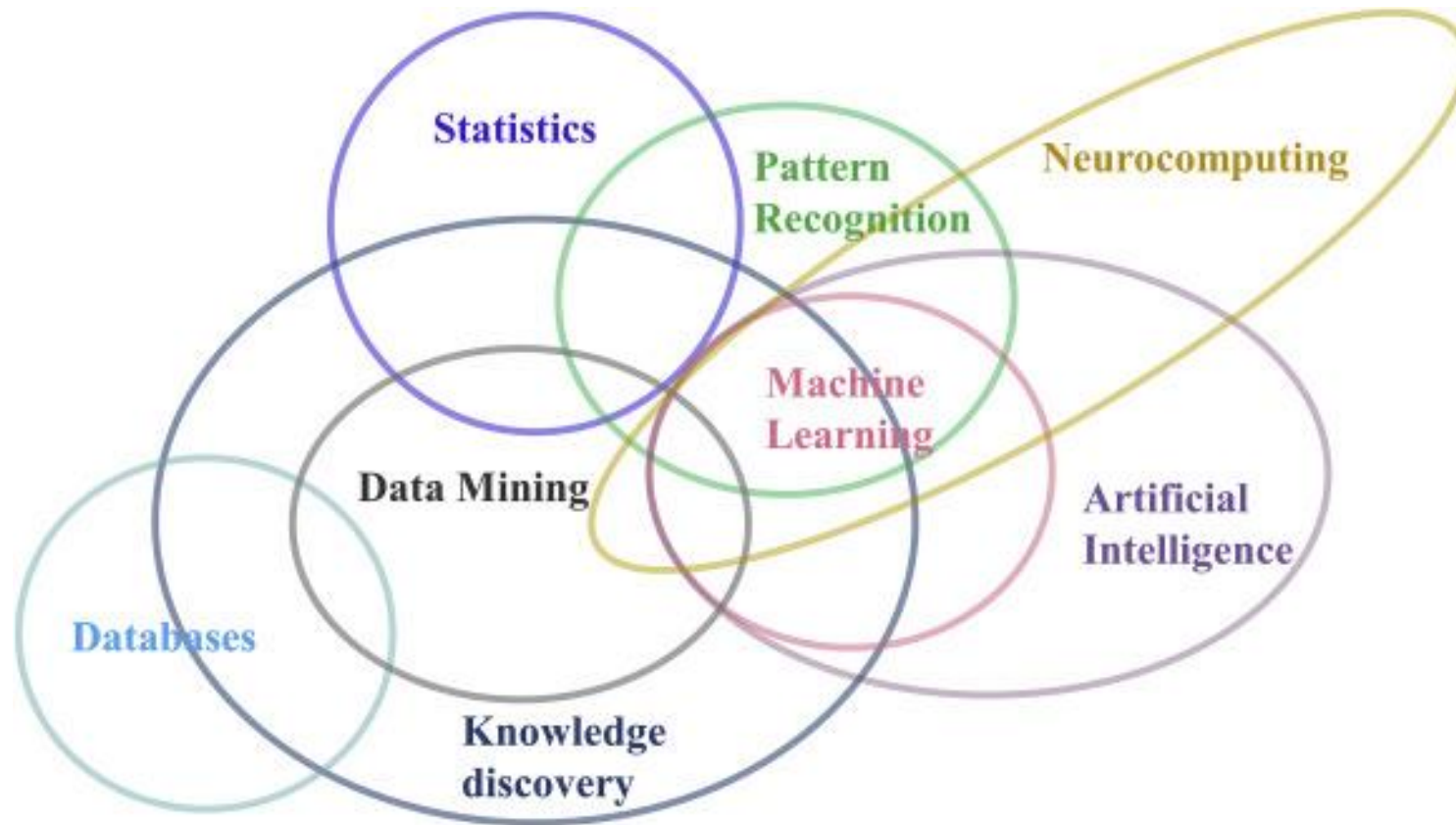


Data Science



- ✓ **Data Mining:** Η διαδικασία συλλογής, οργάνωσης και κατηγοριοποίησης πληροφοριών από μεγάλα σετ δεδομένων.
- ✓ **Knowledge Discovery:** Αποτελεί επέκταση της διαδικασίας εξόρυξης και στοχεύει κυρίως στην εκκαθάριση δεδομένων, τον συνδυασμό δεδομένων από διαφορετικές πηγές καθώς και στην κατανόησή τους.
- ✓ **Databases:** Υποδομές για την αποθήκευση και τον διαμοιρασμό δεδομένων στους ενδιαφερόμενους χρήστες

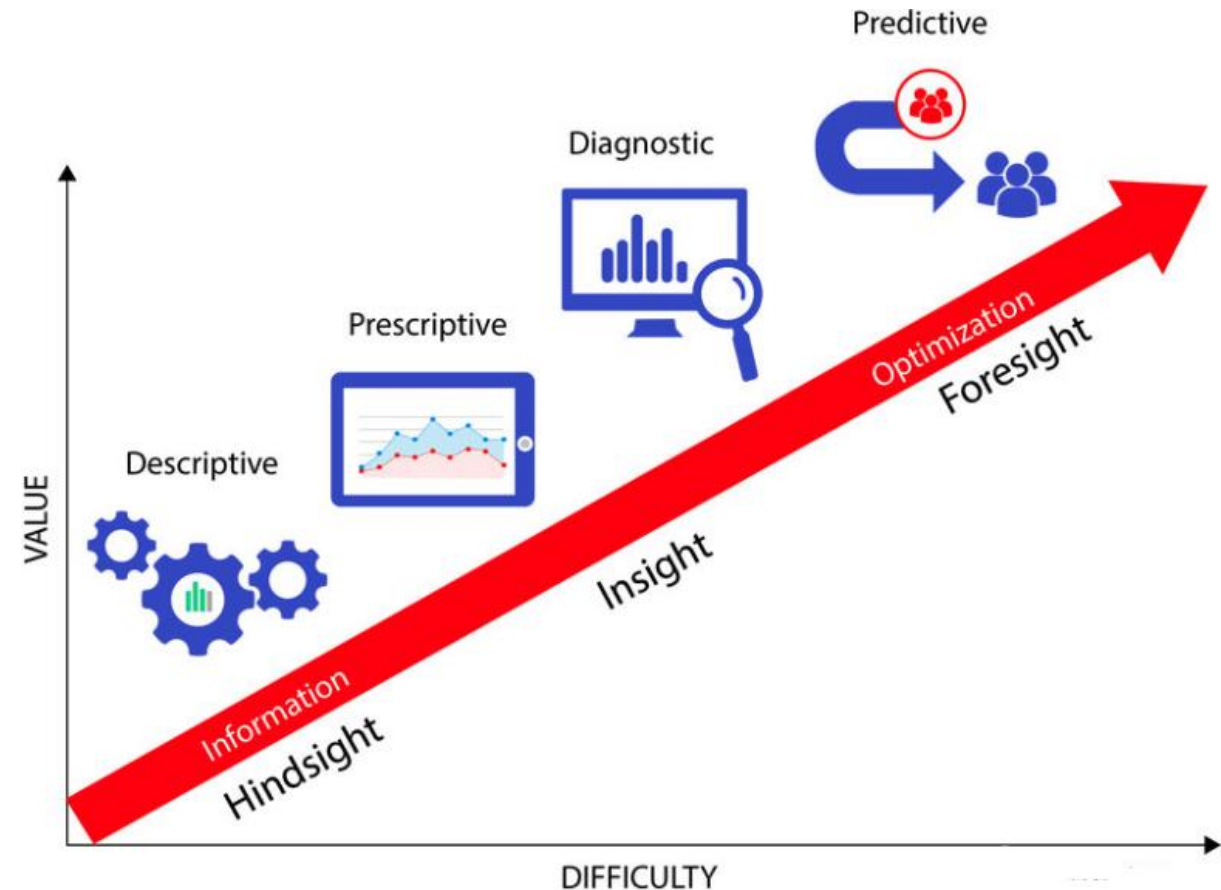
Data Science



- ✓ **Statistics:** Μοντελοποίηση και ανάλυση των δεδομένων με στατιστικά μοντέλα και δείκτες.
- ✓ **Machine Learning:** Μοντελοποίηση και ανάλυση των δεδομένων με χρήση μεθόδων ML
- ✓ **Pattern Recognition:** Αναγνώριση προτύπων στα δεδομένα. Η διαδικασία αυτή μπορεί να γίνει είτε με στατιστικά μοντέλα είτε με μοντέλα μηχανικής μάθησης.
- ✓ **Artificial Intelligence:** Ανάπτυξη ολοκληρωμένων λύσεων οι οποίες βασίζονται σε τεχνικές ML για την εκπαίδευσή τους

Business Intelligence and Analytics

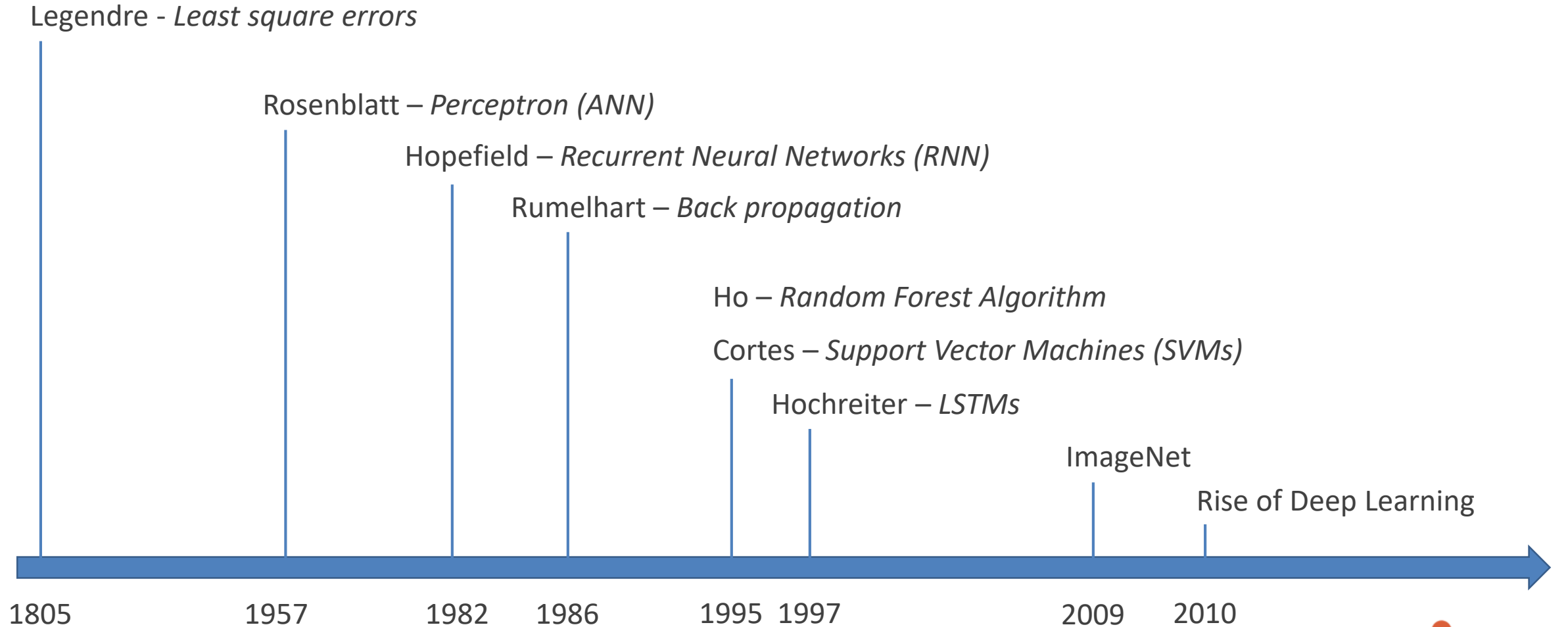
- ✓ Η εισαγωγή των **μεθόδων ανάλυσης δεδομένων** (data analytics) σε επιχειρήσεις και οργανισμούς είναι κρίσιμη για την βελτιστοποίηση των διαδικασιών τους
- ✓ Εργαλεία ανάλυσης δεδομένων μπορούν να έχουν εφαρμογή σε διαφορετικά τμήματα και προβλήματα μέσα στον οργανισμό
 - Ανάλυση ρίσκου στρατηγικών
 - Καθορισμός τιμολογιακής πολιτικής
 - Καθορισμός στρατηγικών marketing
 - Πρόβλεψη πωλήσεων προϊόντων
 - Βελτιστοποίηση αλυσίδας ανεφοδιασμού
 - Διαχείριση ανθρωπίνου δυναμικού



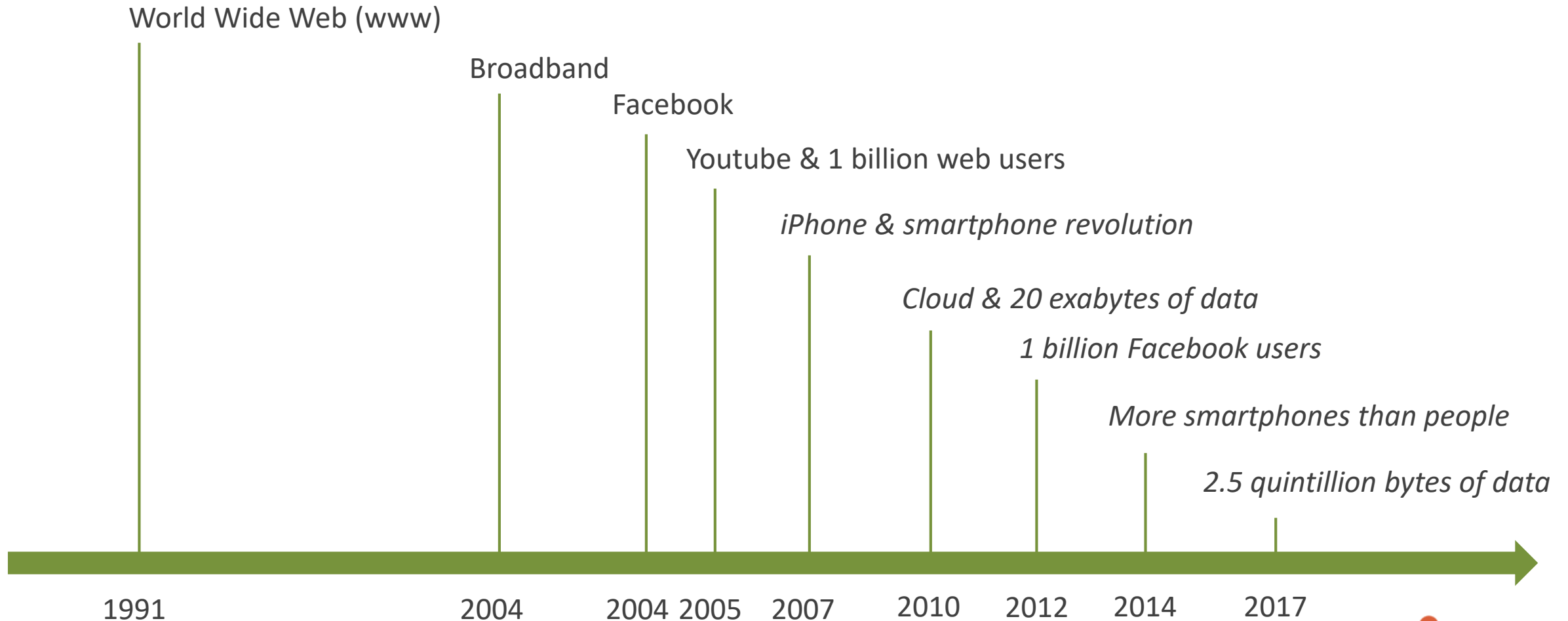
Machine Learning

- ✓ Δυνατότητα **αυτόνομης εκμάθησης** σε συστήματα.
- ✓ Ανάπτυξη μοντέλων τα οποία είναι σε θέση να αναλύσουν και να προσαρμοστούν σε **μεγάλες ποσότητες δεδομένων**.
- ✓ Δυνατότητα αναγνώρισης σύνθετων, **μη γραμμικών σχέσεων** και μοτίβων στα δεδομένα, χωρίς αυτά να είναι γνωστά πριν την εκπαίδευση.
- ✓ Δυνατότητα προσαρμογής και **ενίσχυσης της εκπαίδευσης** των μοντέλων καθώς νέα δεδομένα γίνονται διαθέσιμα.
- ✓ Οι **δυνατότητες των μεθόδων ML ήταν γνωστές** για πολλά χρόνια.
- ✓ Οι πρώτες θεωρητικές προσεγγίσεις έχουν τις **ρίζες τους στον 19^ο αιώνα**.
- ✓ Η αλγόριθμοι ML έγιναν **δημοφιλείς μετά το 1990**.
- ✓ Παράγοντες που συνέβαλλαν στην ανάπτυξη εφαρμογών ML
 - Εξελίξεις στην **θεωρία των αλγορίθμων ML**
 - Ευκολότερη πρόσβαση σε **μεγάλες ποσότητες δεδομένων**
 - Περισσότερη **διαθέσιμη υπολογιστική ισχύς**

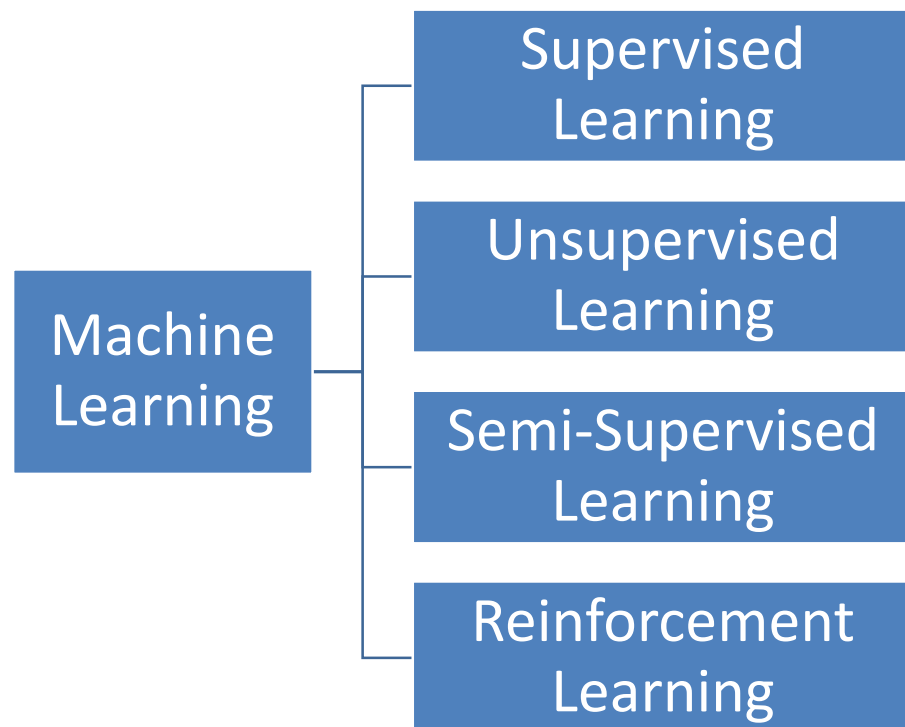
ML Algorithms - Advances



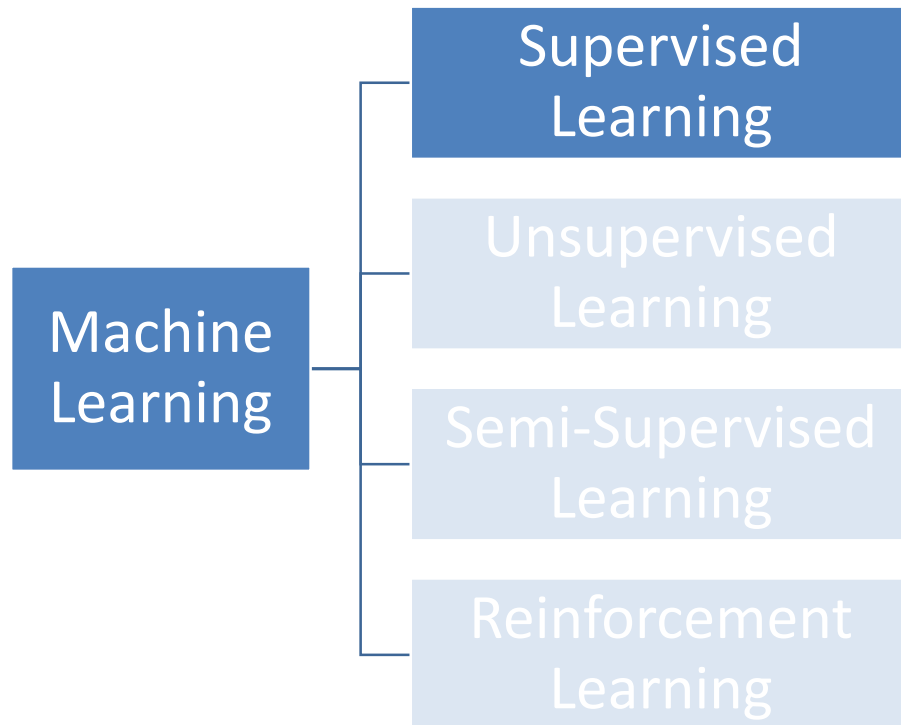
Data Availability



Κατηγοριοποίηση μεθόδων ML

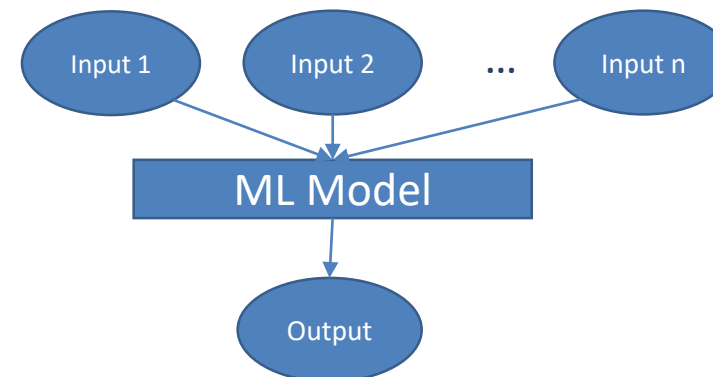


Categories of ML tasks

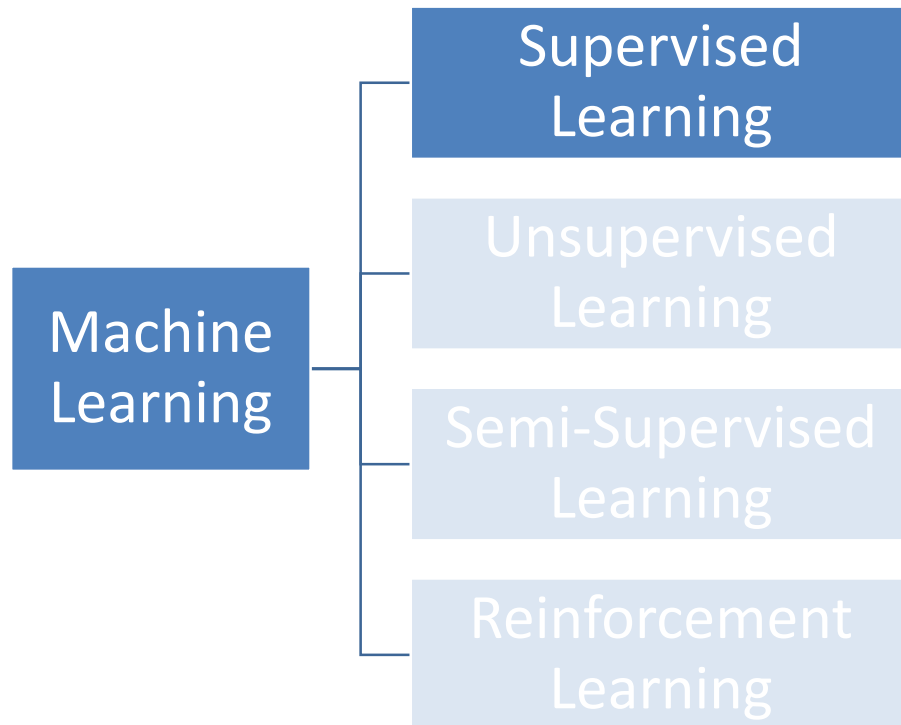


Επιτηρούμενη Μάθηση (Supervised Learning)

- ✓ Στόχος είναι η συσχέτιση των δεδομένων της εισόδου με μια κάποιο επιθυμητό αποτέλεσμα, το οποίο αποτελεί την έξοδο του μοντέλου ML.
- ✓ Τα ιστορικά δεδομένα εκπαίδευσης περιέχουν τις τιμές της εισόδου (**inputs**) καθώς και τις αντίστοιχες αναμενόμενες τιμές της εξόδου (**labels**)



Categories of ML tasks



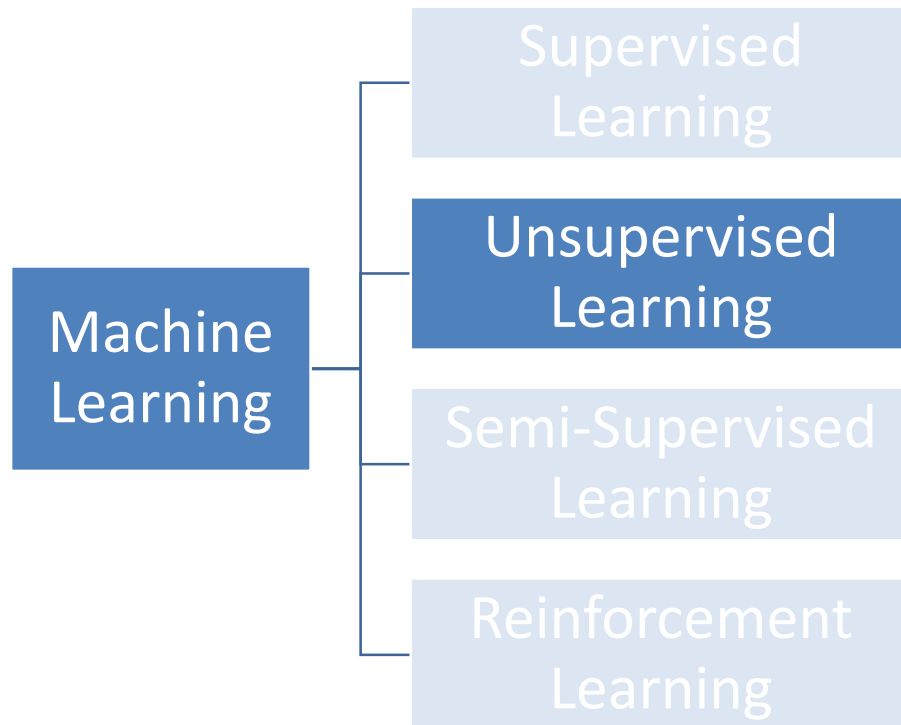
Τα μοντέλα που προκύπτουν από επιτηρούμενη εκπαίδευση είναι κυρίως κατάλληλα για επίλυση προβλημάτων:

- ✓ Ταξινόμησης (**Classification**): Ταξινόμηση δειγμάτων εισόδου σε 2 ή περισσότερες κλάσεις.
- ✓ Παλινδρόμησης (**Regression**): Εκτίμηση της τιμής μίας ή περισσότερων εξαρτημένων μεταβλητών.

Δημοφιλείς τύποι μοντέλων επιτηρούμενης μάθησης:

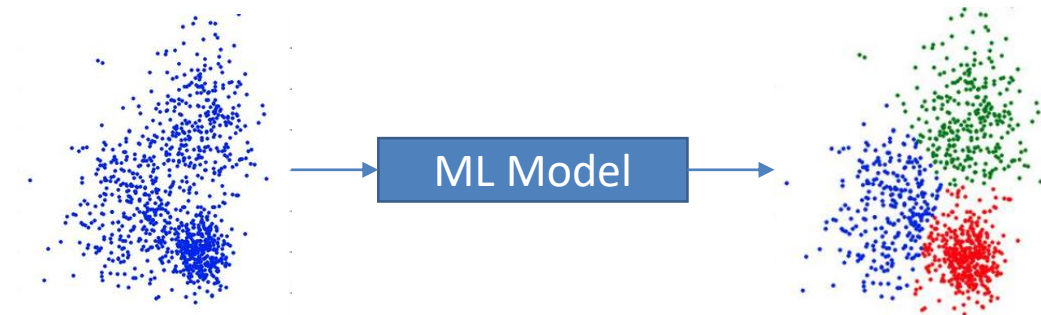
- Support Vector Machines
- Tree-based models
- Neural Networks

Categories of ML tasks

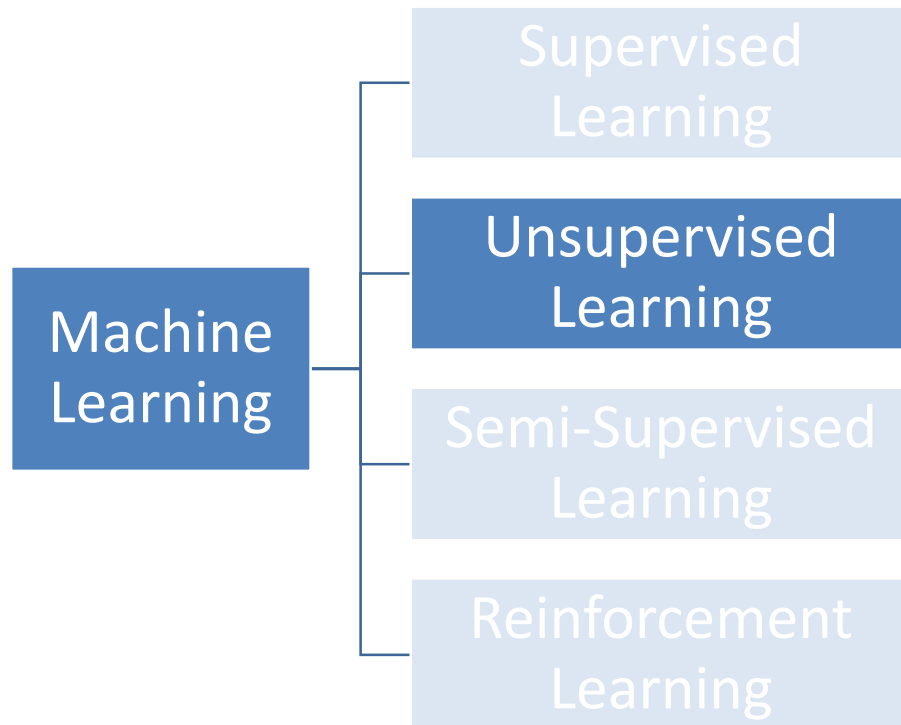


Μη Επιτηρούμενη Μάθηση (Unsupervised Learning)

- ✓ Στόχος είναι η **αναζήτηση δομής στα δεδομένα** η οποία θα αντιπροσωπεύει και θα εξηγεί βέλτιστα στις διαφοροποιήσεις που παρατηρούνται σε αυτά.
- ✓ Τα ιστορικά δεδομένα εκπαίδευσης δεν περιέχουν αναμενόμενες τιμές εξόδου (labels) για κάθε είσοδο, καθώς αυτές δεν είναι γνωστές εκ των προτέρων.



Categories of ML tasks



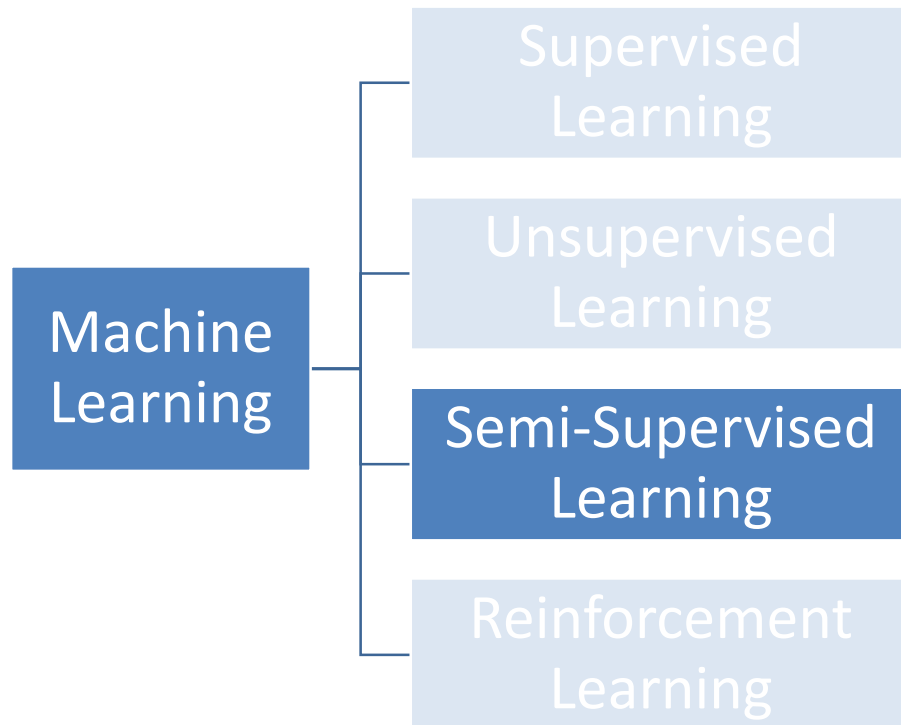
Τα μοντέλα που προκύπτουν από μη επιτηρούμενη εκπαίδευση είναι κυρίως κατάλληλα για επίλυση προβλημάτων:

- ✓ Συσταδοποίησης (**Clustering**)
- ✓ Ανίχνευσης ανωμαλιών (**Anomaly detection**)
- ✓ Μείωσης διαστάσεων (**Dimensionality reduction**)

Δημοφιλείς τύποι μοντέλων μη επιτηρούμενης μάθησης:

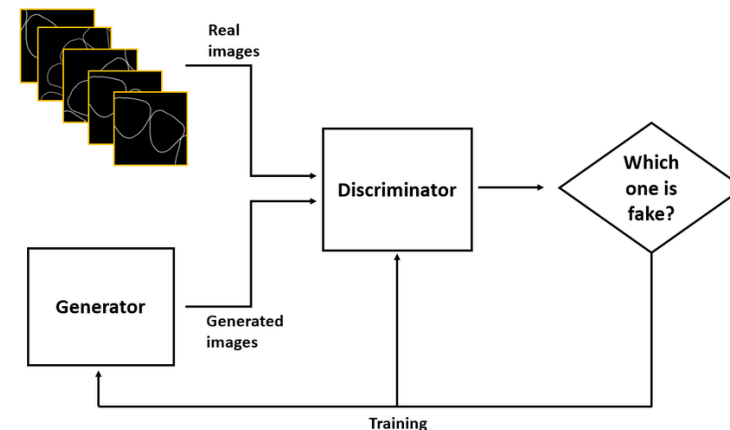
- Αλγόριθμοι Clustering
- Neural Networks (auto-encoders)

Categories of ML tasks

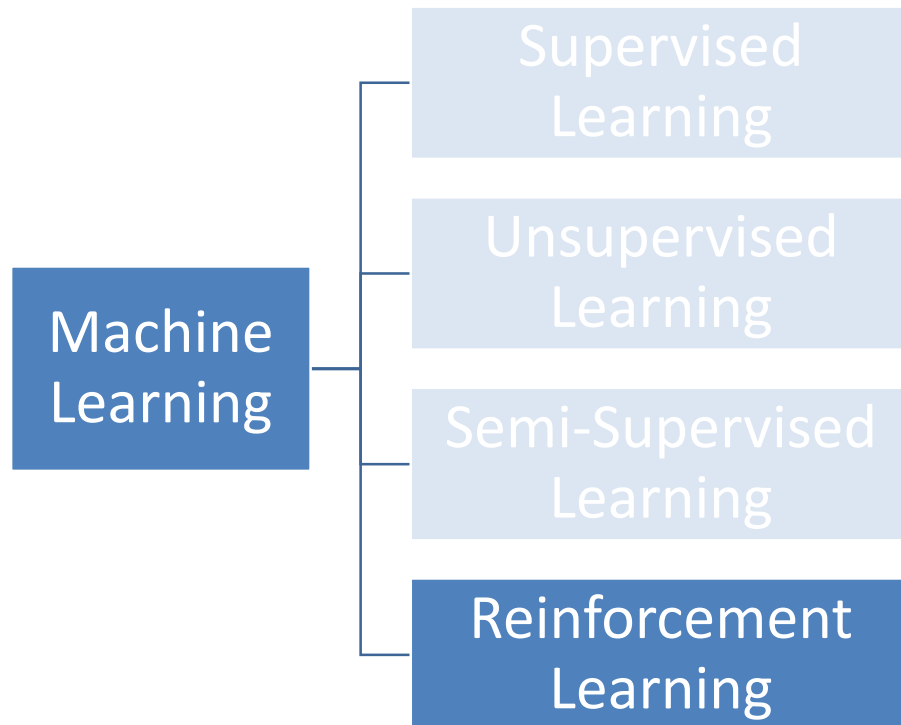


Ημι-Επιτηρούμενη Μάθηση (Semi-supervised Learning)

- ✓ Αποτελεί μια επέκταση των προηγούμενων τεχνικών μάθησης.
- ✓ Γενικά χρησιμοποιείται σε περιπτώσεις που ένας περιορισμένος αριθμός από τις πραγματικές εξόδους (labels) είναι διαθέσιμος.
- ✓ Για την υλοποίηση ημι-επιτηρούμενης μάθησης χρησιμοποιούνται ML μοντέλα από τις προηγούμενες δύο κατηγορίες μάθησης.



Categories of ML tasks

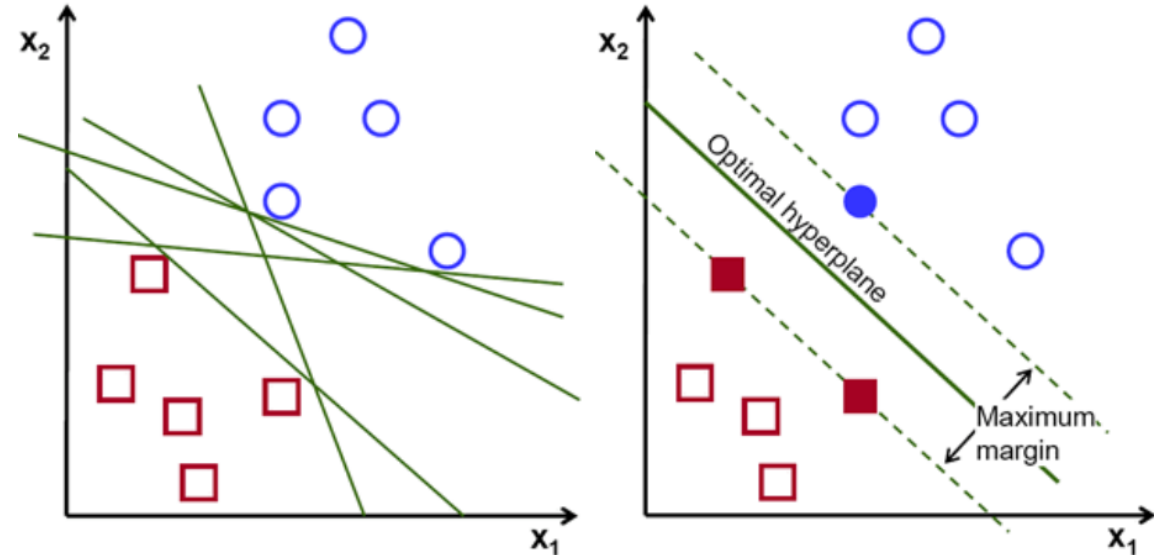


Ενισχυτική μάθηση (Reinforcement Learning)

- ✓ Αποτελεί μια τεχνική μηχανικής μάθησης με την οποία το σύστημα προσπαθεί να μάθει αλληλοεπιδρώντας με το περιβάλλον.
- ✓ Το σύστημα/πράκτορας (**agent**) αποφασίζει για κάποιες ενέργειες (**actions**) και, ανάλογα με τις συνθήκες και την ενέργεια που επιλέχθηκε, λαμβάνει κάποια επιβράβευση (**reward**).
- ✓ Τεχνικές ενισχυτικής μάθησης βρίσκουν εφαρμογή σε εφαρμογές στις οποίες ο στόχος είναι μια σειρά από ενέργειες που οδηγούν σε μια κατάσταση ελαχίστου κόστους (ή μέγιστης ανταμοιβής)
 - Video Games
 - Robotics
 - Self-driving cars

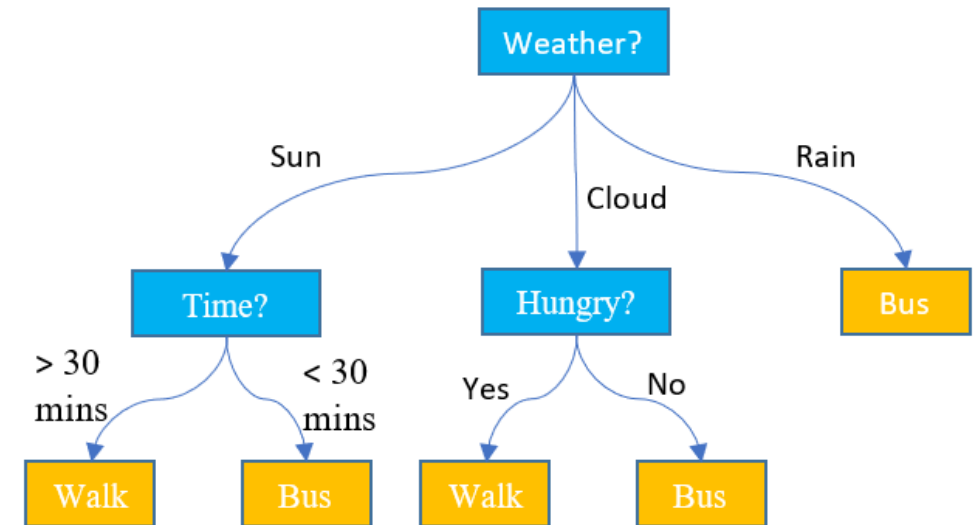
Support Vector Machines

- ✓ Οι μηχανές διανυσμάτων υποστήριξης (**Support Vector Machines** - SVM) κατασκευάζουν ένα υπερ-επίπεδο (hyperplane) στον χώρο των δεδομένων, το οποίο θα **διαχωρίζει βέλτιστα τις διαφορετικές κλάσεις** δεδομένων.
- ✓ SVMs μπορούν να χρησιμοποιηθούν για την επίλυση προβλημάτων ταξινόμησης (**classification**) αλλά και παλινδρόμησης (**regression**).



Decision Trees

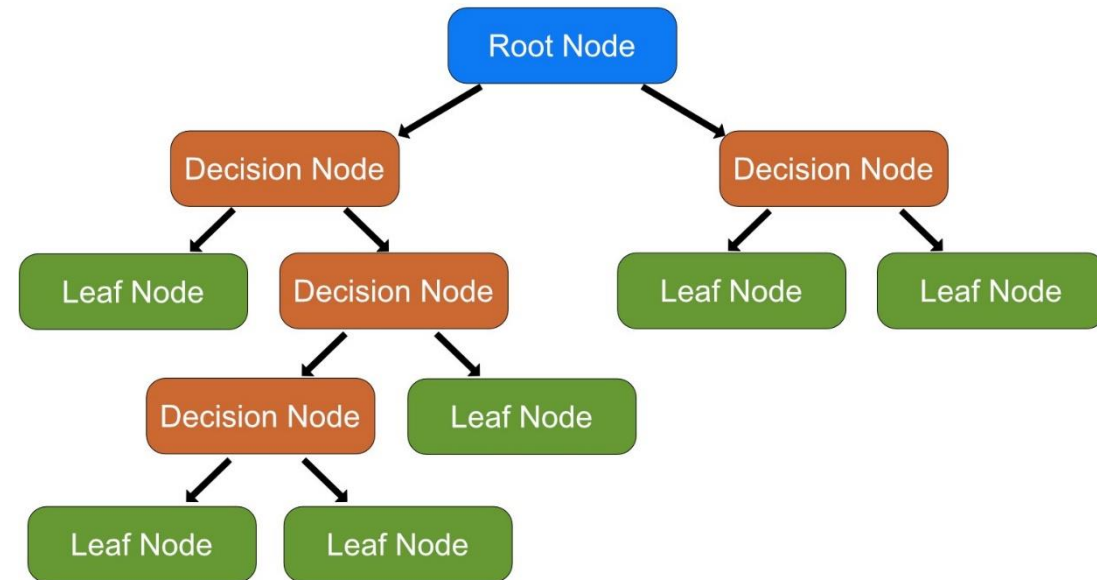
- Τα δέντρα αποφάσεων (**Decision Trees**) αποτελούν μοντέλα ML τα οποία παράγουν προβλέψεις κατασκευάζοντας κάποιους απλούς κανόνες απόφασης με βάση τα διαθέσιμα δεδομένα.
- Μπορούν να χρησιμοποιηθούν για την επίλυση προβλημάτων ταξινόμησης (**classification**) αλλά και παλινδρόμησης (**regression**).



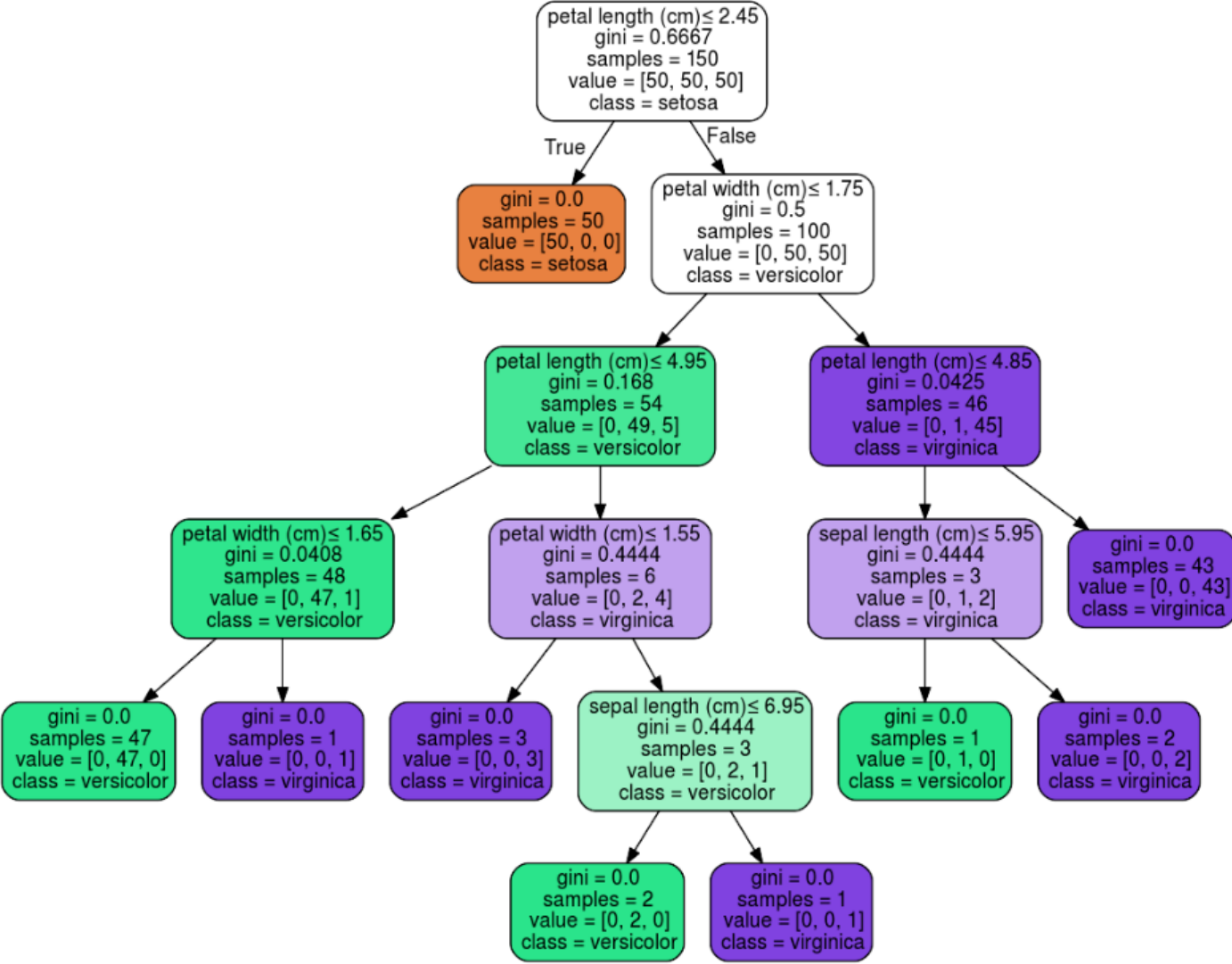
Decision Trees

Τα δέντρα απόφασης είναι δομές οι οποίες αποτελούνται από:

- ✓ Έναν κόμβο ρίζας (**root node**)
 - ✓ Εσωτερικούς κόμβους απόφασης (**decision nodes**)
 - ✓ Φύλλα (**leaf nodes**)
-
- ✓ Στην **ρίζα του δέντρου** είναι περιλαμβάνεται το σύνολο του πληθυσμού των εγγραφών (samples).
 - ✓ Σε κάθε **κόμβο απόφασης** ο πληθυσμός διχοτομείται σύμφωνα με το κριτήριο απόφασης που αντιστοιχεί σε αυτόν.
 - ✓ Το δέντρο καταλήγει σε **φύλλα**, τα οποία αποτελούν και την τελική πρόβλεψη για κάθε εγγραφή.

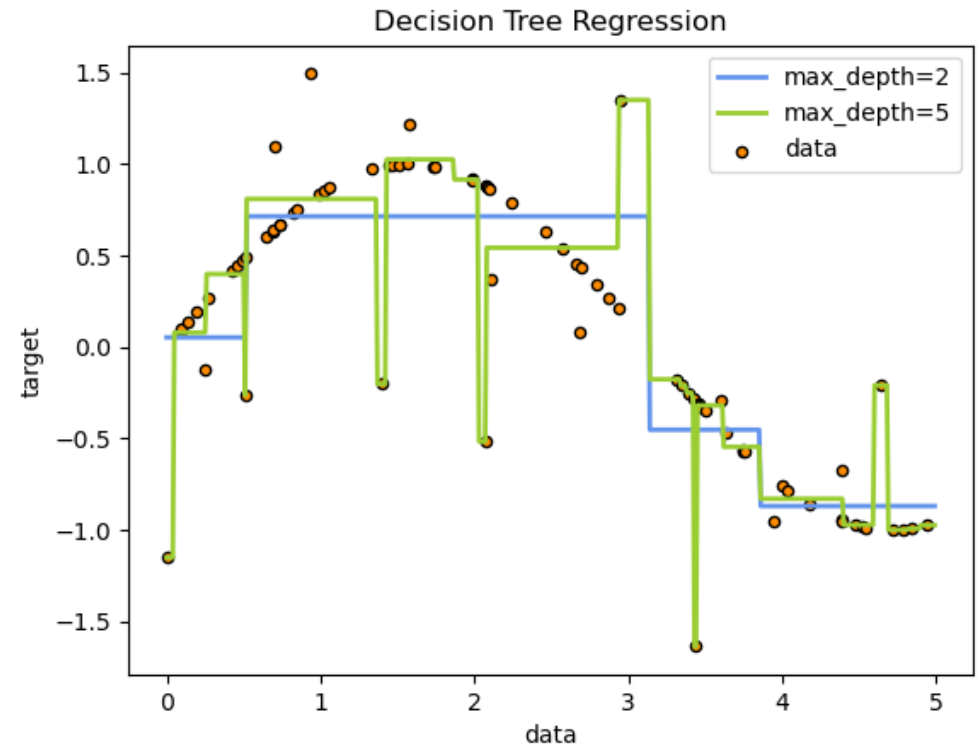


Decision Trees



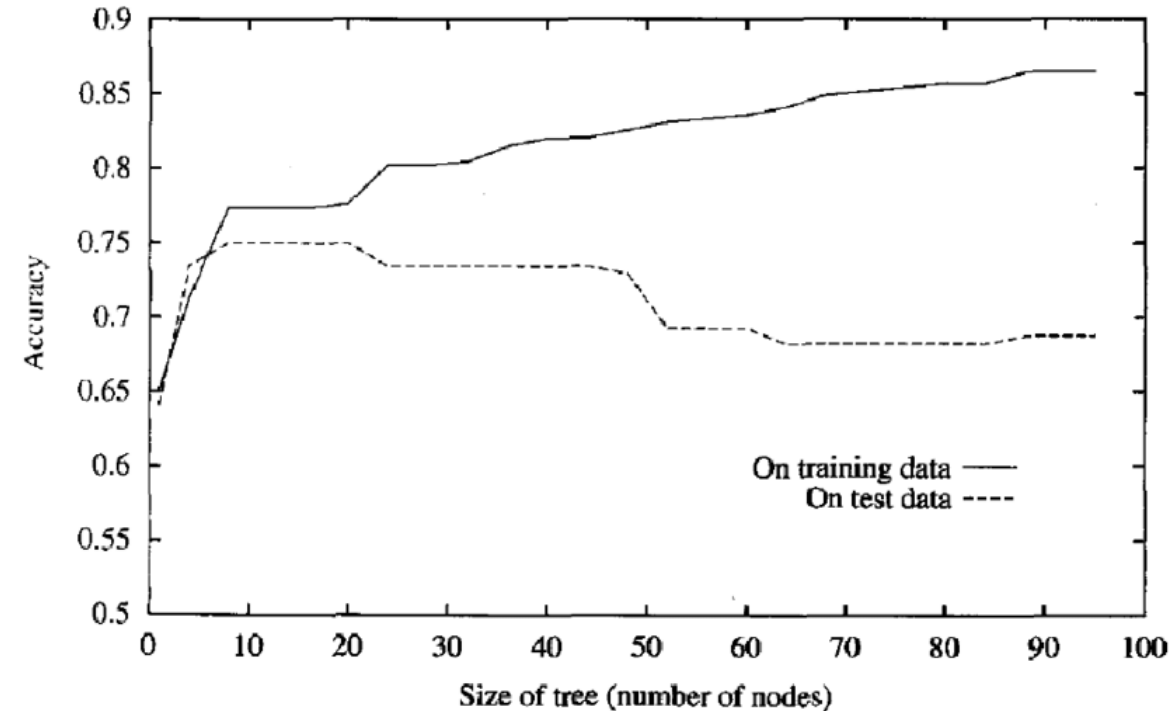
Decision Trees

- ✓ Τα δέντρα απόφασης είναι σχετικά απλά ML μοντέλα.
- ✓ Αποφάσεις ως προς τι παραμέτρους τους καθορίζουν την λειτουργικότητα και την πολυπλοκότητα τους.
 - 1. Κριτήριο βελτιστοποίησης** το οποίο κατευθύνει την εκπαίδευση και την κατασκευή του δέντρου. Καθορίζεται από την φύση του προβλήματος
 - Ταξινόμηση: gini, entropy
 - Παλινδρόμηση: MSE, MAE
 - 2. Μέγιστο βάθος δέντρου** (*max_depth*) δηλαδή το μέγιστο σε μήκος μονοπάτι από τη ρίζα στα φύλλα
 - 3. Ελάχιστος πληθυσμός δειγμάτων στον κόμβο** (*min_samples_split*) προκειμένου να επιτραπεί νέα διχοτόμηση
 - 4. Ελάχιστος πληθυσμός δειγμάτων στα φύλλα** (*min_samples_leaf*) προκειμένου να δημιουργηθούν



Decision Trees

- ✓ Τα δέντρα αποφάσεων αναπτύσσονται σε μεγαλύτερο βαθμό του επιθυμητού με αποτέλεσμα να υπερπροσαρμόζονται (**overfit**) στα δεδομένα εκπαίδευσης.
- ✓ Μπορούμε να κλαδέψουμε (**prune**) τα παραγόμενα δέντρα βάσει της πληροφορίας που αυτά ερμηνεύουν και της πολυπλοκότητάς τους.
- ✓ Υπάρχουν διαφορετικές τεχνικές κλαδέματος ανάλογα με το χρονικό σημείο που αυτό γίνεται:
 - Κατά την κατασκευή του δέντρου (**pre-pruning**)
 - Μετά την κατασκευή του πλήρους δέντρου (**post-pruning**)



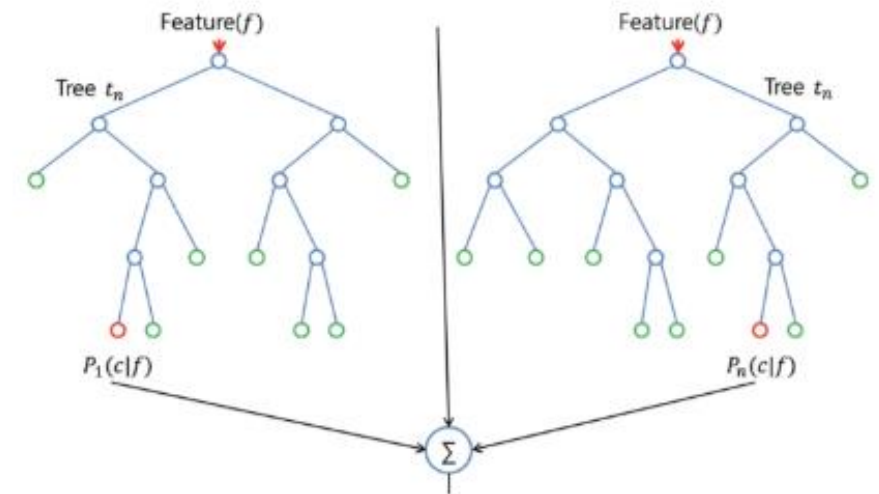
Decision Trees

- + Τα φύλλα προσδιορίζονται μέσω σαφώς ορισμένων κανόνων διχοτόμησης, και τα αποτελέσματα των δέντρων αποφάσεων μπορούν να **γίνουν κατανοητά και να οπτικοποιηθούν**.
- + **Δεν απαιτείται σημαντική προ-επεξεργασία** των αρχικών δεδομένων (scaling & adjustments).
- + Σε σχέση με άλλες μεθόδους μηχανικής μάθησης, εμφανίζουν **μικρότερη πολυπλοκότητα** και μπορούν να εκπαιδευτούν ταχύτερα.
- + Μπορούν να **μοντελοποιήσουν άμεσα τόσο αριθμητικές όσο και κατηγορικές μεταβλητές** χωρίς κάποιον μετασχηματισμό.
- Υπάρχει ο κίνδυνος δημιουργίας ενός υπερβολικά σύνθετου δέντρου το οποίο **υπερ-προσαρμόζεται στα δεδομένα** εκπαίδευσης (over-fitting).
- Μπορούν να γίνουν **σχετικά ασταθή** όταν τα δεδομένα εκπαίδευσης εμφανίζουν διακυμάνσεις.
- Η χρήση ευριστικών αλγορίθμων για την κατασκευή του βέλτιστου δέντρου σημαίνει πως **το τελικό αποτέλεσμα μπορεί να μην είναι το συνολικά βέλτιστο**.
- Όταν κάποιες κλάσεις στα δεδομένα εκπαίδευσης υπερτερούν πληθυσμιακά, τότε τα **δέντρα είναι ευάλωτα σε προκαταλήψεις**.

Adaptive Boosting

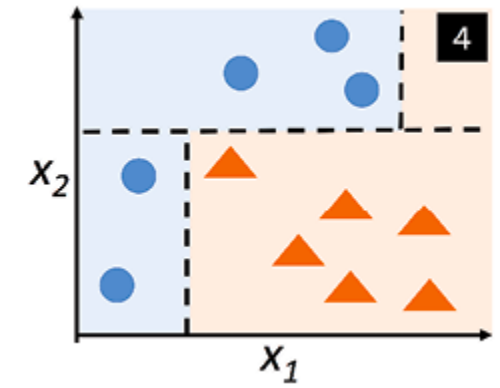
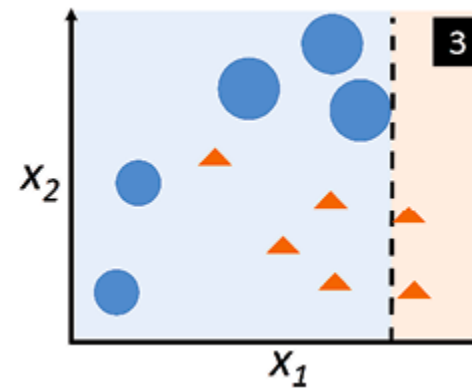
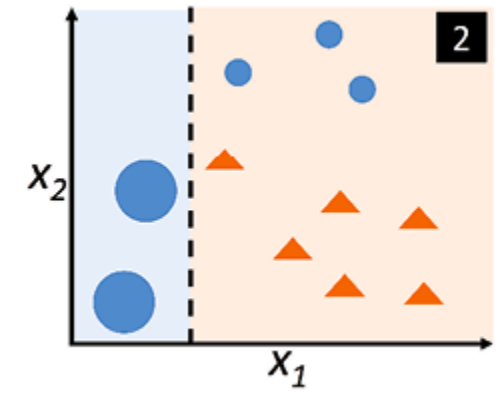
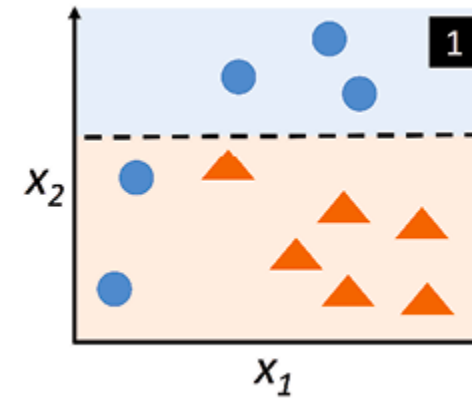
- Τα δέντρα αποφάσεων έχουν περιορισμένες δυνατότητες μοντελοποίησης. Η χρήση ενός απλού δέντρου δεν προτείνεται στην πράξη. Ο **συνδυασμός δέντρων αποφάσεων** (weak learners) οδηγεί σε σημαντικά μεγαλύτερη ακρίβεια πρόβλεψης.
- Ο αλγόριθμος Adaptive Boosting (**AdaBoost**) παρέχει μια εναλλακτική προσέγγιση συνδυάζοντας γραμμικά απλά δένδρα αποφάσεων.
- Με την εφαρμογή της τεχνικής boosting, το τελικό αποτέλεσμα του αλγορίθμου δίνεται από τον παρακάτω τύπο, όπου όπου f_t είναι οι έξοδοι των επιμέρους weak learners, x οι μεταβλητές εισόδου και T το πλήθος τους:

$$F_T(x) = \sum_{t=1}^T f_t(x)$$



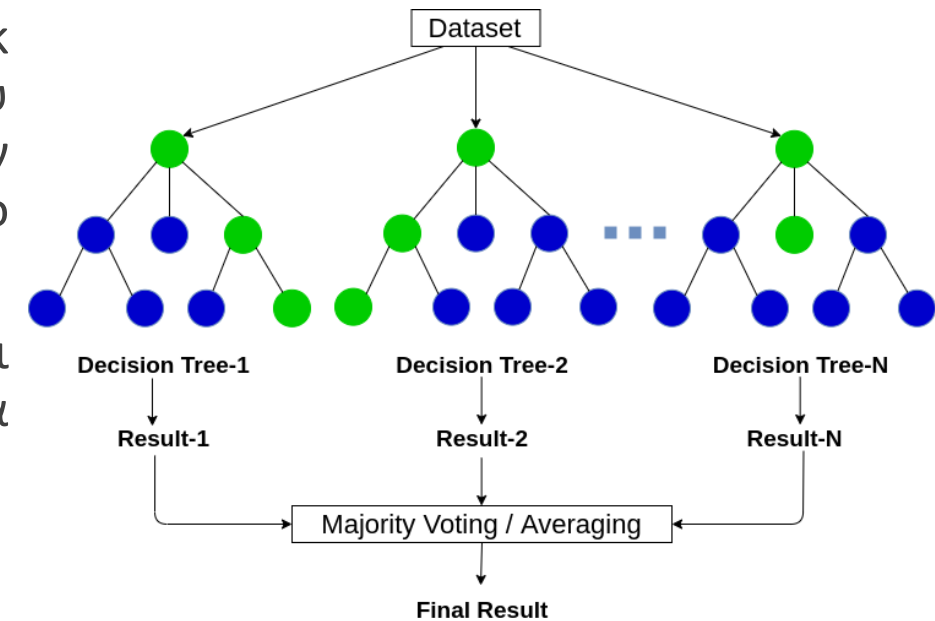
Adaptive Boosting

1. Εκπαίδευση ενός απλού δέντρου - weak learner 1.
2. Αξιολόγηση ακρίβειας ταξινόμησης weak learner 1. (εικ 1)
3. Εκπαίδευση δεύτερου απλού δέντρου - weak learner, δίνοντας μεγαλύτερο βάρος στα λάθος ταξινομημένα δείγματα από το προηγούμενο δέντρο.
4. Αξιολόγηση ακρίβειας ταξινόμησης weak learner 2. (εικ 2)
5. Εκπαίδευση τρίτου απλού δέντρου - weak learner, δίνοντας μεγαλύτερο βάρος στα λάθος ταξινομημένα δείγματα των προηγούμενων δέντρων.
6. Αξιολόγηση ακρίβειας ταξινόμησης weak learner 3. (εικ 3)
7. Συνδυασμός των 3 δέντρων – weak learners σε ένα τελικό, πιο ισχυρό μοντέλο. (εικ 4)



Random Forests

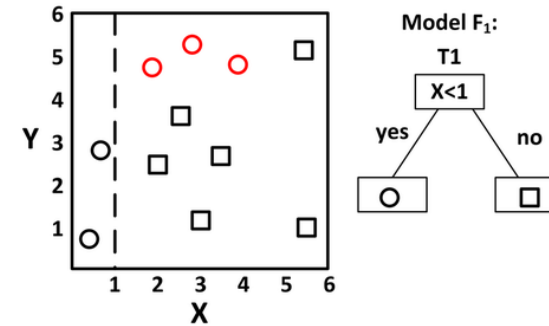
- Τα **Random Forests** αποτελούν μια διαφορετική επιλογή ως προς τον συνδυασμό απλών δέντρων αποφάσεων.
- Σε αντίθεση με τον αλγόριθμο Ada Boost, όπου οι weak learners συνδυάζονται με βαρύτητες κατά την ανάπτυξη του τελικού δέντρου, τα Random Forests αναφέρονται στην ανάπτυξη ανεξάρτητων weak learners και στον απλό **συνδυασμό των ψήφων τους**.
- Κάθε δέντρο το οποίο συμμετέχει στο Random Forest έχει εκπαιδευτεί με μοναδικό τρόπο και συνεπώς προσφέρει μια διαφορετική «οπτική».
 - Με διαφορετικό σύνολο δεδομένων (samples)
 - Με διαφορετικό σύνολο μεταβλητών εισόδου (features)



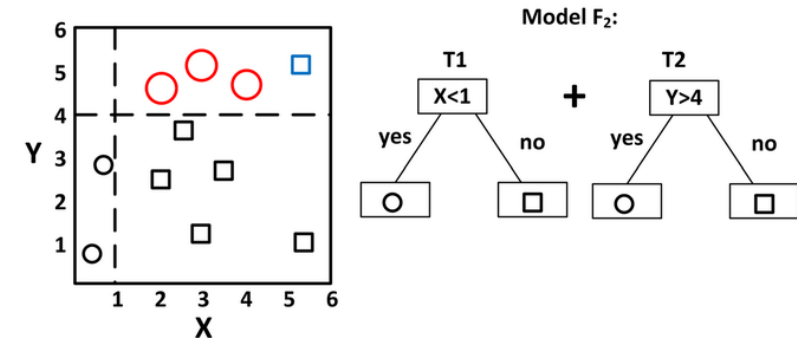
Gradient Boosting

- Ο αλγόριθμος του **Gradient Boosting** αποτελεί ακόμα μια επιλογή για τον συνδυασμό απλών δέντρων αποφάσεων.
- Σε αντίθεση με τα Random Forests, στην περίπτωση του Gradient Boosting τα δέντρα αποφάσεων δεν αναπτύσσονται ανεξάρτητα, αλλά σειριακά και **ενισχύεται σταδιακά η γνώση των weak learners** που κατά το προηγούμενο βήμα εμφάνισαν το μεγαλύτερο σφάλμα.
- Η ενίσχυση γίνεται μέσω της **προσθήκης ενός επιπλέον weak learner** στο αντίστοιχο φύλλο του προηγούμενου weak learner.
- Το **τελικό δέντρο που δημιουργείται διορθώνει τα λάθη του** και καταλήγει σε πιο ακριβή αποτελέσματα.
- Ο αλγόριθμος Gradient Boosting είναι **ευάλωτος στην υπερπροσαρμογή** (overfitting) αλλά μπορεί ταυτόχρονα να δώσει πολύ λεπτομερείς λύσεις σε πολύπλοκα προβλήματα.

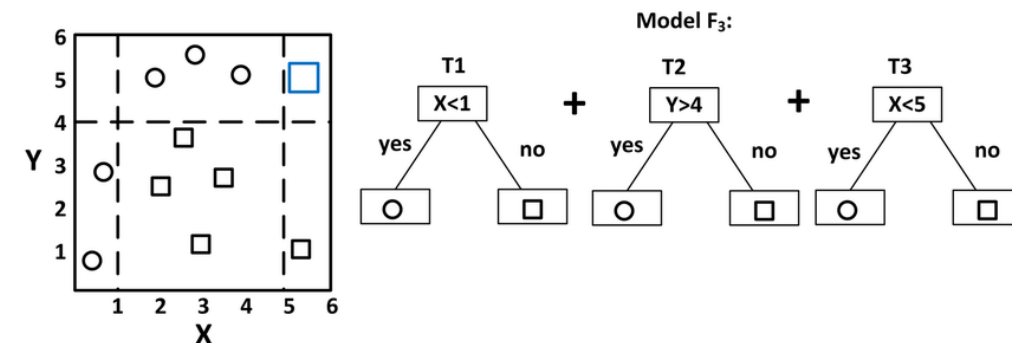
Iteration 1



Iteration 2

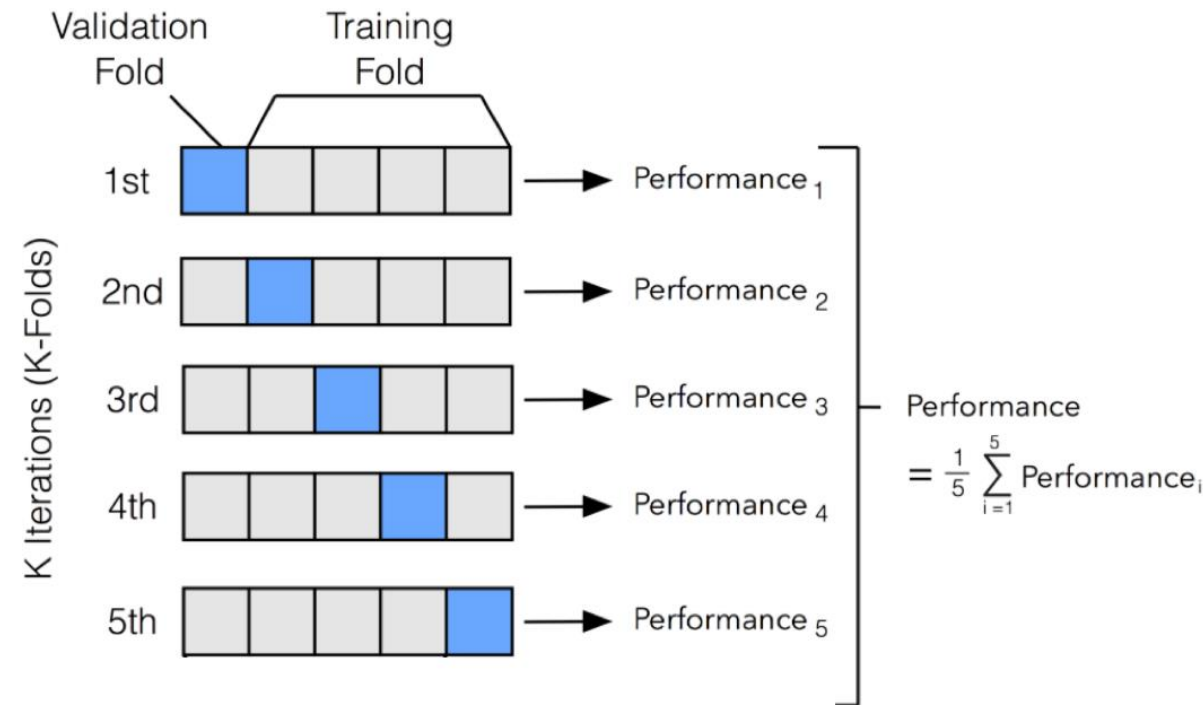


Iteration 3



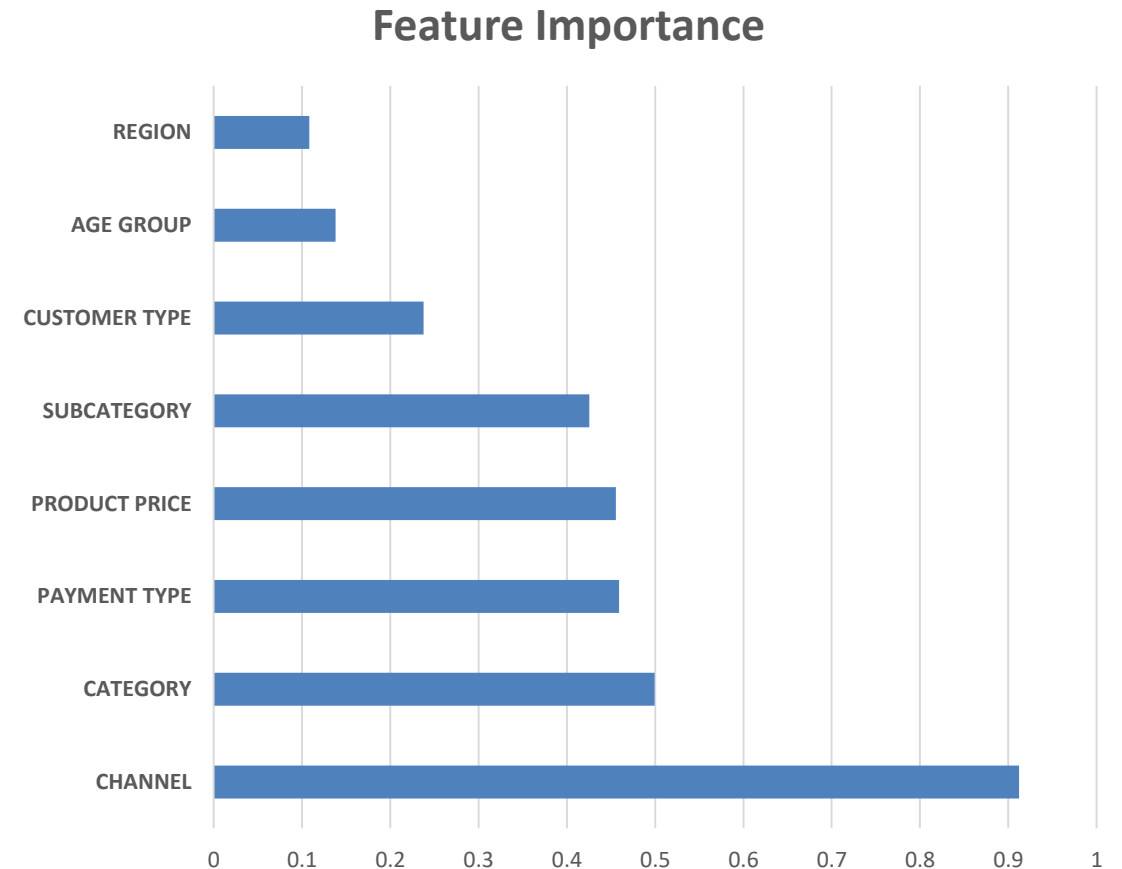
Model Validation

- Η αξιολόγηση διαφορετικών ML μοντέλων και η επιλογή του καλύτερου από αυτά είναι μια ιδιαίτερα σημαντική διαδικασία.
- Η παρακολούθηση αποκλειστικά του σφάλματος εκπαίδευσης (**train loss**) των μοντέλων μπορεί να είναι παραπλανητική.
- Είναι απαραίτητη η χρήση κάποια **τεχνικής επαλήθευσης** (validation) του μοντέλου.
- Συχνά, η στρατηγική του **cross validation** χρησιμοποιείται για την αντικειμενική αξιολόγηση μοντέλων.



Feature Importance

- Κατά την ανάπτυξη μοντέλων παραγωγής προβλέψεων, είναι σημαντική η **αξιολόγηση των ανεξάρτητων μεταβλητών** (features) που είναι διαθέσιμες.
- Προκειμένου να έχουμε αντίληψη σχετικά με τη σημαντικότητα των ανεξάρτητων μεταβλητών στη διαδικασία παραγωγής των τελικών προβλέψεων, εξετάζουμε τα μεγέθη που αναφέρονται ως **feature importance**.
- Υπάρχουν **διαφορετικοί τρόποι υπολογισμού** της σημαντικότητας των μεταβλητών, οι οποίοι διαφέρουν ανάλογα με τον αλγόριθμο που χρησιμοποιείται.



Data Pre-processing

Ανάλογα με το μοντέλο που χρησιμοποιείται αλλά και την φύση των δεδομένων, είναι πιθανό να υπάρχει ανάγκη για προ-επεξεργασία των δεδομένων (**data pre-processing**).

Καθαρισμός των δεδομένων (data cleaning)

- Εξάλειψη ακραίων τιμών (outliers)
- Συμπλήρωση τιμών που απουσιάζουν (missing values)

Μετασχηματισμός μεταβλητών (feature transformation)

- Αλλαγή τύπου (data type) ή μορφής (format) των δεδομένων.
- Κωδικοποίηση της πληροφορίας κάποιων μεταβλητών (π.χ. One-hot encoding)

Εξαγωγή νέων μεταβλητών (feature engineering)

- Διαχωρισμός υπαρχόντων μεταβλητών σε επιμέρους στοιχεία (πχ ημερολογιακά).
- Δημιουργία νέων μεταβλητών.

Tips!

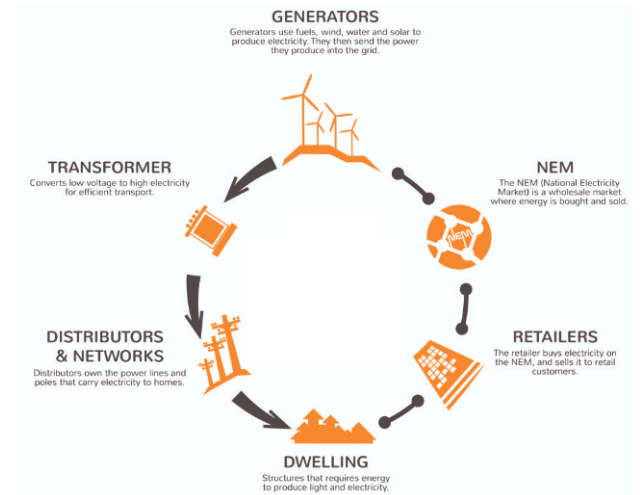
- Η ανάπτυξη ενός προβλεπτικού μοντέλου θα πρέπει πάντα να προηγείται από μια βασική εξερεύνηση των δεδομένων (**exploratory data analysis - EDA**).
- Είναι σημαντικό να αποφασίζεται νωρίς το μέτρο αξιολόγησης (**evaluation metric**) με το οποίο αξιολογούνται τα αποτελέσματα των μοντέλων.
- Η προ-επεξεργασία (**pre-processing**) των δεδομένων παίζει σημαντικό ρόλο στην τελική ακρίβεια των μοντέλων.
- Η υλοποίηση ενός σταθερού πλαισίου αξιολόγησης των μοντέλων (**model validation framework**) επιταχύνει τις δοκιμές και διασφαλίζει ότι τα αποτελέσματα τους είναι συγκρίσιμα.
- Η οπτικοποίηση (**visualization**) τόσο των δεδομένων όσο και των αποτελεσμάτων βοηθάει σημαντικά στην κατανόηση και ερμηνεία τους.
- Πάντα έχει αξία η δοκιμή διαφορετικών μοντέλων σε κάθε νέο πρόβλημα. Το βέλτιστο μοντέλο για ένα πρόβλημα μπορεί να **μην είναι βέλτιστο για κάποιο άλλο** πρόβλημα.

Python Example

Παράδειγμα πρόβλεψης με χρήση μοντέλων ML

Το πρόβλημα που καλούμαστε να λύσουμε ανήκει στον χώρο των προβλέψεων ενέργειας και, πιο συγκεκριμένα, αφορά την πρόβλεψη της μελλοντικής τιμής της ενέργειας στο Βέλγιο.

- ✓ **Στόχος (Target):** Η τιμή της ηλεκτρικής ενέργειας στο Βέλγιο
- ✓ **Συχνότητα Δεδομένων (Data Frequency):** Ωριαία (hourly) δεδομένα
- ✓ **Ορίζοντας πρόβλεψης (Forecasting horizon):** 1 εβδομάδα = 168 ώρες
- ✓ **Διαθέσιμες μεταβλητές:**
 - Οι χρονική στιγμή των παρατηρήσεων.
 - Η παραγωγή ενέργειας στο Βέλγιο.
 - Η παραγωγή ενέργειας στην Γαλλία.
 - Οι εθνικές αργίες στο Βέλγιο.



Python Example

```
import pandas as pd  
import numpy as np
```

Από τις πιο **βασικές βιβλιοθήκες** της Python.
Προσθέτουν νέες δομές δεδομένων (πίνακες, σειρές), έτοιμες συναρτήσεις, κλπ.

```
import matplotlib.pyplot as plt  
import seaborn as sns
```

Δημοφιλείς βιβλιοθήκες για **οπτικοποίηση** δεδομένων

```
from sklearn.ensemble import RandomForestRegressor  
import xgboost as xgb
```

- ✓ XGBoost: Βασίζεται στον αλγόριθμο του Gradient Boosting.
- ✓ Κατάλληλο για regression, classification και sorting προβλήματα.
- ✓ Αποτελεί ένα από τα ισχυρότερα ML μοντέλα.

- ✓ Scikit-Learn: η πιο **βασική ML βιβλιοθήκη** στην Python
- ✓ Παρέχει μεγάλο αριθμό **διαφορετικών μοντέλων**
- ✓ Καλύπτει **διαφορετικές κατηγορίες προβλημάτων** (regression, classification, clustering, sorting)
- ✓ Προσφέρει **βοηθητικές λειτουργίες** για προεπεξεργασία δεδομένων, αξιολόγηση μοντέλων, κλπ.

Python Example

```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
```

```
# Read Data
```

```
path = 'TrainSet.csv'
df_all = pd.read_csv(path, sep=',', decimal='.')
```

Φόρτωση των δεδομένων
σε **Pandas DataFrame**

```
print(df_all.describe())
```

```
print(df_all.head())
```

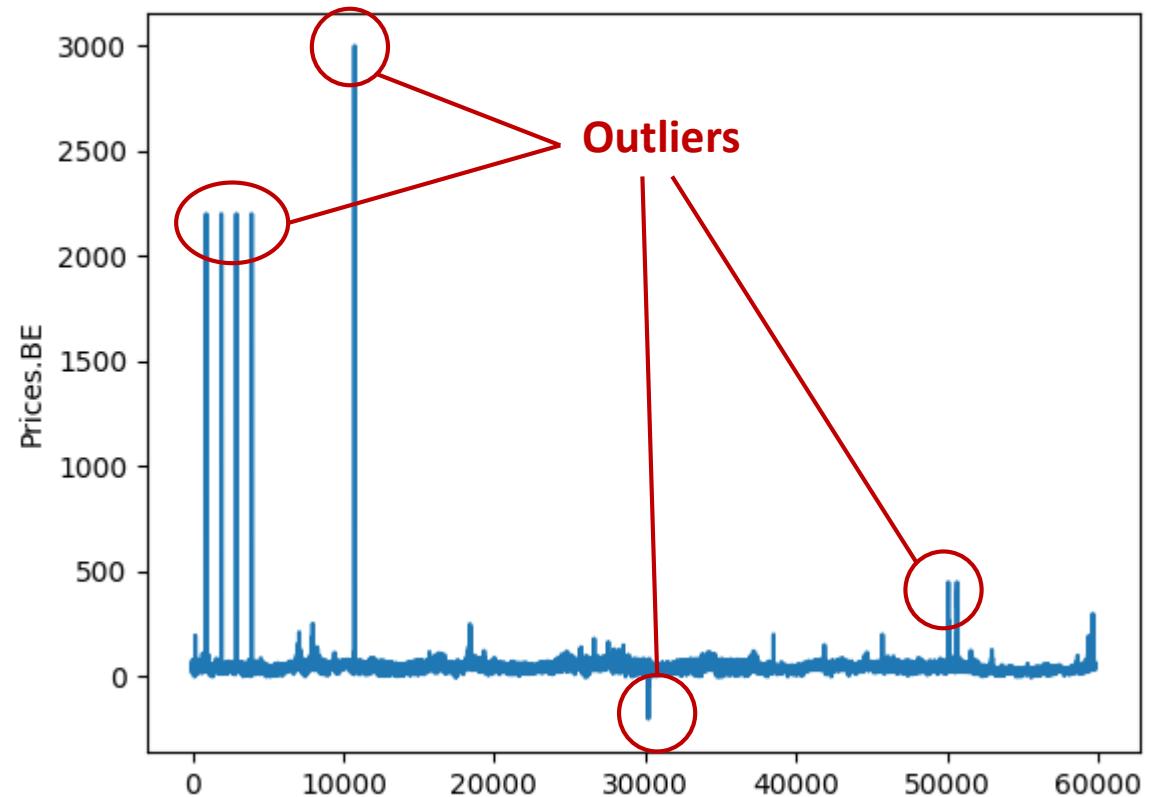
	datetime_utc	Generation_BE	...	Prices_BE	holidaysBE
0	2010-01-04 00:00:00	13143.7	...	37.15	0
1	2010-01-04 01:00:00	13143.7	...	33.47	0
2	2010-01-04 02:00:00	13143.7	...	28.00	0
3	2010-01-04 03:00:00	13143.7	...	21.29	0
4	2010-01-04 04:00:00	13143.7	...	16.92	0

	Generation_BE	Generation_FR	Prices_BE	holidaysBE
count	59808.000000	59808.000000	59779.000000	59808.000000
mean	14558.188959	59176.443185	44.406865	0.026485
std	4385.094942	10330.524662	27.882055	0.160573
min	7678.125000	33153.000000	-200.000000	0.000000
25%	12328.700000	51316.000000	32.985000	0.000000
50%	13398.482500	57676.000000	44.420000	0.000000
75%	14896.045000	65908.000000	54.620000	0.000000
max	35778.512500	93517.000000	2999.000000	1.000000

Python Example

```
# Visualize data  
sns.lineplot(data=df_all['Prices.BE'])  
plt.show()
```

Δημιουργία ενός **απλού γραφήματος**
για την απεικόνιση της επιλεγμένης
μεταβλητής

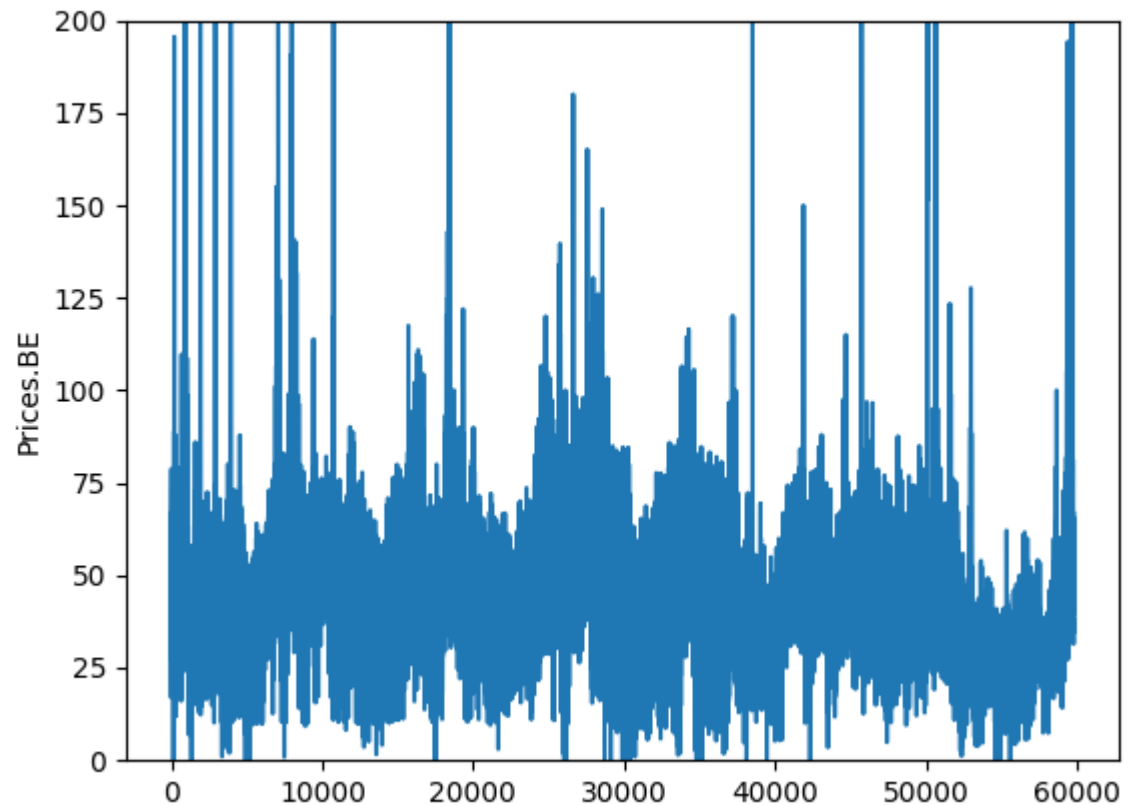


Python Example

```
# Visualize data  
sns.lineplot(data=df_all['Prices.BE'])  
plt.show()
```

```
# Visualize data - limit Y-axis  
sns.lineplot(data=df_all['Prices.BE'])  
plt.ylim(0, 200)  
plt.show()
```

Περιορισμός του διαστήματος
του άξονα Y (0 με 200)



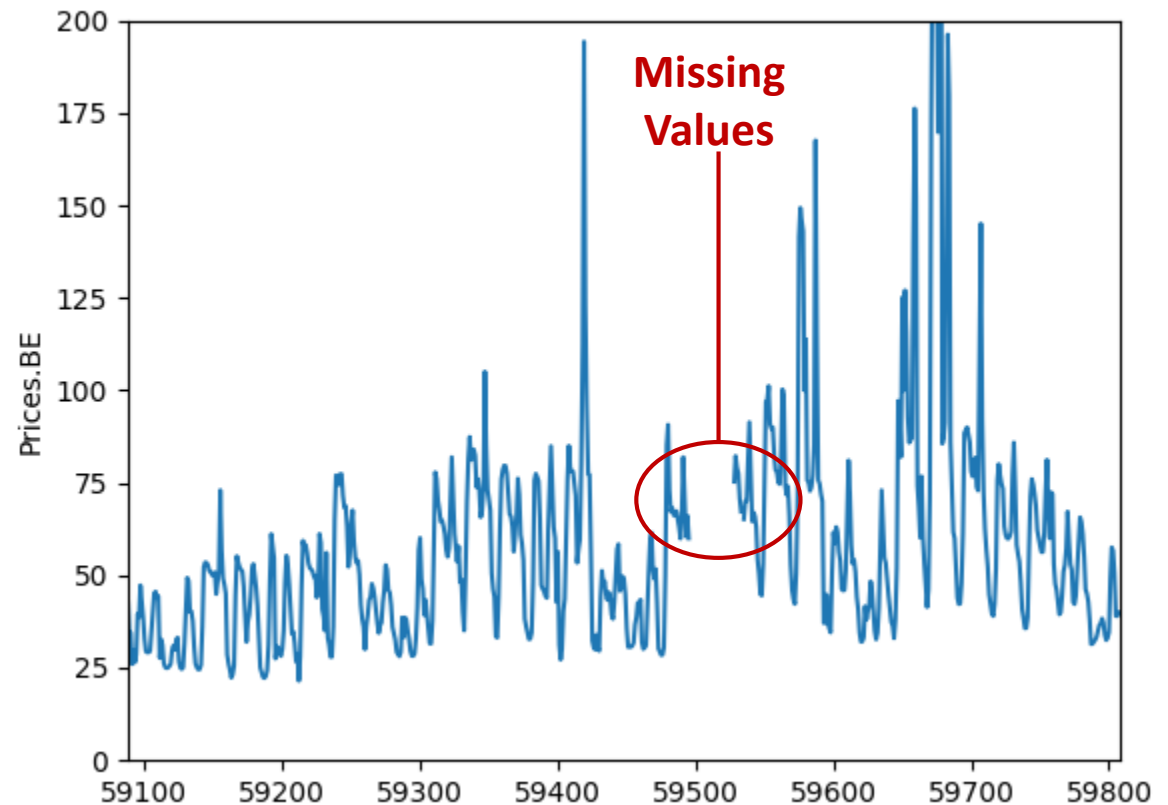
Python Example

```
# Visualize data
sns.lineplot(data=df_all['Prices.BE'])
plt.show()

# Visualize data - limit Y-axis
sns.lineplot(data=df_all['Prices.BE'])
plt.ylim(0, 200)
plt.show()

# Visualize data - limit Y-axis & plot last 30 days
sns.lineplot(data=df_all['Prices.BE'])
plt.ylim(0, 200)
plt.xlim(df_all.shape[0] - (30 * 24), df_all.shape[0])
plt.show()
```

Περιορισμός του διαστήματος του άξονα X, ώστε να απεικονίσουμε τις τελευταίες 30 ημέρες



Python Example

```
# Find all missing values
missing_values_index_list = df_all[df_all['Prices.BE'].isnull()].index.to_list()
print(len(missing_values_index_list))
print(df_all.loc[missing_values_index_list])

for i in missing_values_index_list:
    df_all.loc[i, 'Prices.BE'] = df_all['Prices.BE'].iloc[i - 168]
```

29 missing values

Οι τιμές που λείπουν συμπληρώνονται με την τιμή που παρατηρήθηκε την αντίστοιχη ώρα, την αντίστοιχη μέρα, την προηγούμενη βδομάδα

	datetime_utc	Generation_BE	Generation_FR	Prices_BE	holidaysBE
59496	2016-10-18 00:00:00	9200.3375	48167.0	NaN	0
59497	2016-10-18 01:00:00	9201.8200	46300.0	NaN	0
59498	2016-10-18 02:00:00	9211.8550	46482.0	NaN	0
59499	2016-10-18 03:00:00	9227.1825	45650.0	NaN	0
59500	2016-10-18 04:00:00	9257.5400	45059.0	NaN	0
59501	2016-10-18 05:00:00	9313.8375	45142.0	NaN	0
59502	2016-10-18 06:00:00	9397.6950	48633.0	NaN	0
59503	2016-10-18 07:00:00	9490.1775	51597.0	NaN	0
59504	2016-10-18 08:00:00	10004.7750	52651.0	NaN	0
59505	2016-10-18 09:00:00	12458.7000	52529.0	NaN	0
59506	2016-10-18 10:00:00	14430.6375	52537.0	NaN	0
59507	2016-10-18 11:00:00	15114.1650	52105.0	NaN	0
59508	2016-10-18 12:00:00	15043.3150	50970.0	NaN	0
59509	2016-10-18 13:00:00	14432.0700	50628.0	NaN	0
59510	2016-10-18 14:00:00	13408.2650	50137.0	NaN	0
59511	2016-10-18 15:00:00	12297.1925	49389.0	NaN	0
59512	2016-10-18 16:00:00	11376.8875	49309.0	NaN	0
59513	2016-10-18 17:00:00	10406.4300	49316.0	NaN	0
59514	2016-10-18 18:00:00	9745.6750	50368.0	NaN	0
59515	2016-10-18 19:00:00	9631.6800	55508.0	NaN	0
59516	2016-10-18 20:00:00	9628.3350	54147.0	NaN	0
59517	2016-10-18 21:00:00	9645.0500	51333.0	NaN	0
59518	2016-10-18 22:00:00	9857.8325	49829.0	NaN	0
59519	2016-10-18 23:00:00	10350.6275	49455.0	NaN	0
59521	2016-10-19 01:00:00	10440.4250	47161.0	NaN	0
59523	2016-10-19 03:00:00	10456.9775	46460.0	NaN	0
59525	2016-10-19 05:00:00	10480.7475	46129.0	NaN	0
59527	2016-10-19 07:00:00	10503.5100	52286.0	NaN	0
59534	2016-10-19 14:00:00	14432.6800	50915.0	NaN	0

Python Example

```
# Find all missing values
missing_values_index_list = df_all[df_all['Prices.BE'].isnull()].index.to_list()
print(len(missing_values_index_list))
print(df_all.loc[missing_values_index_list])

for i in missing_values_index_list:
    df_all.loc[i, 'Prices.BE'] = df_all['Prices.BE'].iloc[i - 168]

# Deal with the outliers
lower_limit = df_all['Prices.BE'].quantile(0.001)
upper_limit = df_all['Prices.BE'].quantile(0.999)
df_all['Prices.BE'] = df_all['Prices.BE'].clip(lower=lower_limit, upper=upper_limit)
```

Για την αναγνώριση των ακραίων τιμών ορίζουμε φίλτρο το οποίο έχει σαν όρια το **0.1%** και το **99.9%** των τιμών.

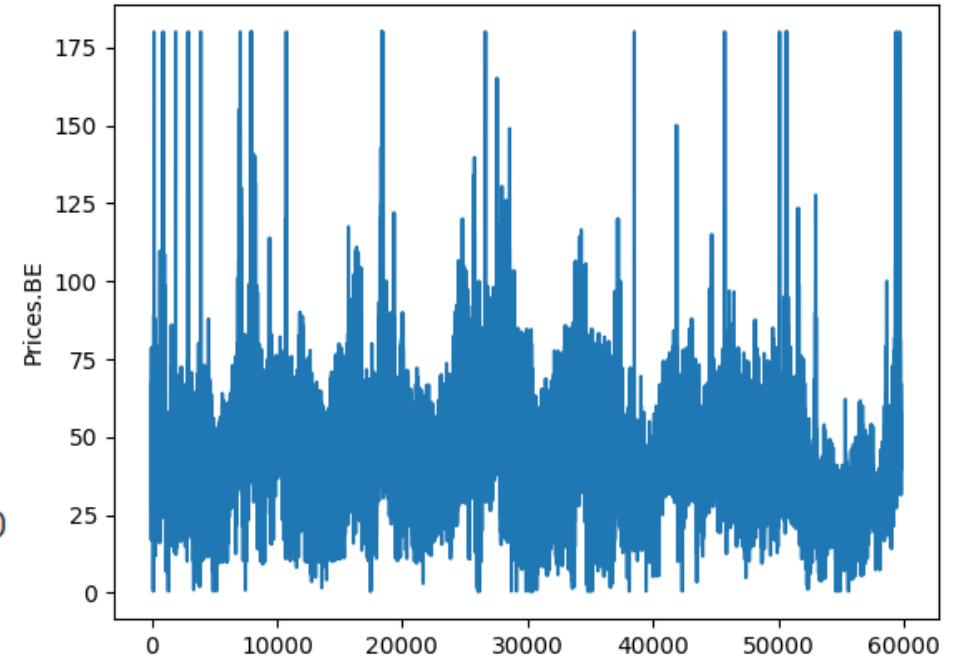
Όποια τιμή βρίσκεται έξω από τα όρια που ορίστηκαν, «ψαλιδίζεται».

Python Example

```
# Find all missing values
missing_values_index_list = df_all[df_all['Prices.BE'].isnull()].index.to_list()
print(len(missing_values_index_list))
print(df_all.loc[missing_values_index_list])

for i in missing_values_index_list:
    df_all.loc[i, 'Prices.BE'] = df_all['Prices.BE'].iloc[i - 168]

# Deal with the outliers
lower_limit = df_all['Prices.BE'].quantile(0.001)
upper_limit = df_all['Prices.BE'].quantile(0.999)
df_all['Prices.BE'] = df_all['Prices.BE'].clip(lower=lower_limit, upper=upper_limit)
sns.lineplot(data=df_all['Prices.BE'])
plt.show()
```



Python Example

```
# Extract additional features  
df_all['Date'] = pd.to_datetime(df_all['datetime_utc'])  
df_all['Year'] = df_all['Date'].dt.year  
df_all['Month'] = df_all['Date'].dt.month  
df_all['Weekday'] = df_all['Date'].dt.dayofweek  
df_all['Hour'] = df_all['Date'].dt.hour
```

Ορισμένες φορές είναι απαραίτητη η **μετατροπή τύπου** μιας μεταβλητής (π.χ. από «string» σε «datetime»)

- ✓ Ένας ML αλγόριθμος **δεν μπορεί να ερμηνεύσει την πληροφορία** που περιέχει μια ημερομηνία με τον τρόπο που θα το έκανε ένας άνθρωπος.
- ✓ Προτείνεται η **εξαγωγή νέων χαρακτηριστικών** με την μορφή ανεξάρτητων μεταβλητών

Python Example

```
# Extract additional features
df_all['Date'] = pd.to_datetime(df_all['datetime_utc'])
df_all['Year'] = df_all['Date'].dt.year
df_all['Month'] = df_all['Date'].dt.month
df_all['Weekday'] = df_all['Date'].dt.dayofweek
df_all['Hour'] = df_all['Date'].dt.hour

df_all['Lag168'] = np.nan
df_all['Lag336'] = np.nan
for i in range(168, df_all.shape[0]):
    df_all.loc[i, 'Lag168'] = df_all['Prices.BE'].iloc[i - 168]
for i in range(336, df_all.shape[0]):
    df_all.loc[i, 'Lag336'] = df_all['Prices.BE'].iloc[i - 336]
df_all.dropna(axis=0, how='any', inplace=True)
```

Κατά την εξαγωγή τέτοιων μεταβλητών
κάποια από τα **αρχικά δείγματα θα
πρέπει να εξαιρεθούν** από το σετ
εκπαίδευσης.

- ✓ Σε προβλήματα παλινδρόμησης, η **μοντελοποίηση της εποχικότητας** από το ML μοντέλο δεν είναι εύκολη.
- ✓ Η εξαγωγή κάποιων νέων μεταβλητών, οι οποίες περιέχουν παλεθοντικές τιμές (**lags**), βοηθάει εμμέσως στον προσδιορισμό κάποιων μοτίβων.
- ✓ Στην προκειμένη περίπτωση εξετάζονται **lags, 7 και 14 ημερών** τα οποία μπορούν να δώσουν μια καλύτερη αίσθηση για την τιμή, με βάση την αντίστοιχη μέρα και ώρα στο παρελθόν.

Python Example

```
# Split dataset
fh = 168
insample = df_all.iloc[:-168, :]
x_train = np.asarray(insample[selected_features[:-1]])
y_train = np.asarray(insample['Prices.BE'])

outsample = df_all.iloc[-168:, :]
x_test = np.asarray(outsample[selected_features[:-1]])
y_test = np.asarray(outsample['Prices.BE'])
```

- ✓ Οι παρατηρήσεις των τελευταίων 7 ημερών θα χρησιμοποιηθούν για την αξιολόγηση (**evaluation**) του μοντέλου.
- ✓ Οι παρατηρήσεις αυτών των 7 ημερών έχουν εξαιρεθεί από το σετ εκπαίδευσης (**insample**).
- ✓ Αποθηκεύονται σε διαφορετικό Data Frame (**outsample**) για να χρησιμοποιηθούν στην αξιολόγηση.

Python Example

```
# Train an XGBoost Model
```

```
xgb_model = xgb.XGBRegressor(learning_rate=0.1, n_estimators=200, subsample=0.9)
```

```
xgb_model.fit(x_train, y_train)
```

```
xgb_predictions = xgb_model.predict(x_test)
```

Ποσοστό των δειγμάτων
με τα οποία εκπαιδεύεται
κάθε estimator

Βήμα εκπαίδευσης

Μέγιστος αριθμός
weak learners

Μέθοδος που καλείται
για την εκπαίδευση του
μοντέλου με βάση τα
x_train & y_train

Μέθοδος που καλείται
για την παραγωγή
προβλέψεων, για κάθε
sample του x_test

Python Example

```
# Train an XGBoost Model  
xgb_model = xgb.XGBRegressor(learning_rate=0.1, n_estimators=200, subsample=0.9)  
xgb_model.fit(x_train, y_train)  
xgb_predictions = xgb_model.predict(x_test)
```

```
# Train a Random Forest Model  
rf_model = RandomForestRegressor(n_estimators=300)  
rf_model.fit(x_train, y_train)  
rf_predictions = rf_model.predict(x_test)
```

Πλήθος απλών δέντρων (weak learners)
που χρησιμοποιούνται στο δάσος

Μέθοδος που καλείται
για την εκπαίδευση του
μοντέλου με βάση τα
x_train & y_train

Μέθοδος που καλείται
για την παραγωγή
προβλέψεων, για κάθε
sample του x_test

Python Example

```
# Train an XGBoost Model
xgb_model = xgb.XGBRegressor(learning_rate=0.1, n_estimators=200, subsample=0.9)
xgb_model.fit(x_train, y_train)
xgb_predictions = xgb_model.predict(x_test)

# Train a Random Forest Model
rf_model = RandomForestRegressor(n_estimators=300)
rf_model.fit(x_train, y_train)
rf_predictions = rf_model.predict(x_test)

# Print results
xgb_error = np.mean(200 * np.abs(xgb_predictions - y_test) / (np.abs(xgb_predictions) + np.abs(y_test)))
print('XGBoost:', round(xgb_error, 3))
rf_error = np.mean(200 * np.abs(rf_predictions - y_test) / (np.abs(rf_predictions) + np.abs(y_test)))
print('Random Forest:', round(rf_error, 3))
```

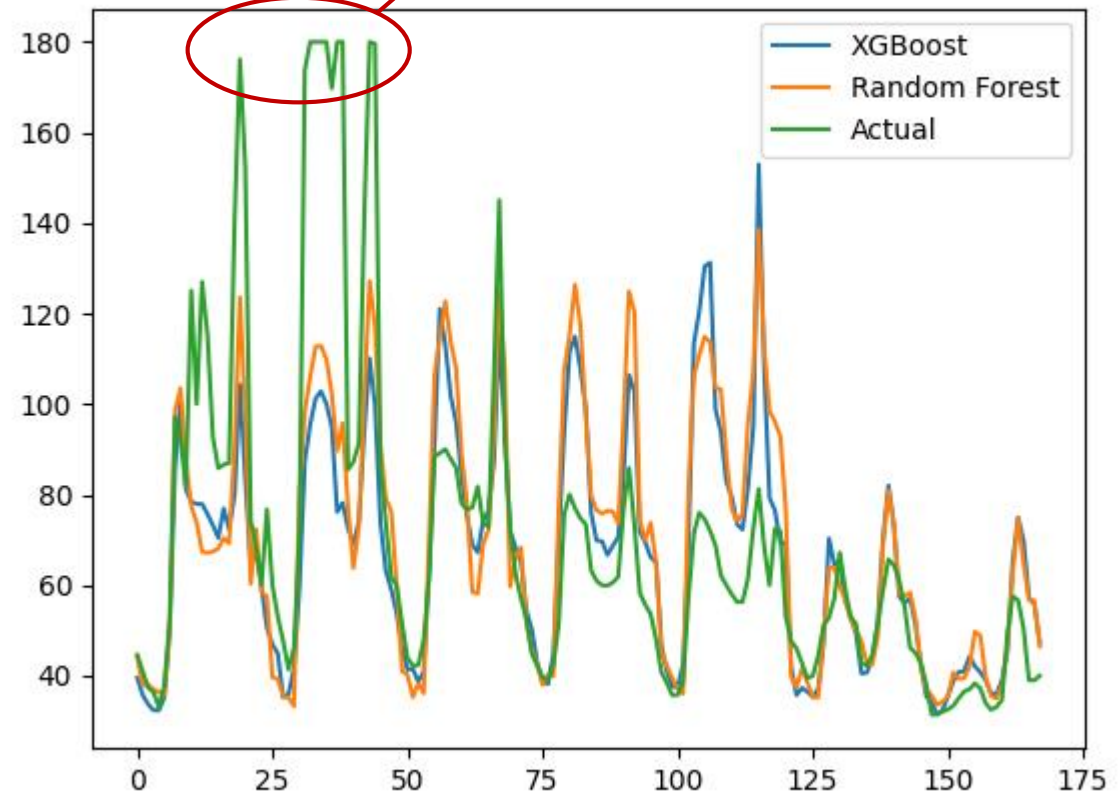
XGBoost: 20.065

Random Forest: 21.224

Python Example

```
# Plot predictions vs reality
plt.plot(xgb_predictions, label='XGBoost')
plt.plot(rf_predictions, label='Random Forest')
plt.plot(y_test, label='Actual')
plt.legend()
plt.show()
```

Τιμές που ψαλιδίστηκαν καθώς
κρίθηκαν ως outliers

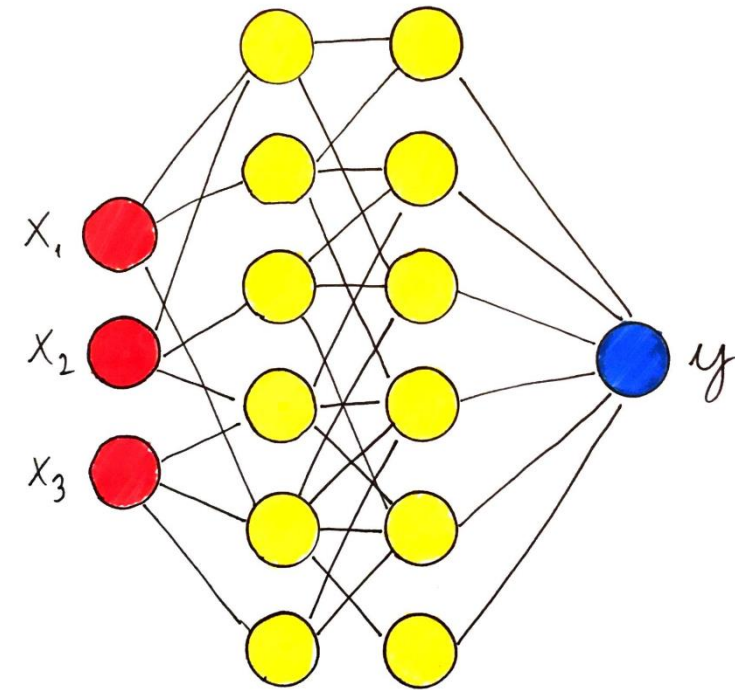


Νευρωνικά Δίκτυα

Neural Networks

Neural Networks

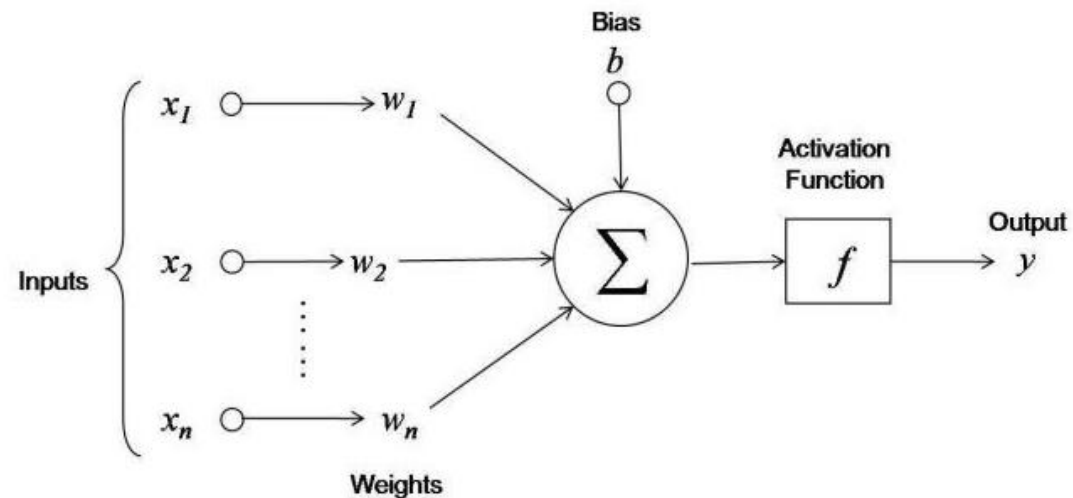
- Τα νευρωνικά δίκτυα (**Neural Networks - NN**) αποτελούν μία από τις πιο διαδεδομένες μεθόδους μηχανικής μάθησης.
- Τα νευρωνικά δίκτυα προσπαθούν να προσδιορίσουν τις παραμέτρους μίας **γενικευμένης μη γραμμικής εξίσωσης** η οποία καλείται να μοντελοποιήσει την σχέση μεταξύ των μεταβλητών εισόδου και εξόδου.
- Λόγω της αύξησης του πλήθους διαθέσιμων δεδομένων, της μεγαλύτερης διαθέσιμης υπολογιστικής ισχύς και της καλύτερης απόδοσης των αλγορίθμων που αξιοποιούνται για την εκπαίδευση και διαχείρισή τους, η **χρήση των νευρωνικών δικτύων αυξήθηκε εκθετικά** τα τελευταία χρόνια.
- Τα νευρωνικά δίκτυα μπορούν να χρησιμοποιηθούν σε εφαρμογές επιβλεπόμενης μάθησης - ταξινόμησης (**classification**) & παλινδρόμησης (**regression**).
- Τα νευρωνικά δίκτυα μπορούν να χρησιμοποιηθούν σε εφαρμογές μη επιβλεπόμενης (**unsupervised**) αλλά και ημι-επιβλεπόμενης (**semi-supervised**) μάθησης.



Neural Networks

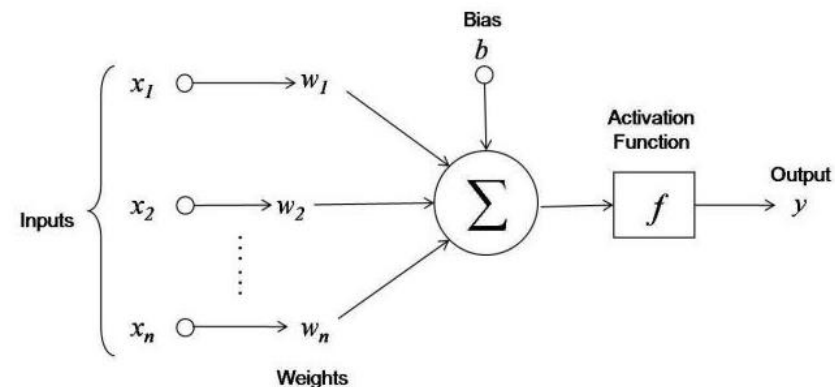
- Τα βασικά στοιχεία με τα οποία χτίζονται τα νευρωνικά δίκτυων είναι οι **νευρώνες (neurons)** και οι συνδέσεις μεταξύ τους.
- Κάθε νευρώνας εκτελεί μια **μη γραμμική γραμμική παλιδρόμηση**.
- Στην πιο απλή περίπτωση νευρώνων (**perceptron**) η έξοδος κάθε νευρώνα δίνεται από την παρακάτω εξίσωση:

$$y = f\left(\sum_{i=1}^n (x_i * w_i) + b\right)$$

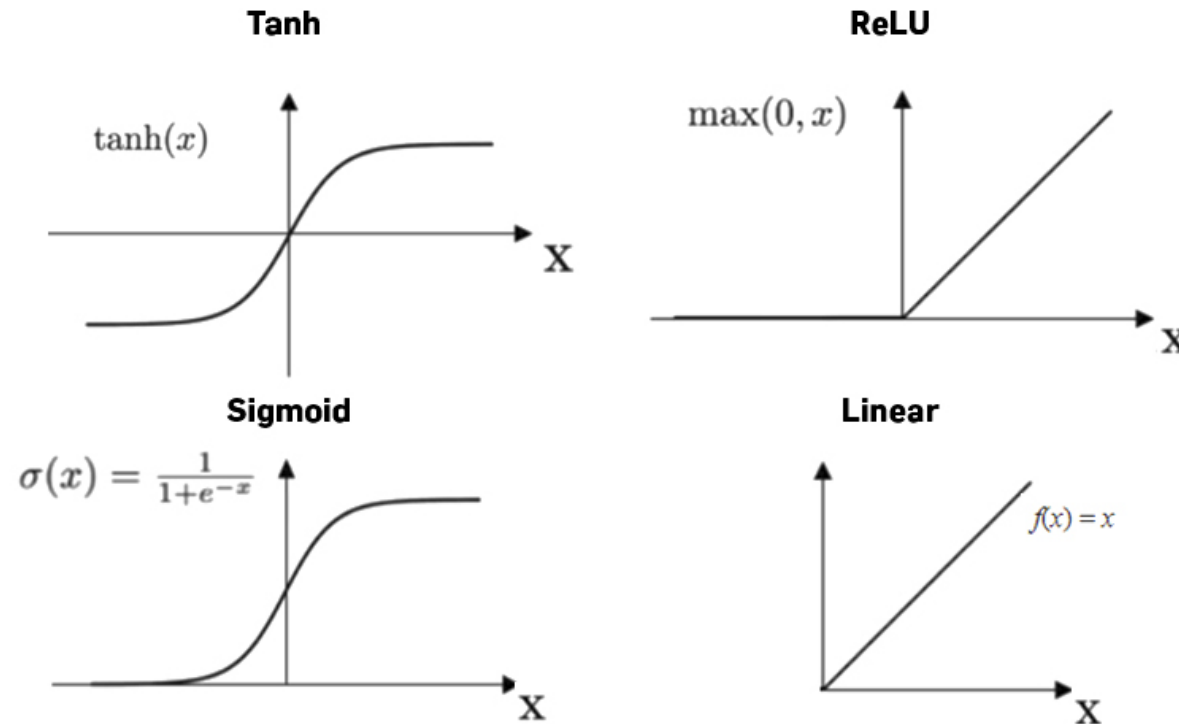


Neural Networks - Activation Function

- Ένα από τα πιο σημαντικά χαρακτηριστικά των νευρώνων είναι η συνάρτηση ενεργοποίησης (**activation function**), η οποία δίνει στο νευρωνικό δίκτυο τη μη γραμμική του συμπεριφορά.
- Η ενεργοποίηση μπορεί να γίνει θεωρητικά μέσω οποιασδήποτε συνάρτησης, με τις πιο δημοφιλείς επιλογές να είναι ωστόσο οι **relu**, **sigmoid** και **tanh**.
- Η συνάρτηση ενεργοποίησης επιλέγεται λαμβάνοντας υπόψιν το πρόβλημα προς επίλυση.



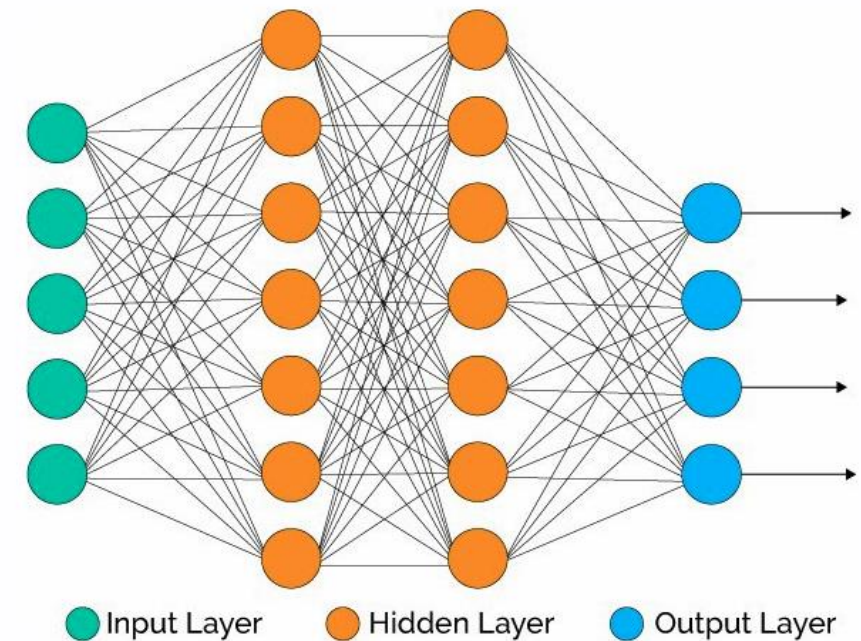
Neural Networks - Activation Function



Αν η γραμμική (linear) συνάρτηση επιλεγεί σαν συνάρτηση ενεργοποίησης για **όλους** τους νευρώνες του δικτύου, **τότε το νευρωνικό δίκτυο χάνει την μη γραμμικότητά του** και η συμπεριφορά του προσεγγίζει αυτή ενός μοντέλου γραμμικής πολλαπλής παλινδρόμησης.

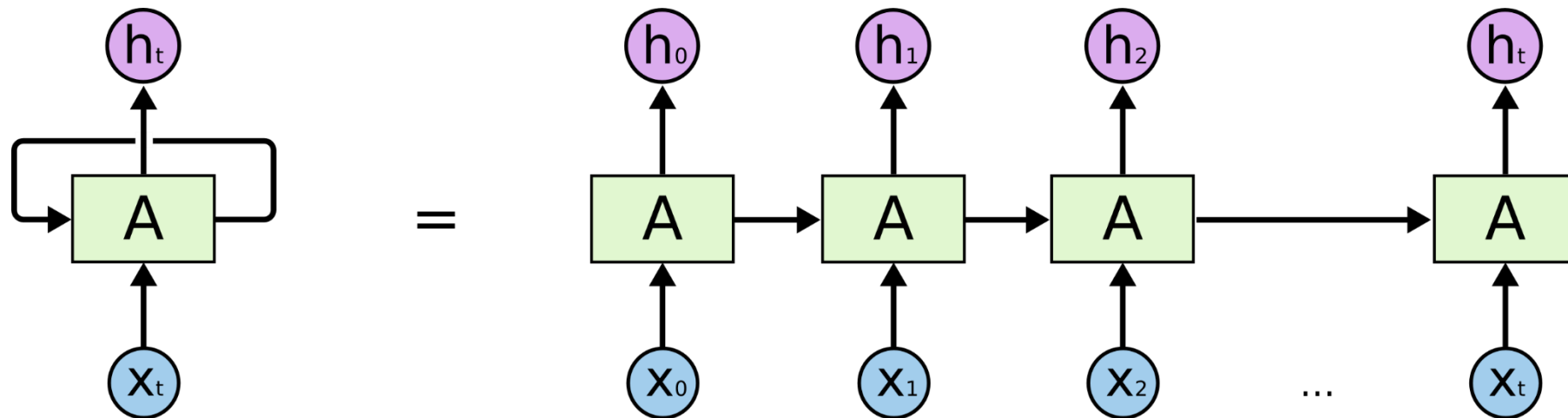
Neural Networks - MLP

- Η πιο απλή κατηγορία νευρωνικών δικτύων αποτελείται αποκλειστικά από perceptrons, τα οποία οργανούνται σε στρώματα (**layers**).
- Νευρωνικά δίκτυα που περιλαμβάνουν περισσότερα τέτοια στρώματα ονομάζονται **Multilayer Perceptrons (MLPs)**.
- Κάθε perceptron τροφοδοτείται με δεδομένα από κάθε perceptron του προηγούμενου layer και τροφοδοτεί με την σειρά του κάθε perceptron του επόμενου layer.
- Ένα **MLP εκπαιδεύεται προσαρμόζοντας τα βάρη** (weights) μεταξύ των perceptrons.



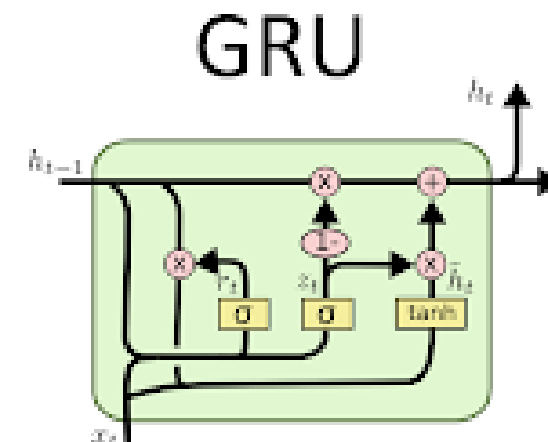
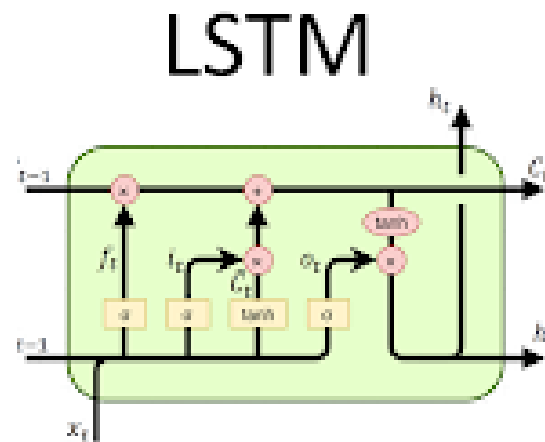
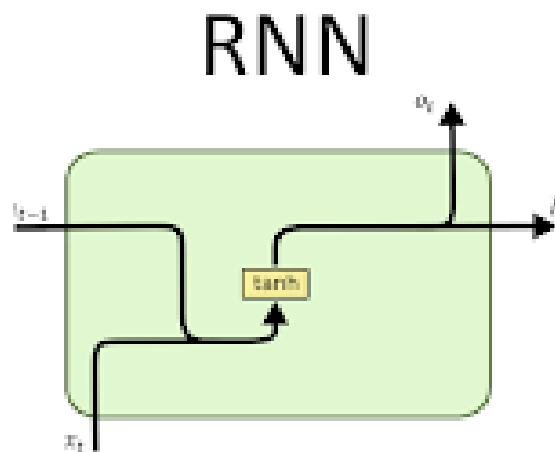
Neural Networks - Recurrent

- Μια πιο σύνθετη κατηγορία νευρώνων σε σχέση με τα απλά perceptrons είναι οι **νευρώνες με δυνατότητα «μνήμης»**.
- Η λειτουργία της μνήμης υλοποιείται με χρήση ενός **μηχανισμού ανατροφοδότησης**.
- Δίκτυα τα οποία αποτελούνται από τέτοιους νευρώνες ονομάζονται ανατροφοδοτούμενα (**recurrent networks**).



Neural Networks - Recurrent

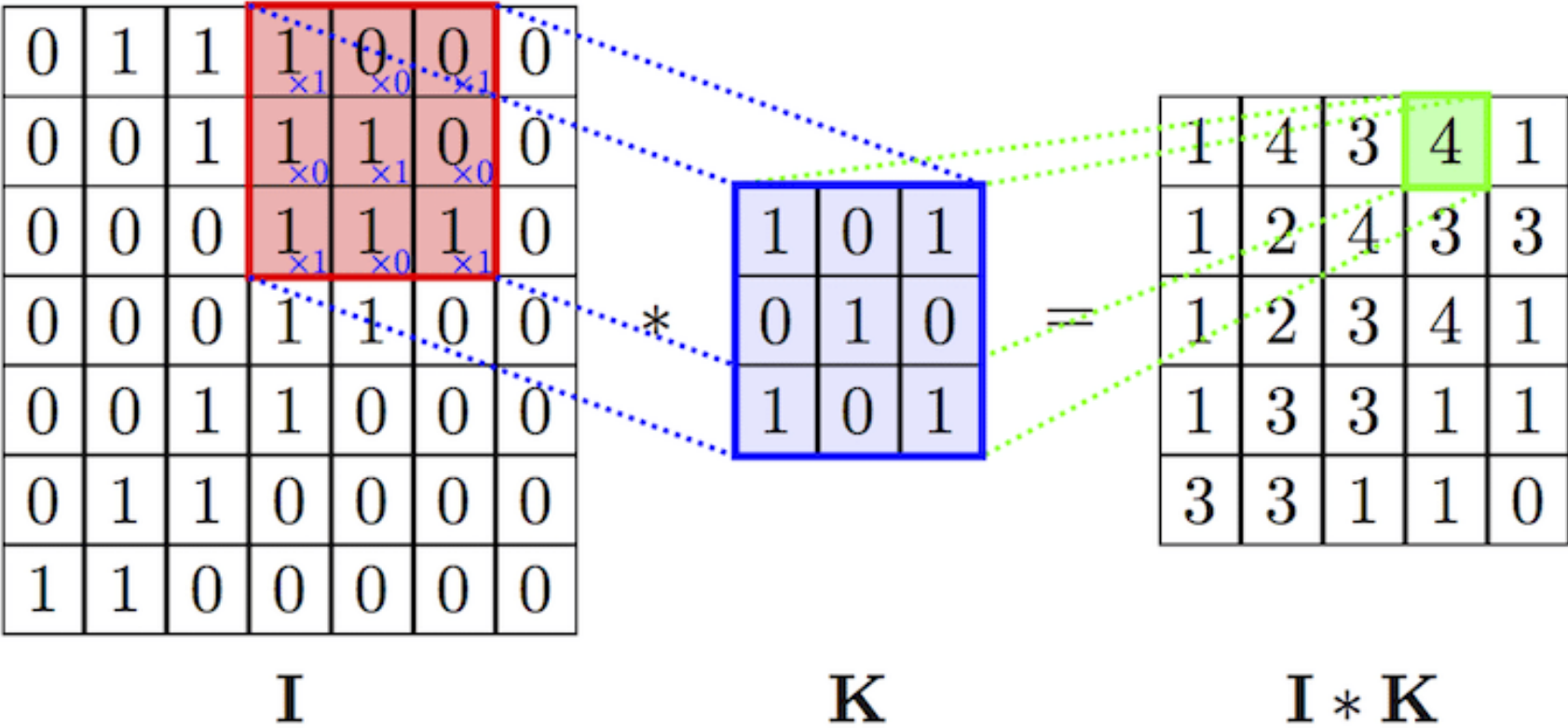
- Τα Recurrent Neural Networks (RNN) είναι σχεδιασμένα να μοντελοποιούν δεδομένα για οποία η **έννοια του χρόνου είναι σημαντική**.
- Αποτελούν **δημοφιλή επιλογή** για εφαρμογές πρόβλεψης χρονοσειρών με νευρωνικά δίκτυα.
- Υπάρχουν διαφορετικές επιλογές ως προς τον τύπο των recurrent νευρώνων, οι οποίες **διαφοροποιούνται ως προς τον μηχανισμό μνήμης** που διαθέτουν.



Neural Networks - Convolutional

- Μια διαφορετική προσέγγιση στην δομή και την λειτουργία των layers προτείνουν τα συνελκτικά δίκτυα (**convolutional networks**)
- Τα layers δεν αποτελούνται από νευρώνες αλλά από ένα **σετ «φίλτρων»** τα οποία «σκανάρουν» και μετασχηματίζουν το διάλυμα της εισόδου.
- Τα convolutional δίκτυα έγιναν ευρύτερα γνωστά λόγω της επιτυχίας τους σε **εφαρμογές επεξεργασίας και αναγνώρισης εικόνων** (VGG, ResNet, κλπ).
- Η χρήση τους σε εφαρμογές πρόβλεψης χρονοσειρών είναι ακόμα περιορισμένη.

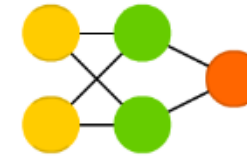
Neural Networks - Convolutional



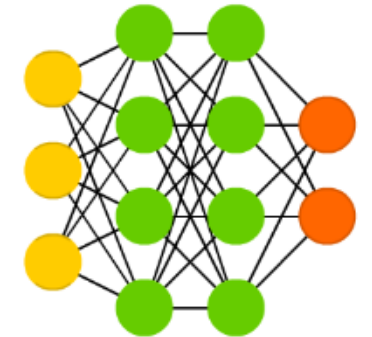
Neural Networks - Architecture

- ✓ Ο καθορισμός της «**αρχιτεκτονικής**» του νευρωνικού δικτύου αποτελεί την πρώτη καθοριστική επιλογή που πρέπει να κάνει κάποιος.
- ✓ Η βέλτιστη αρχιτεκτονική εξαρτάται από την **φύση του προβλήματος** καθώς και την **διαθεσιμότητα των δεδομένων**.
- ✓ Η αρχιτεκτονική του δικτύου αφορά:
 - Το πλήθος των hidden layers
 - Το πλήθος των nodes στα hidden layers
 - Τον τρόπο διασυνδέσης των layers
 - Χαρακτηριστικά που σχετίζονται με τον σχεδιασμό των nodes

Feed Forward

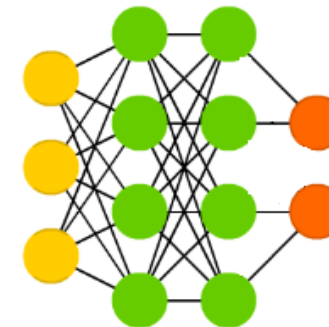


Deep Feed Forward

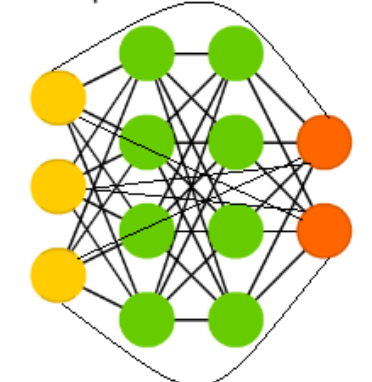


Fully connected

Deep Feed Forward



Deep Feed Forward



With shortcuts

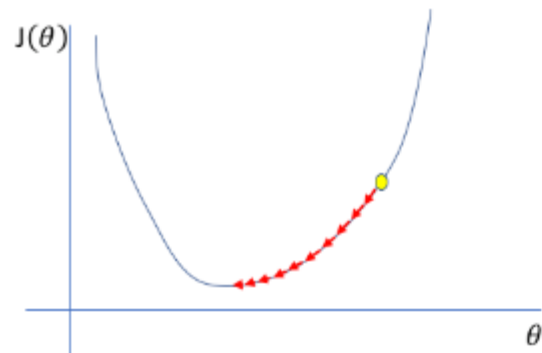
Neural Networks - Training

- Κατά την διαδικασία της εκπαίδευσης ενός νευρωνικού δικτύου βελτιστοποιούνται οι **τιμές των βαρών** που συνδέουν τους νευρώνες καθώς και η **τιμή του όρου bias** καθέ νευρώνα.
- Ανάλογα με την φύση του προβλήματος, επιλέγεται η κατάλληλη **συνάρτηση για τον υπολογισμό του σφάλματος** του νευρωνικού δικτύου.
- Κατά την εκπαίδευση, κάθε είσοδος (input) παρουσιάζεται στο δίκτυο και από αυτήν υπολογίζεται κάποιο αποτέλεσμα (output). Αυτό συγκρίνεται με την αναμενόμενη έξοδο, και **υπολογίζεται το σφάλμα εκπαίδευσης**.
- Βάσει του σφάλματος, **τα βάρη μεταβάλλονται** κατά μία ποσότητα ούτως ώστε η ακρίβεια του δικτύου να βελτιωθεί.
- Η ενημέρωση των τιμών των βαρών ξεκινάει από τα βάρη που βρίσκονται πιο κοντά στην έξοδο του δικτύου και καταλήγει σε αυτά πιο κοντά στην είσοδό του (**backpropagation**)

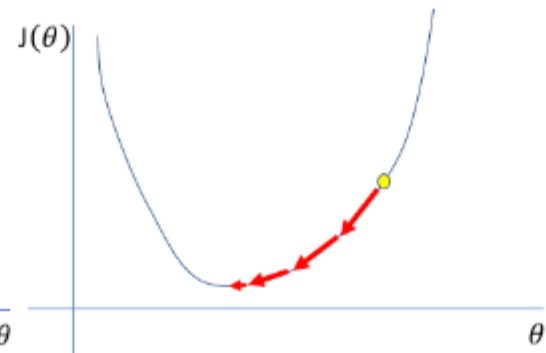
Neural Networks - Training

- Καθοριστικό ρόλο στην διαδικασία της εκμάθησης παίζει ο συντελεστής εκμάθησης (**learning rate**) που έχει επιλεχθεί.

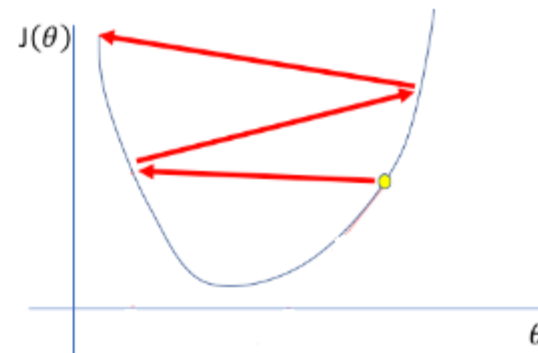
$$w(k) = w(k - 1) + \beta(d - y)x$$



Μεγάλοι χρόνοι εκτέλεσης και κίνδυνος να οδηγηθούμε σε τοπικά ελάχιστα



Σταδιακή μετάβαση στο βέλτιστο

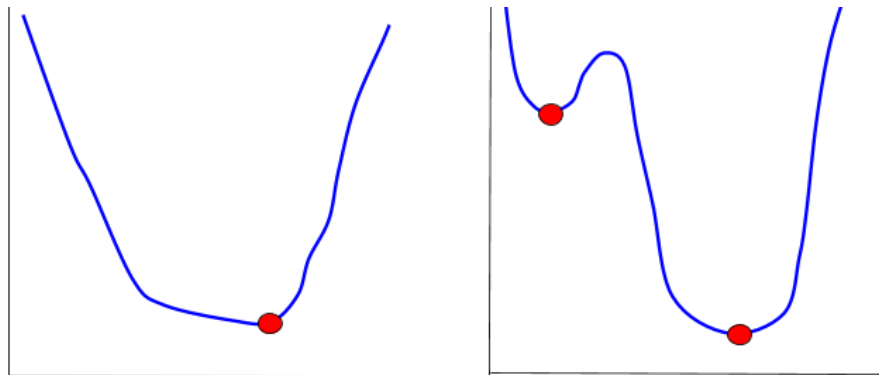


Απότομες μεταβολές που δυσκολεύουν τη σύγκλιση και μπορεί να οδηγήσουν σε μη βέλτιστες λύσεις

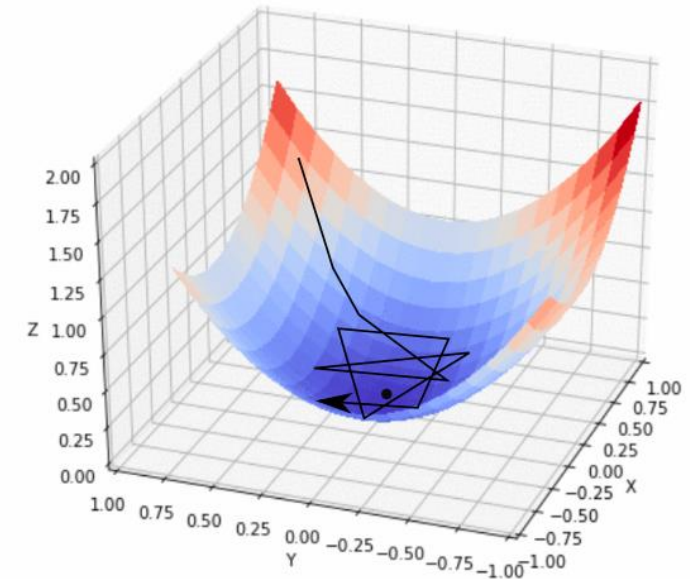
- Προτιμάται η επιλογή προσαρμοζόμενου συντελεστή εκμάθησης (**adaptive learning rate**) ο οποίος συνδυάζει γρήγορη σύγκλιση και ακρίβεια.

Neural Networks - Training

- Στην γενική περίπτωση η συνάρτηση του σφάλματος σε σχέση με τις παραμέτρους του δικτύου είναι αρκετά πιο σύνθετη και η **εύρεση του ολικού ελάχιστου είναι δύσκολη**.
- Οι **αρχικοποίηση των βαρών του δικτύου παίζει σημαντικό ρόλο** στην σύγκλιση των παραμέτρων.
- Η επιλογή κατάλληλου learning rate είναι ιδιαίτερα σημαντική.



Προβλήματα βελτιστοποίησης με
ολικά και μερικά ελάχιστα



Neural Networks - Training

- Συνήθως ο υπολογισμός των σφαλμάτων εκπαίδευσης και η ανανέωση των βαρών με βάση αυτό δεν γίνεται ξεχωριστά για κάθε ζευγάρι εισόδου-εξόδου των δεδομένων.
- Τα δεδομένα της εκπαίδευσης οργανούνται σε πακέτα (**batches**) για κάθε ένα από τα οποία υπολογίζεται ένα σφάλμα εκπαίδευσης.
- Η πρόβλεψη ενός batch δεδομένων, ο υπολογισμός του σφάλματος του και η ανανέωση των βαρών του δικτύου με βάση αυτό, ολοκληρώνουν ένα βήμα εκπαίδευσης (**training step / iteration**).
- Όταν όλα τα διαθέσιμα batches περάσουν από την διαδικασία της εκπαίδευσης, και το δίκτυο έχει «δει» όλα τα διαθέσιμα δεδομένα μια φορά, ολοκληρώνεται μια εποχή (**epoch**) εκπαίδευσης.
- Συνήθως η διαδικασία εκπαίδευσης ενός νευρωνικού δικτύου περιλαμβάνει **περισσότερες από μια εποχές εκπαίδευσης**.

Neural Networks - Training

Η επιλογή του κατάλληλου μεγέθους για τα batches είναι σημαντική

- Στην πρώτη περίπτωση που επιλεγεί πολύ **μικρό μέγεθος batch** η εκπαίδευση είναι πιο **αργή**, με υψηλό ταυτόχρονα **ρίσκο υπερ-προσαρμογής** (overfit).
- Στην περίπτωση που επιλεγεί πολύ **μεγάλο μέγεθος batch** η εκπαίδευση του δικτύου γίνεται με σχετικά **μικρότερη ακρίβεια**.

Κριτήρια για την επιλογή του κατάλληλου μεγέθους των batches είναι:

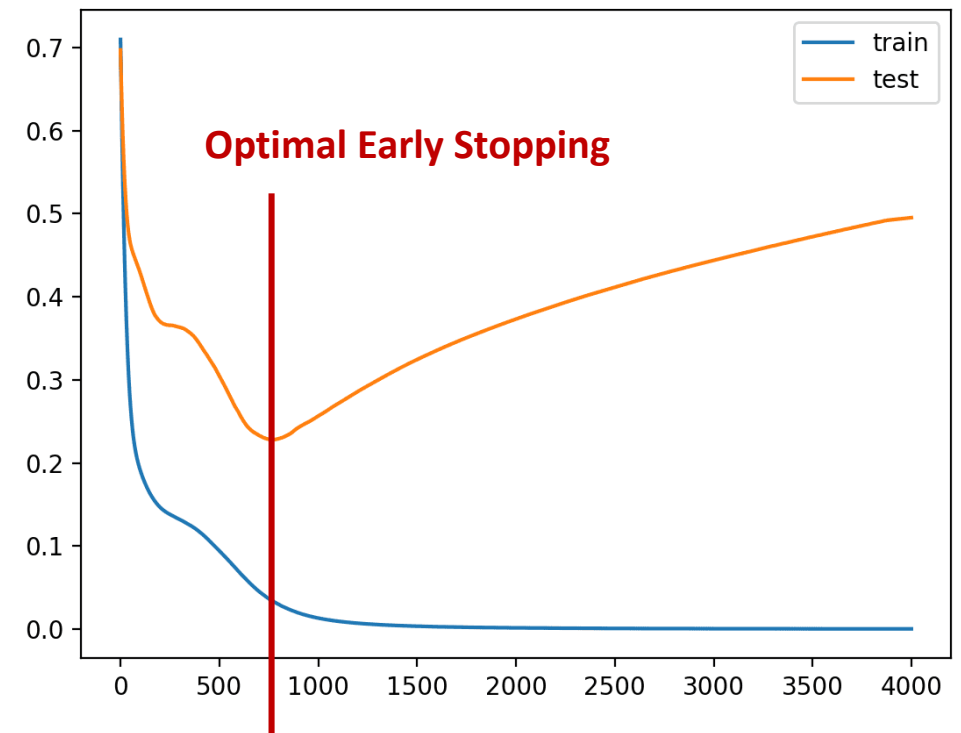
- Το **πλήθος των δεδομένων** εκπαίδευσης
- Η **φύση των δεδομένων** εκπαίδευσης
- Η **πολυπλοκότητα του δικτύου** που εκπαιδεύεται
- Οι διαθέσιμοι **υπολογιστικοί πόροι**

Neural Networks - Training

- ✓ Για την αποτελεσματική εκπαίδευση ενός νευρωνικού δικτύου απαιτούνται περισσότερες από μια εποχές εκπαίδευσης.
- ✓ Ωστόσο, **δεν μπορούμε να γνωρίζουμε εξαρχής το απαιτούμενο πλήθος των εποχών εκπαίδευσης** και πρέπει να τον προσδιορίσουμε εμπειρικά.

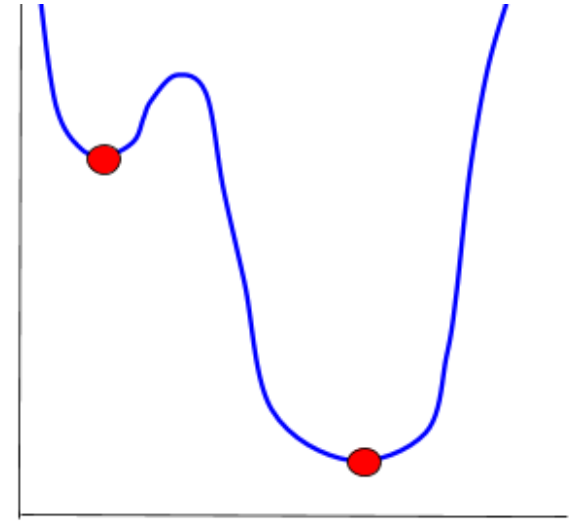
Συνήθως επιλέγουμε να τερματίζουμε τη διαδικασία

- Μετά από K εποχές, όπου K ο αριθμός της εποχής μετά το πέρας της οποίας η μείωση του σφάλματος εκπαίδευσης (loss) ήταν αμελητέα – **early stopping**.
- Μετά από K_{max} εποχές, όπου K_{max} ένα **ανώτατο προκαθορισμένο όριο εποχών** - χρησιμοποιείται κυρίως σε περιπτώσεις όπου δεν επιτυγχάνεται σύγκλιση.



Ensembles of models

- Κοινό χαρακτηριστικό των περισσότερων μεθόδων ML αποτελεί ο **μη-ντετερμινιστικός χαρακτήρας** τους.
- Η **τυχειότητα** κατά την αρχικοποίηση αλλά και την εκπαίδευση των ML μοντέλων παίζει σημαντικό ρόλο στην τελική τους απόδοση.



Πώς επιλέγω το πιο «τυχερό» μοντέλο για τις προβλέψεις μου?

Ensembles of models

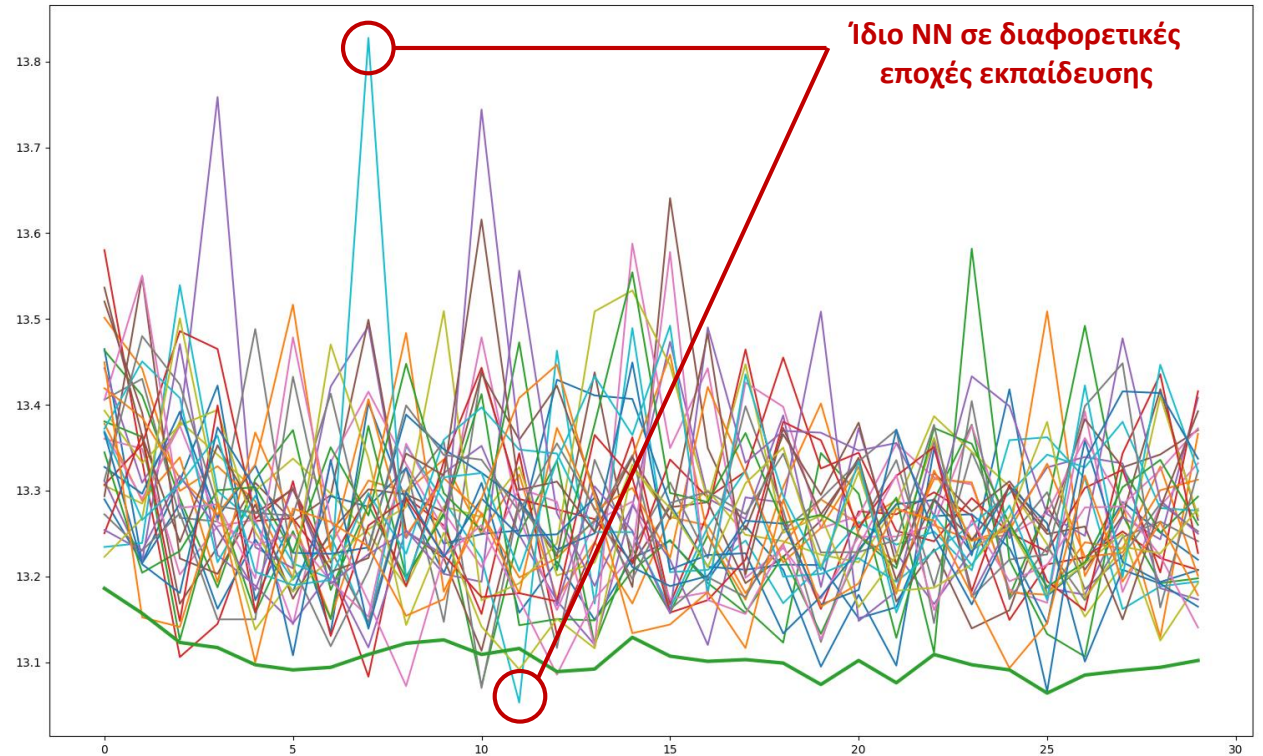
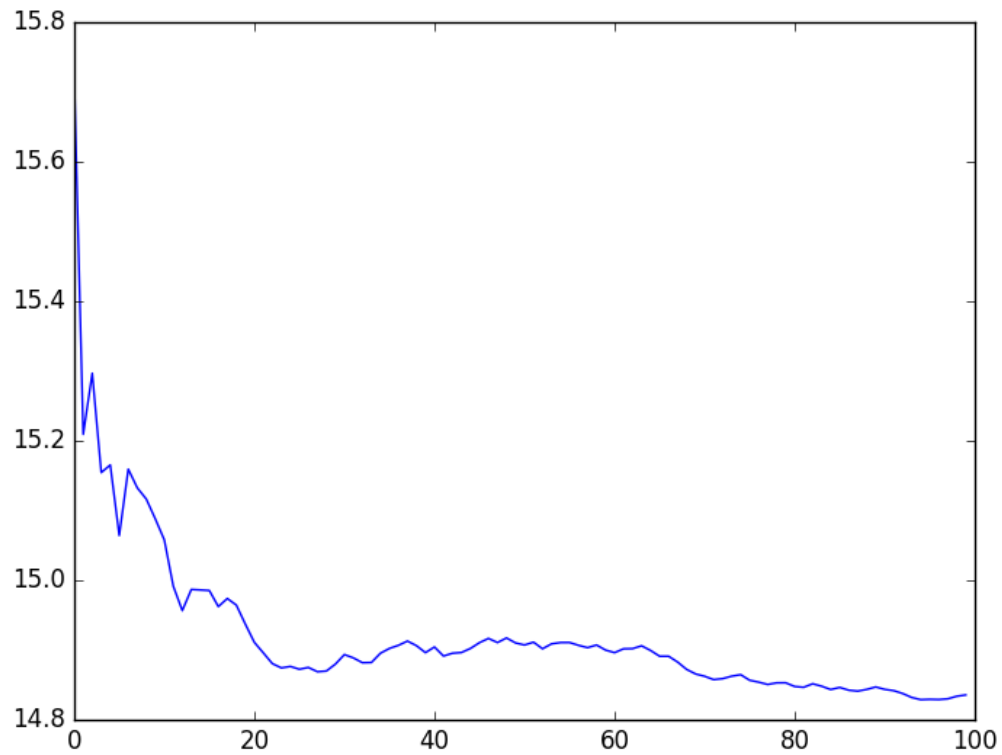
Πώς επιλέγω το πιο «τυχερό» μοντέλο για τις προβλέψεις μου?

- ✓ Η επιλογή του μοντέλου με βάση την απόδοση του κατά την εκπαίδευση κρύβει κινδύνους - **overfit**.
- ✓ Κάθε μοντέλο **συμπεριφέρεται διαφορετικά** όταν καλείται να προβλέψει ένα **διαφορετικό σετ δεδομένων**.

Ο πιο αποτελεσματικός τρόπος για την εξάλειψη της επίδρασης της τυχαιότητας είναι ο συνδυασμός (**ensemble**) προβλέψεων πολλών μοντέλων

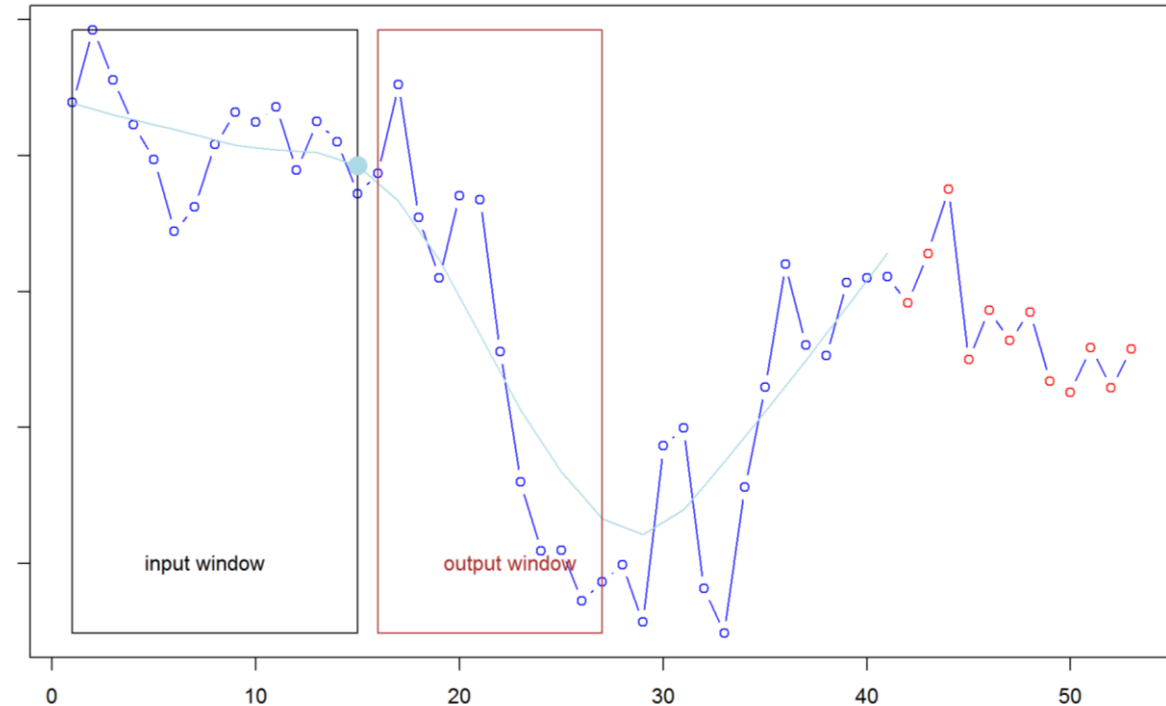
- ✓ Κάθε μοντέλο **αρχικοποιείται και εκπαιδεύεται τυχαία** με επιλογή διαφορετικού random seed.
- ✓ Οι τελικές προβλέψεις προκύπτουν ως ο συνδυασμός των μεμονομένων προβλέψεων με χρήση μέσου όρου (**mean**) ή της διάμεσου (**median**) αυτών.

Ensembles of models



Forecasting with NN - Inputs

- ✓ Για την πρόβλεψη χρονοσειρών, οι είσοδοι (inputs) του NN περιλαμβάνουν **παρελθοντικές τιμές (lags)** της χρονοσειράς η οποία προβλέπεται.
- ✓ Το NN αποτελεί ουσιαστικά ένα **μη γραμμικό auto-regression μοντέλο** όπου κάθε μελλοντική παρατήρηση $Y_t, Y_{t+1}, \dots, Y_{t+h}$ προβλέπεται βάσει των παρελθοντικών της παρατηρήσεων $Y_{t-1}, Y_{t-2}, \dots, Y_{t-k}$.
- ✓ Ο αριθμός των παρελθοντικών τιμών που λαμβάνονται υπόψιν (**look-back window**) είναι μια ιδιαίτερα σημαντική παράμετρος του δικτύου.
- ✓ Αξίζει να σημειωθεί πως οι είσοδος του NN **μπορεί να περιλαμβάνει και άλλες μεταβλητές** (features), πέρα από τα lags



Forecasting with NN - Outputs

Ο ορίζοντας πρόβλεψης H καθορίζεται από την φύση του προβλήματος που βρίσκεται μπροστά μας. Ωστόσο, υπάρχουν **διαφορετικές στρατηγικές για την παραγωγή προβλέψεων**, που οδηγούν στην επίλυση του προβλήματος.

Iterative

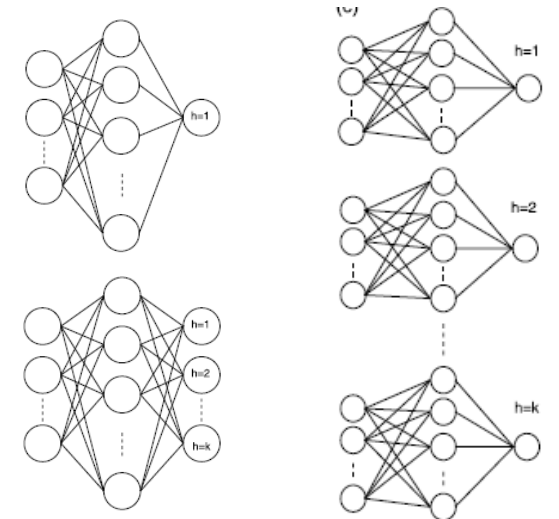
- Το κάθε μοντέλο NN εκπαιδεύεται στην παραγωγή **one-step-ahead προβλέψεων**.
- Το NN διαθέτει μόνο **έναν κόμβο εξόδου**.
- Κάθε παραγόμενη **πρόβλεψη γίνεται μέρος της εισόδου** που θα οδηγήσει στην επόμενη και
- Η **διαδικασία επαναλαμβάνεται H φορές** για την παραγωγή H προβλέψεων.

Direct

- Το κάθε μοντέλο NN εκπαιδεύεται στην **παραγωγή όλων των προβλέψεων** ταυτόχρονα.
- Το NN διαθέτει **H κόμβους εξόδου**.
- Η **διαδικασία πρόβλεψης γίνεται σε ένα βήμα** για την παραγωγή H προβλέψεων.

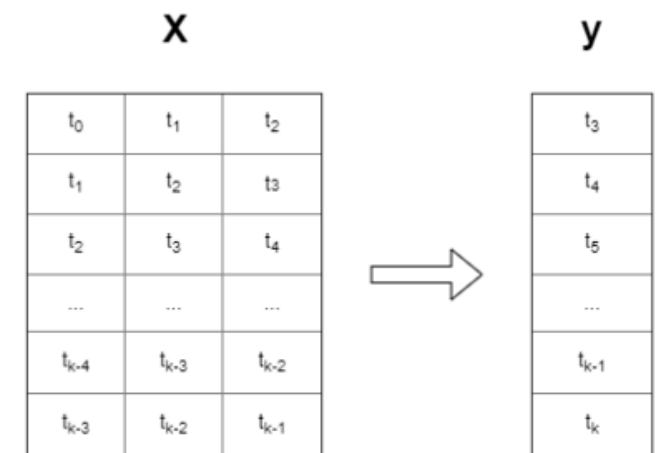
Multi-NN

- Το κάθε μοντέλο NN εκπαιδεύεται στην μίας πρόβλεψης, που αντιστοιχεί σε **ένα σημείο του ορίζοντα H** .
- Το κάθε NN διαθέτει **έναν κόμβο εξόδου**.
- Απαιτείται **εκπαίδευση H διαφορετικών NN** για να καλυφθεί ο ζητούμενος ορίζοντας πρόβλεψης.



Forecasting with NN - Training dataset

- ✓ Η διαδικασία της εκπαίδευσης (**model training**) ενός ML μοντέλου είναι ιδιαίτερα σημαντική.
- ✓ Η κατασκευή του σετ δεδομένων με το οποίο θα γίνει η εκπαίδευση (**training set**) παίζει σημαντικό ρόλο στην τελική ακρίβεια του μοντέλου.
- ✓ Η εκπαίδευση γίνεται με δείγματα (**training samples**) που περιλαμβάνουν την είσοδο και την αναμενόμενη έξοδο, βάσει των ιστορικών στοιχείων.
- ✓ Ιδιαίτερα για την εκπαίδευση NN, απαιτείται κανονικοποίηση (**standardization** ή **scaling**) των δεδομένων.
- ✓ Σε προβλήματα πρόβλεψης χρονοσειρών υπάρχουν διαφορετικές προσεγγίσεις κατά την κατασκευή του training set ανάλογα:
 - Με τον όγκο των διαθέσιμων δεδομένων
 - Την φύση των διαθέσιμων δεδομένων
 - Την φύση του προβλήματος
 - Τον πολυπλοκότητα του NN που χρησιμοποιείται
 - Τους διαθέσιμους υπολογιστικούς πόρους



Forecasting with NN - Training dataset

Localized training

- ✓ Το μοντέλο **εκπαιδεύεται βάσει των δεδομένων μιας χρονοσειράς** την οποία και μαθαίνει να προβλέπει. Τα training samples δημιουργούνται με την τεχνική των κυλιόμενων παραθύρων (**rolling windows**).
- ✓ Πρόκειται για την **«παραδοσιακή» προσέγγιση** στην ανάπτυξη NN για προβλέψεις χρονοσειρών.
- ✓ Χρησιμοποιείται ακόμα με **επιτυχία σε ορισμένες εφαρμογές** προβλέψεων.



- + Όταν απαιτείται πρόβλεψη μικρού αριθμού χρονοσειρών, η εκπαίδευση είναι γρήγορη.
- + Τα NN λαμβάνουν υπόψιν τους τις ιδιαιτερότητες της χρονοσειράς την οποία προβλέπουν.
- + Δεν υπάρχει ανάγκη για συλλογή και επεξεργασία ενός μεγάλου συνόλου χρονοσειρών.
- Όταν οι χρονοσειρές δεν περιέχουν μεγάλο αριθμό παρατηρήσεων, η εκπαίδευση δεν είναι αποτελεσματική.
- Τα NN δεν μπορούν να εκμεταλλευτούν γνώση από άλλες, παρόμοιες χρονοσειρές.
- Δεν είναι εύκολο να επεκταθεί όταν απαιτείται η παραγωγή προβλέψεων για μεγάλο αριθμό χρονοσειρών.

Forecasting with NN - Training dataset

Global training

- ✓ Το μοντέλο **εκπαιδεύεται βάσει των δεδομένων ενός μεγάλου σετ χρονοσειρών**.
- ✓ Κάθε χρονοσειρά μπορεί να **συνεισφέρει ένα ή περισσότερα** training samples.
- ✓ Πρόκειται για μια πιο **μοντέρνα προσέγγιση** στην ανάπτυξη NN για προβλέψεις χρονοσειρών.

- + Τα NN μπορούν να εκμεταλλευτούν γνώση από άλλες, παρόμοιες χρονοσειρές.
- + Δεν απαιτούνται χρονοσειρές με μεγάλο αριθμό ιστορικών παρατηρήσεων.
- + Ένα NN εκπαιδευμένο με αυτόν τον τρόπο μπορεί να παρέχει άμεσα προβλέψεις για μεγάλο αριθμό χρονοσειρών.

- Χρειάζονται σημαντικοί υπολογιστικοί πόροι για την εκπαίδευση, ιδιαίτερα όταν το μέγεθος του training set είναι μεγάλο.
- Τα NN δίνουν μικρότερη σημασία στις ιδιαιτερότητες κάποιων χρονοσειρών.
- Ανάλογα με το πρόβλημα που επιλύεται θα πρέπει να επιλέγονται κατάλληλα σετ χρονοσειρών.

Forecasting with NN - Hyper parameters

Η επιλογή της κατάλληλης αρχιτεκτονικής για τα NN της πρόβλεψης είναι μια κρίσιμη αλλά σύνθετη διαδικασία.

- Ο αριθμός των παρελθοντικών τιμών (lags), καθώς και η απόφαση χρήσης ή μη επιπλέον ανεξάρτητων μεταβλητών (features) καθορίζει το μέγεθος της εισόδου του δικτύου (**input layer**).
- Ο ορίζοντας πρόβλεψης H , καθώς και η επιλογή στρατηγικής για την παραγωγή των προβλέψεων, καθορίζουν τον αριθμό των νευρώνων στην έξοδο του NN (**output layer**).
- Σε εφαρμογές προβλέψεων, για τους νευρώνες των κρυφών στρωμάτων (**hidden layers**), ως συνάρτηση ενεργοποίησης (**activation function**) συνήθως επιλέγεται η Relu.
- Για τους νευρώνες της εξόδου, ως συνάρτηση ενεργοποίησης η γραμμική (**linear/identity**).
- Για τον αριθμό των κρυφών στρωμάτων (**number of hidden layers**), το μέγεθος αυτών (**hidden layer size**), τον τύπο των nodes (**perceptrons, recurrent, convolutional**) καθώς και για τις υπόλοιπες παραμέτρους δεν υπάρχει κάποιος αυστηρός κανόνας.

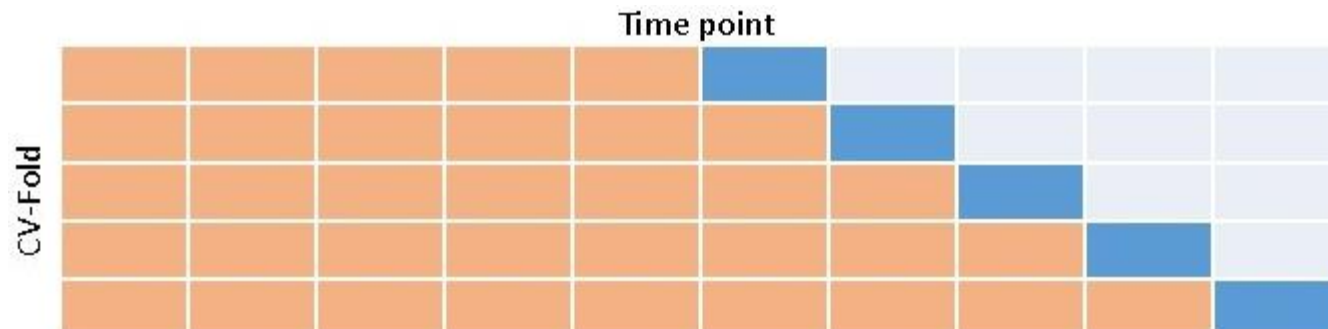
Forecasting with NN - Hyper parameters

Η παραμετροποίηση της εκπαίδευσης έχει και αυτή σημασία.

- Για την αξιολόγηση του σφάλματος εκπαίδευσης (train loss) επιλέγεται ένα εκ των σφαλμάτων παλινδρόμησης (**regression losses**). Συνηθισμένες επιλογές είναι τα MAE, MSE και sMAPE.
- Ως προς τον αλγόριθμο βελτιστοποίησης του δικτύου (**optimizer**), συνηθισμένες επιλογές είναι αυτές του Adam και του κλασσικού SGD.
- Η χρήση **early stopping** γενικά προτείνεται ώστε να αποφευχθούν φαινόμενα overfit αλλά και για εξοικονόμηση χρόνου.
- Τιμές για παραμέτρους όπως το μέγεθος των batches (**batch size**), ο συντελεστής εκμάθησης (**learning rate**), καθώς και άλλες, θα πρέπει να επιλέγονται ανάλογα με την εφαρμογή.

Hyper parameter optimization

- ✓ Ιδιαίτερα σε ότι αφορά τα NN, υπάρχει **μεγάλος αριθμός υπερ-παραμέτρων** για τις οποίες πρέπει να επιλεγεί η βέλτιστη τιμή.
- ✓ Συνήθως, οι υπερ-παραμέτροι επηρεάζουν άμεσα η μία την άλλη. Συνεπώς η βελτιστοποίηση των τιμών τους **δεν μπορεί να γίνεται σειριακά**, αλλά ταυτόχρονα και συνδυαστικά.
- ✓ Για την αξιολόγηση των διαφορετικών τιμών των υπερ-παραμέτρων θα πρέπει να χρησιμοποιείται κάποιας μορφής **validation**.
- ✓ Υπάρχουν διαφορετικές **τεχνικές βελτιστοποίησης**:
 - ✓ Grid search
 - ✓ Random search
 - ✓ Bayesian optimization



ML in Forecasting

- Οι αλγόριθμοι ML **άργησαν να γίνουν αποδεκτοί** από την κοινότητα των προβλέψεων
- Τα προβλήματα που σχετίζονται με την παραγωγή προβλέψεων παρουσιάζουν **ιδιαίτερες δυσκολίες** τις οποίες τα μοντέλα ML δεν ήταν σε θέση να λύσουν.

Ο **διαγωνισμός προβλέψεων M3**, το 2000, είναι ενδεικτικός για την θέση του ML στον χώρο των προβλέψεων:

- Συνολικά **24 μέθοδοι αξιολογήθηκαν** σε ένα σύνολο 3003 χρονοσειρών.
- Μόνο **μία συμμετοχή** έκανε χρήση μηχανικής μάθησης.
- Οι προβλέψεις του νευρωνικού δικτύου που συμμετείχε ήταν **λιγότερο ακριβείς** από αυτές άλλων στατιστικών μεθόδων.

ML in Forecasting

- Από το 2000, υπήρξαν **σημαντικές εξελίξεις** στον χώρο της μηχανικής μάθησης.
- Αλγόριθμοι ML άρχισαν να **χρησιμοποιούνται ευρύτερα** από εταιρείες.

Το 2007, ο **διαγωνισμός προβλέψεων NN3**, επιχείρησε να αξιολογήσει ξανά τις ML μεθόδους πρόβλεψης:

- Συνολικά **59 μέθοδοι αξιολογήθηκαν** σε ένα σύνολο 111 χρονοσειρών.
- 46 από τις συμμετοχές έκαναν χρήση μηχανικής μάθησης.
- Αν και οι ML μέθοδοι εμφάνισαν κάποια βελτίωση, τα στατιστικά μοντέλα συνέχισαν να παρουσιάζουν μεγαλύτερη ακρίβεια πρόβλεψης.

ML in Forecasting

Το 2018, ο **διαγωνισμός προβλέψεων M4**, άλλαξε οριστικά την αντίληψη της κοινότητας γύρω από την χρήση ML μεθόδων:

- Αν και οι καθαρά ML μέθοδοι δεν εντυπωσίασαν με την απόδοσή τους, έγινε σαφές ότι η χρήση μηχανικής μάθησης προσφέρει κάποια πλεονεκτήματα.
 - Οι πρώτες δύο μέθοδοι έκαναν χρήση ML σε συνδυασμό με παραδοσιακές μεθόδους πρόβλεψης.
-
- ✓ Πρέπει να σημειωθεί πως σε συγκεκριμένες εφαρμογές προβλέψεων, αλγόριθμοι ML ήδη χρησιμοποιούνταν με μεγαλύτερη επιτυχία για κάποια χρόνια.
 - ✓ Μεγάλες εταιρείες (πχ Amazon) είχαν ήδη ξεκινήσει να αναπτύσουν ML μοντέλα πρόβλεψης

Hybrid Models - ESRNN

- Μια εναλλακτική προσέγγιση για την δημιουργία μοντέλων πρόβλεψης είναι η μείξη στατιστικών και ML μοντέλων και η δημιουργία υβριδίων (**hybrids**).
- Το πλεονέκτημα αυτής της προσέγγισης βρίσκεται στο ότι κάθε κατηγορία μοντέλων **καλύπτει τις αδυναμίες** της άλλης.
- Η μέθοδος που κατέλαβε **την πρώτη θέση στον διαγωνισμό προβλέψεων M4**, αναπτύχθηκε από τον Slawek Smyl, και ήταν ένα τέτοιο υβρίδιο.
- Η μέθοδος του συνδύαζε την μέθοδο εκθετικής εξομάλυνσης **Holt-Winters** με Recurrent Neural Networks (**RNN**).

$$l_t = \alpha(y_t/s_t) + (1 - \alpha)l_{t-1}$$
$$s_{t+m} = \gamma(y_t/l_t) + (1 - \gamma)s_t$$

Εκθετική εξομάλυνση
Μοντελοποίηση επιπέδου και εποχιακότητας

$$\hat{y}_{t+1..t+h} = RNN(X_t) * l_t * s_{t+1..t+h}$$

RNN
Μοντελοποίηση της τάσης

Ένα επιπλέον ενδιαφέρον σημείο της μεθόδου του Smyl είναι ο συνδυασμός διαφορετικών προσεγγίσεων για τον προσδιορισμό των παραμέτρων:

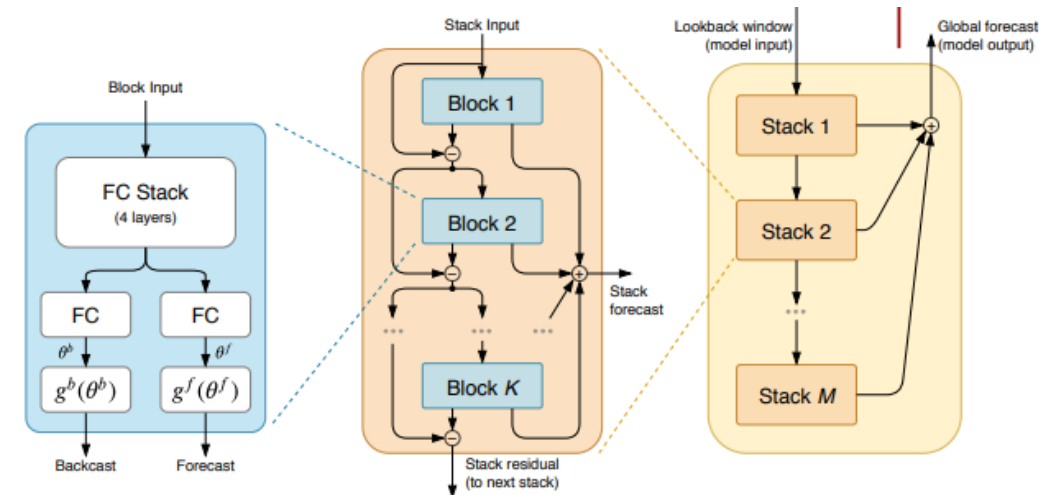
- Οι παράμετροι (weights) του RNN προσδιορίστηκαν μέσω της εκπαίδευσης στο σύνολο των χρονοσειρών
- Οι παράμετροι εξομάλυνσης προσδιορίστηκαν με βάση τα χαρακτηριστικά κάθε σειράς ξεχωριστά

Hybrid Models - Meta Learning

- Μια εξίσου ενδιαφέρουσα υβριδική προσέγγιση, η οποία αναπτύχθηκε από τον **Pablo Montero-Manso**, κατέλαβε την δεύτερη θέση στον διαγωνισμό προβλέψεων M4.
- Η μέθοδος του Montero-Manso βασίζεται στην **παραγωγή προβλέψεων για κάθε χρονοσειρά με κλασσικές στατιστικές μεθόδους** (Naïve, Theta, Arima, κλπ.).
- Η τελική πρόβλεψη για κάθε χρονοσειρά προκύπτει από **τον γραμμικό συνδυασμό των αντίστοιχων προβλέψεων** των απλών μεθόδων.
- Τα βάρη με τα οποία γίνεται ο βέλτιστος γραμμικός συνδυασμός υπολογίζονται από ένα **ML μοντέλο gradient boosting** (xgboost).
- Το ML μοντέλο (**meta-learner**) έχει εκπαιδευτεί ώστε να αναγνωρίζει τον καλύτερο συνδυασμό μεθόδων βάσει των χαρακτηριστικών (τάση, τυχαιότητα, εποχικότητα, αυτοσυσχέτιση, κλπ.) κάθε χρονοσειράς.
- Και σε αυτή την περίπτωση το τελικό μοντέλο έχει λάβει υπόψιν τις **ιδιαιτερότητες κάθε χρονοσειράς** (μέσω των κλασσικών μεθόδων προβλέψεων) αλλά και **μοτίβα μεταξύ διαφορετικών χρονοσειρών** (μέσω του meta-learner).

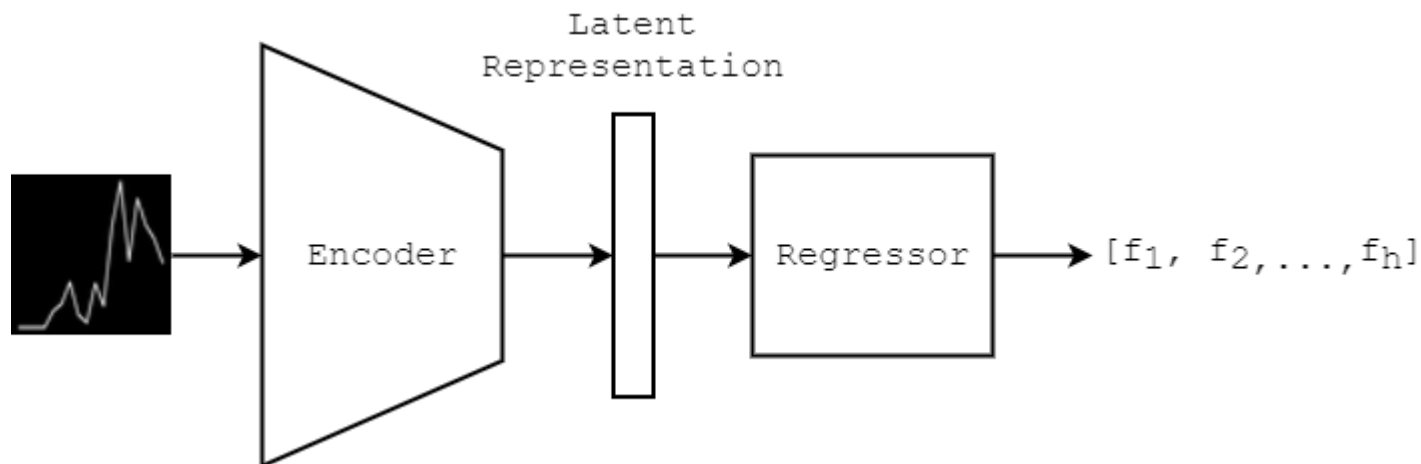
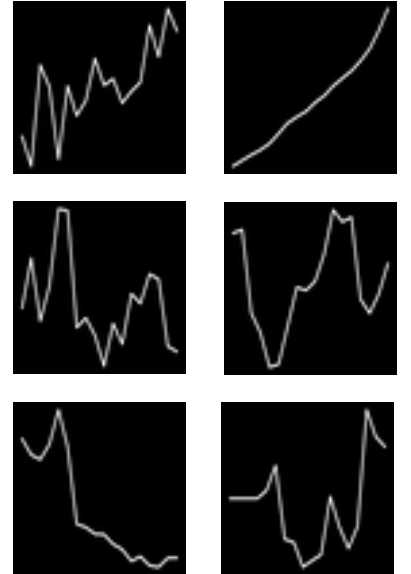
N-Beats - Deep Learning Method

- Το N-Beats αποτελεί μια **καθαρή ML μέθοδο** η οποία προτάθηκε από τους Oreshkin, Carpon, Charados & Bengio το 2020.
- Πρόκειται για μία από τις πρώτες επιτυχημένες προσπάθειες για εφαρμογή **Deep Learning** στον χώρο του forecasting.
- Η αρχιτεκτονική βασίζεται σε μια σειρά από μικρότερα νευρωνικά δίκτυα (**blocks**) τα οποία συνδέονται μεταξύ τους σειριακά και με χρήση παρακάμψεων (**shortcuts**).
- Σκοπός κάθε block είναι να μοντελοποιήσει όποια πληροφορία δεν κατάφερε να μοντελοποιήσει το προηγούμενο block μέσω των υπολειπόμενων σφαλμάτων (**residuals**).



Forecasting with Images

- Μια διαφορετική προσέγγιση που ερευνάται στο εργαστήριο μας είναι η **χρήση εικόνων των χρονοσειρών** για την παραγωγή προβλέψεων.
- Τα νευρωνικά δίκτυα έχουν αποδείξει τις δυνατότητες τους σε εφαρμογές επεξεργασίας εικόνας και στόχος είναι η **μεταφορά της τεχνογνωσίας** στο πεδίο των προβλέψεων.
- Αντί για την κλασική αριθμητική αναπαράσταση, οι χρονοσειρές δίνονται ως **είσοδος στο μοντέλο με την μορφή εικόνων**.
- Το μοντέλο χρησιμοποιεί **2D Convolutional layers** για την επεξεργασία τους.
- Η έξοδος του μοντέλου είναι οι ζητούμενες **αριθμητικές προβλέψεις**.



Forecasting with NN - Example

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf
```

```
import pandas as pd
import numpy as np
```

- ✓ **Tensorflow**: Πλατφόρμα που αναπτύχθηκε από την Google
- ✓ Ένα από τα πιο δημοφιλή back-ends για την ανάπτυξη NN

Από τις πιο **βασικές βιβλιοθήκες** της Python. Προσθέτουν νέες δομές δεδομένων (πίνακες, σειρές), έτοιμες συναρτήσεις, κλπ.

Forecasting with NN - Example

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf

import pandas as pd
import numpy as np

# Loop through the series
path = 'TrainSet.csv'
df_insample = pd.read_csv(path, sep=',', decimal='.')
```

Φόρτωση των δεδομένων
σε ένα Data Frame

Forecasting with NN - Example

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf

import pandas as pd
import numpy as np

# Loop through the series
path = 'TrainSet.csv'
df_insample = pd.read_csv(path, sep=',', decimal='.')

# Experiment parameters
input_size = 24
forecasting_horizon = 6
```

Καθορισμός των παραμέτρων

Το NN θα χρησιμοποιεί τις 24 πιο πρόσφατες παρατηρήσεις για να προβλεψει τις επόμενες 6

Forecasting with NN - Example

Global Training

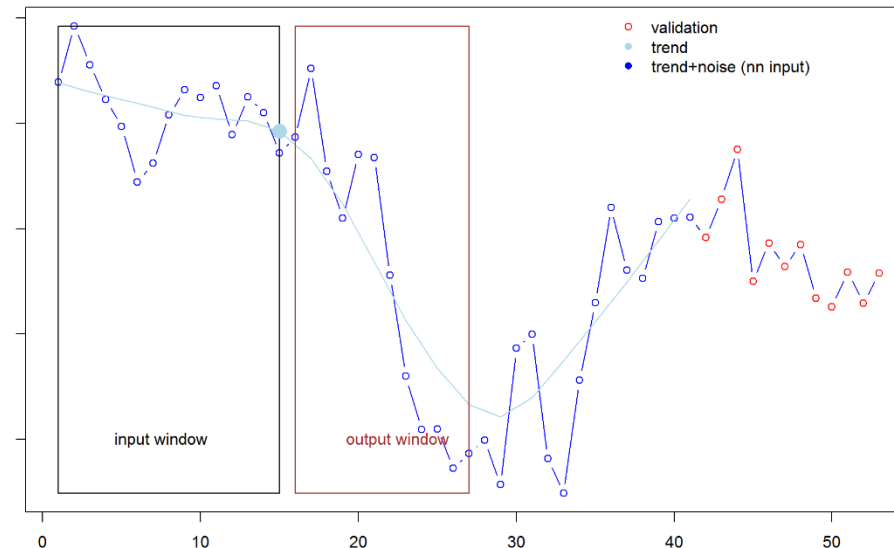
Αξιοποιούνται όλες οι διαθέσιμες
χρονοσειρές του dataset

```
df_train = list([])
df_test = list([])
for i in range(df_insampl.shape[0]):
    # Read series
    ts = np.asarray(df_insampl.iloc[i, :].dropna())

    # y_train
    y_train = ts[-forecasting_horizon:]
    # x_train
    x_train = ts[-(input_size + forecasting_horizon):-forecasting_horizon]
    # x_test
    x_test = ts[-input_size:]
```

Καθαρισμός των missing values

Εξαγωγή ενός παραθύρου ανά χρονοσειρά



Forecasting with NN - Example

```
df_train = list([])
df_test = list([])
for i in range(df_insample.shape[0]):
    # Read series
    ts = np.asarray(df_insample.iloc[i, :].dropna())

    # y_train
    y_train = ts[-forecasting_horizon:]
    # x_train
    x_train = ts[-(input_size + forecasting_horizon):-forecasting_horizon]
    # x_test
    x_test = ts[-input_size:]

    # Scale training data
    scaler = MinMaxScaler(feature_range=(0, 1))
    x_train = scaler.fit_transform(np.reshape(x_train, (-1, 1))).reshape(-1)
    y_train = scaler.transform(np.reshape(y_train, (-1, 1))).reshape(-1)

    # Scale test data
    scaler = MinMaxScaler(feature_range=(0, 1))
    x_test = scaler.fit_transform(np.reshape(x_test, (-1, 1))).reshape(-1)
```

- ✓ **Scaling** των δεδομένων εκπαίδευσης
- ✓ **Min-Max scaling** από 0 έως 1
- ✓ Επιτρέπουμε στο `y_train` να πάρει **τιμές λίγο έξω από τα όρια**.

Η διαδικασία επαναλαμβάνεται και για το `x_test`

Forecasting with NN - Example

```
# Save training sample
out_train_sample = list(x_train)
out_train_sample.extend(list(y_train))
df_train.append(out_train_sample)

# Save test sample
out_test_sample = list([scaler.data_min_[0], scaler.data_max_[0]])
out_test_sample.extend(list(x_test))
df_test.append(out_test_sample)

# Convert to Numpy arrays
df_train = np.asarray(df_train)
df_test = np.asarray(df_test)

# Split training and validation sets
x_train = df_train[:, :input_size]
y_train = df_train[:, -forecasting_horizon:]
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2)
```

Validation set

Ένα μέρος των διαθέσιμων δεδομένων δεν θα χρησιμοποιηθεί για την εκπαίδευση αλλά για την αξιολόγηση αυτής!

Forecasting with NN - Example

```
# Build simple MLP
inputs = tf.keras.layers.Input(shape=(input_size,))
x = tf.keras.layers.Dense(int(input_size * 1.5), activation='relu', kernel_initializer='he_uniform')(inputs)
x = tf.keras.layers.Dense(int(input_size * 1.5), activation='relu', kernel_initializer='he_uniform')(x)
x = tf.keras.layers.Dense(int(input_size * 1.5), activation='relu', kernel_initializer='he_uniform')(x)
lout = tf.keras.layers.Dense(forecasting_horizon, activation='linear', kernel_initializer='he_uniform')(x)
model = tf.keras.models.Model(inputs=inputs, outputs=lout)
```

Forecasting with NN - Example

```
# Build simple MLP
inputs = tf.keras.layers.Input(shape=(input_size,))
x = tf.keras.layers.Dense(int(input_size * 1.5), activation='relu', kernel_initializer='he_uniform')(inputs)
x = tf.keras.layers.Dense(int(input_size * 1.5), activation='relu', kernel_initializer='he_uniform')(x)
x = tf.keras.layers.Dense(int(input_size * 1.5), activation='relu', kernel_initializer='he_uniform')(x)
lout = tf.keras.layers.Dense(forecasting_horizon, activation='linear', kernel_initializer='he_uniform')(x)
model = tf.keras.models.Model(inputs=inputs, outputs=lout)

# Compile the model
opt = tf.keras.optimizers.Adam(lr=0.001)
model.compile(optimizer=opt, loss='mae')
```

Forecasting with NN - Example

```
# Fit the model
es = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=2, patience=10,
                                     min_delta=0.0001, restore_best_weights=True)
model.fit(x_train, y_train, batch_size=64, epochs=100, validation_data=(x_val, y_val), callbacks=[es])

# Save the trained model
model.save('TrainedModel.h5')

# Get predictions
predictions = model.predict(x_test[:, 2:])

# Scale back the predictions
d_min = df_test[:, 0]
d_max = df_test[:, 1]
predictions = (predictions * (d_max - d_min)) + d_min
```

Forecasting with NN - Example

```
# Fit the model
es = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=2, patience=10,
                                     min_delta=0.0001, restore_best_weights=True)
model.fit(x_train, y_train, batch_size=64, epochs=100, validation_data=(x_val, y_val), callbacks=[es])

# Save the trained model
model.save('TrainedModel.h5')
```