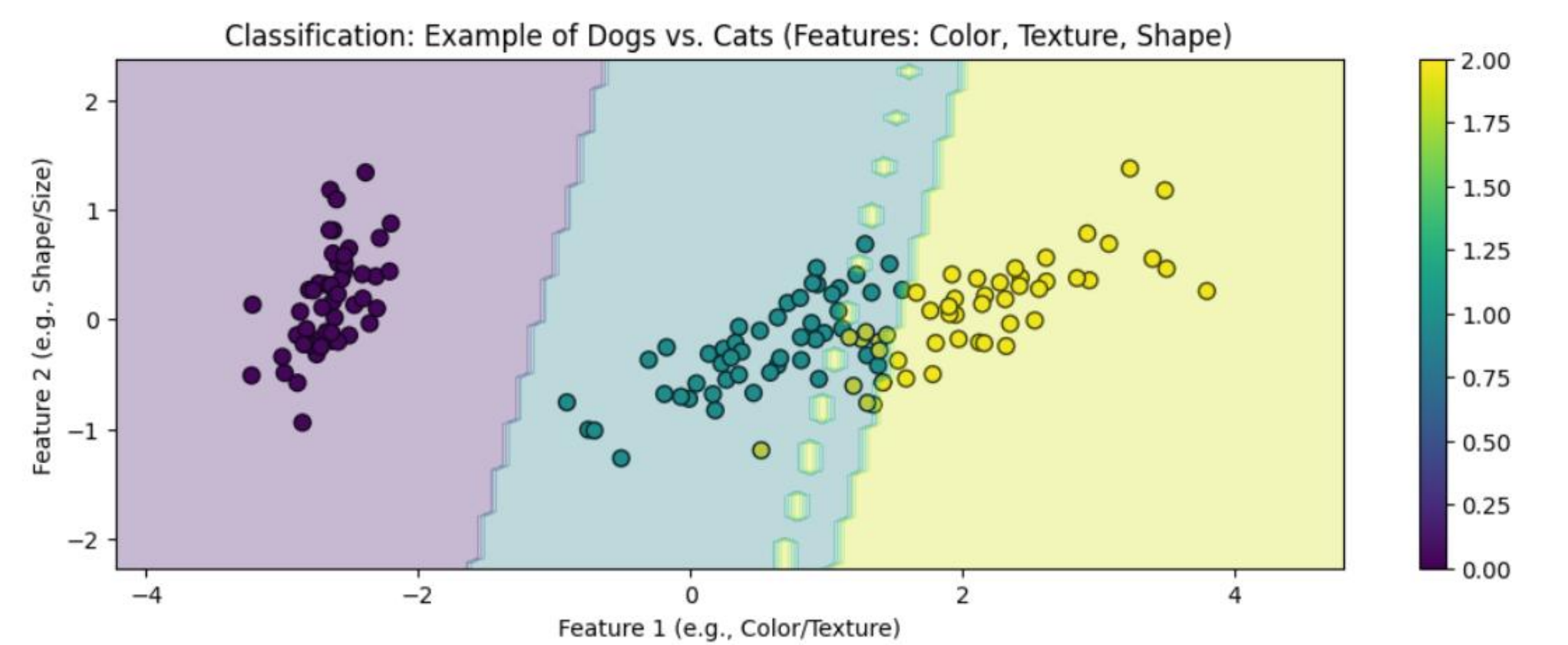
# CLASSIFICATION

Μάνια Ζωγράφου

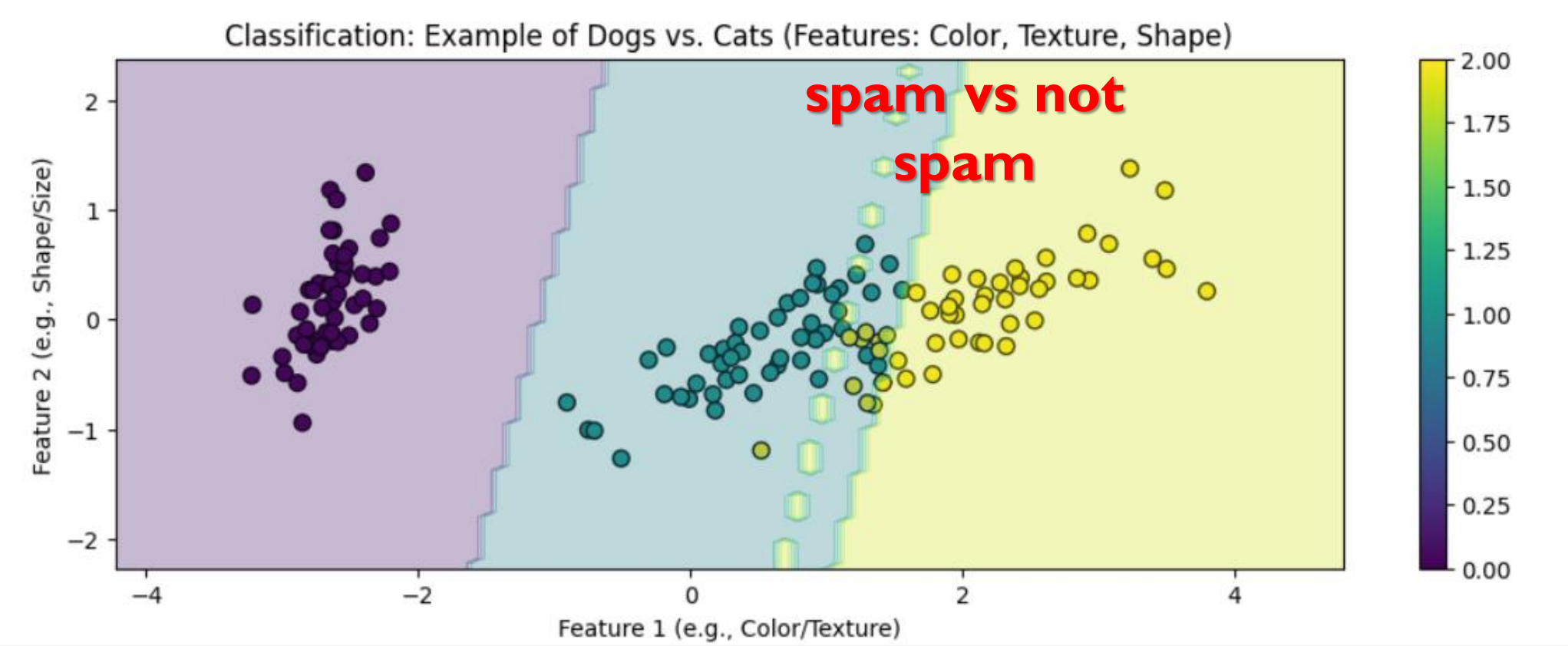
# INTRO TO CLASSIFICATION

# CLASSIFICATION




https://www.geeksforgeeks.org/machine-learning/getting-started-with-classification/

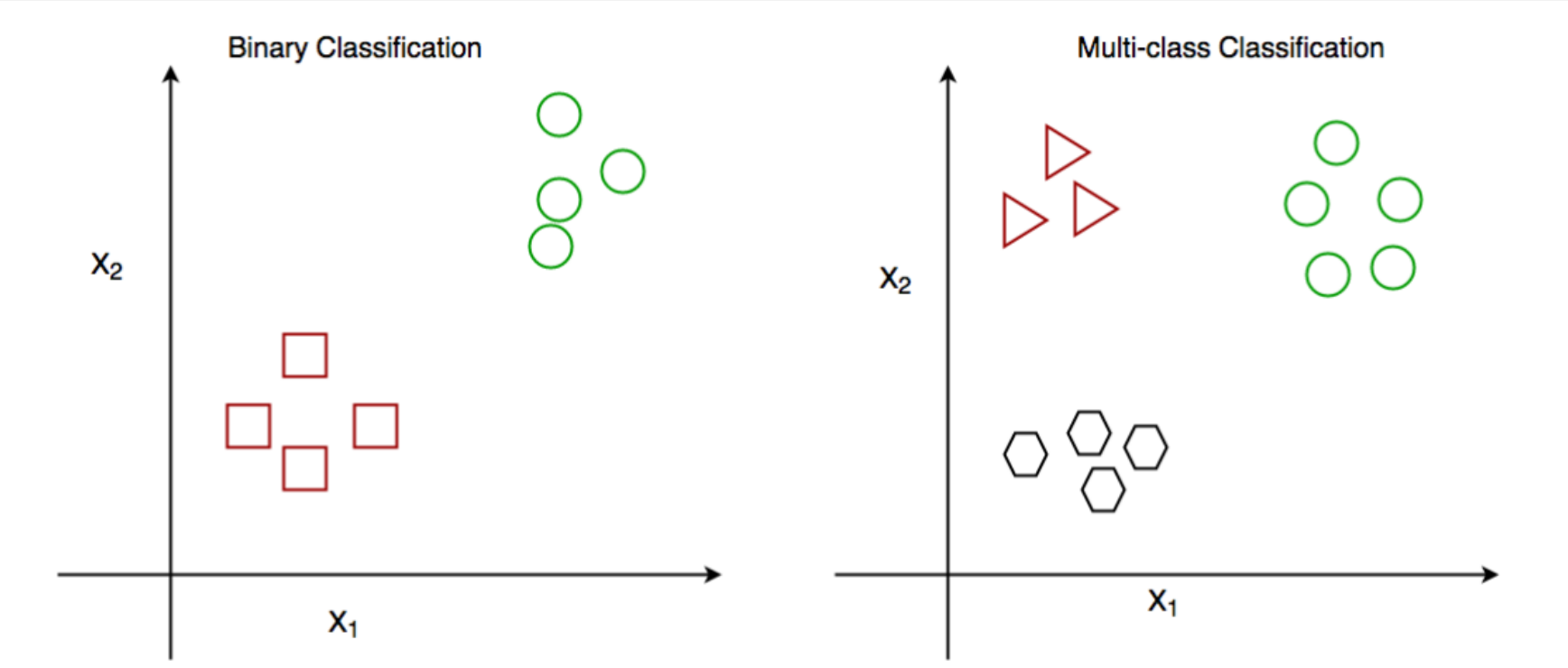
# CLASSIFICATION



# TYPES

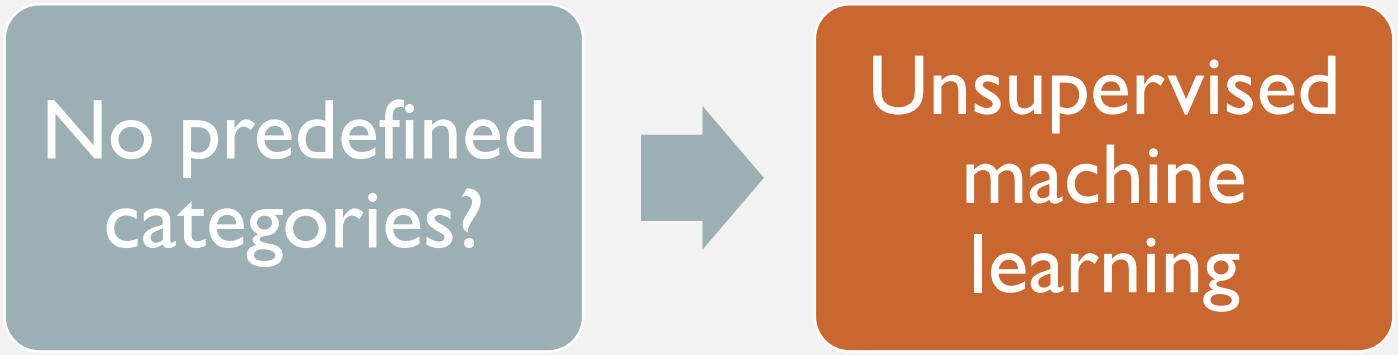
- **Binary Classification**
  - *0 or 1*
- **Multiclass Classification**
  - ***cat***, ***dog***, *and* ***bird***
- **Multi-Label Classification**
  - *movie* ⇛ ***action*** *and* ***comedy***

# TYPES

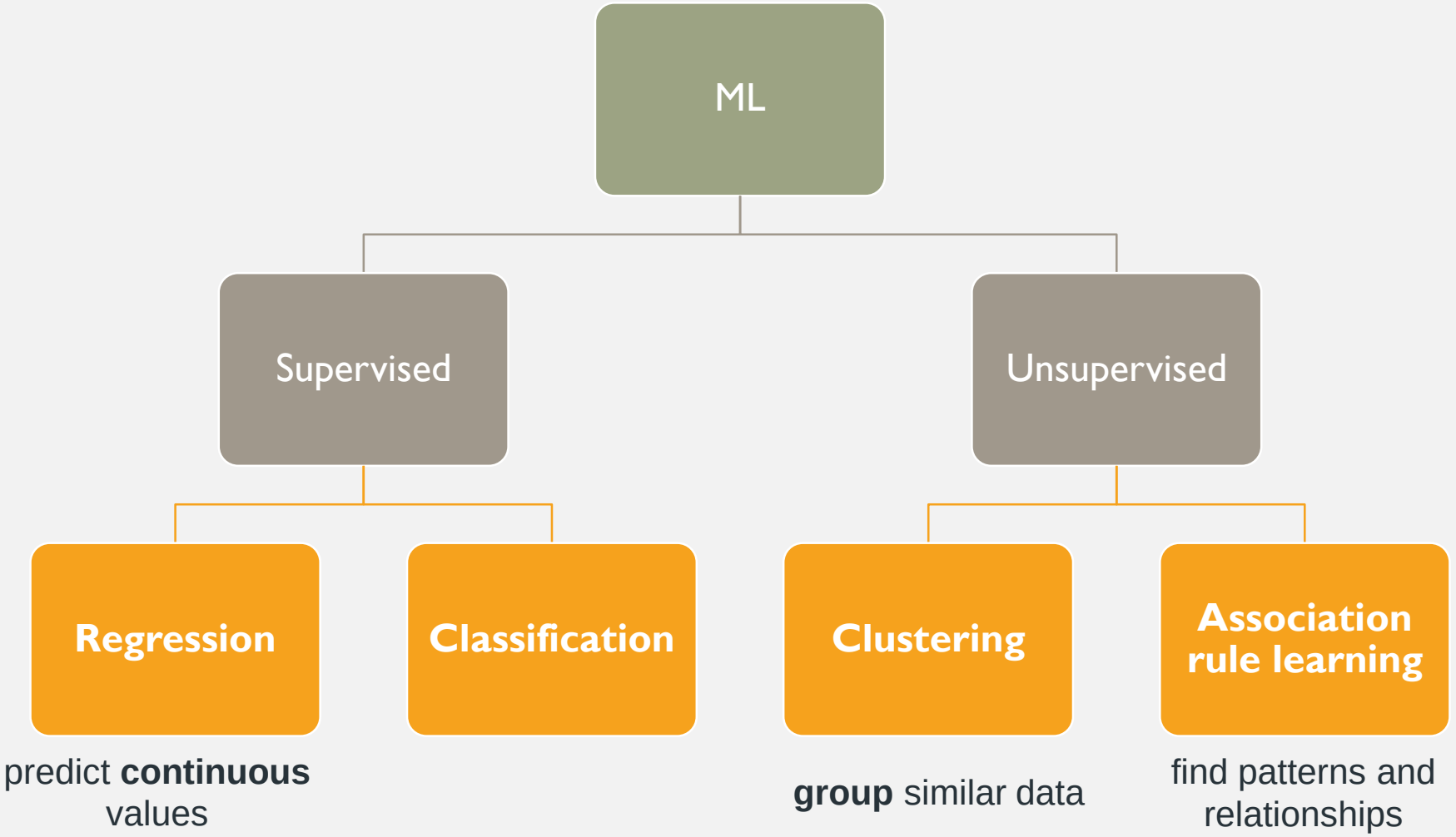


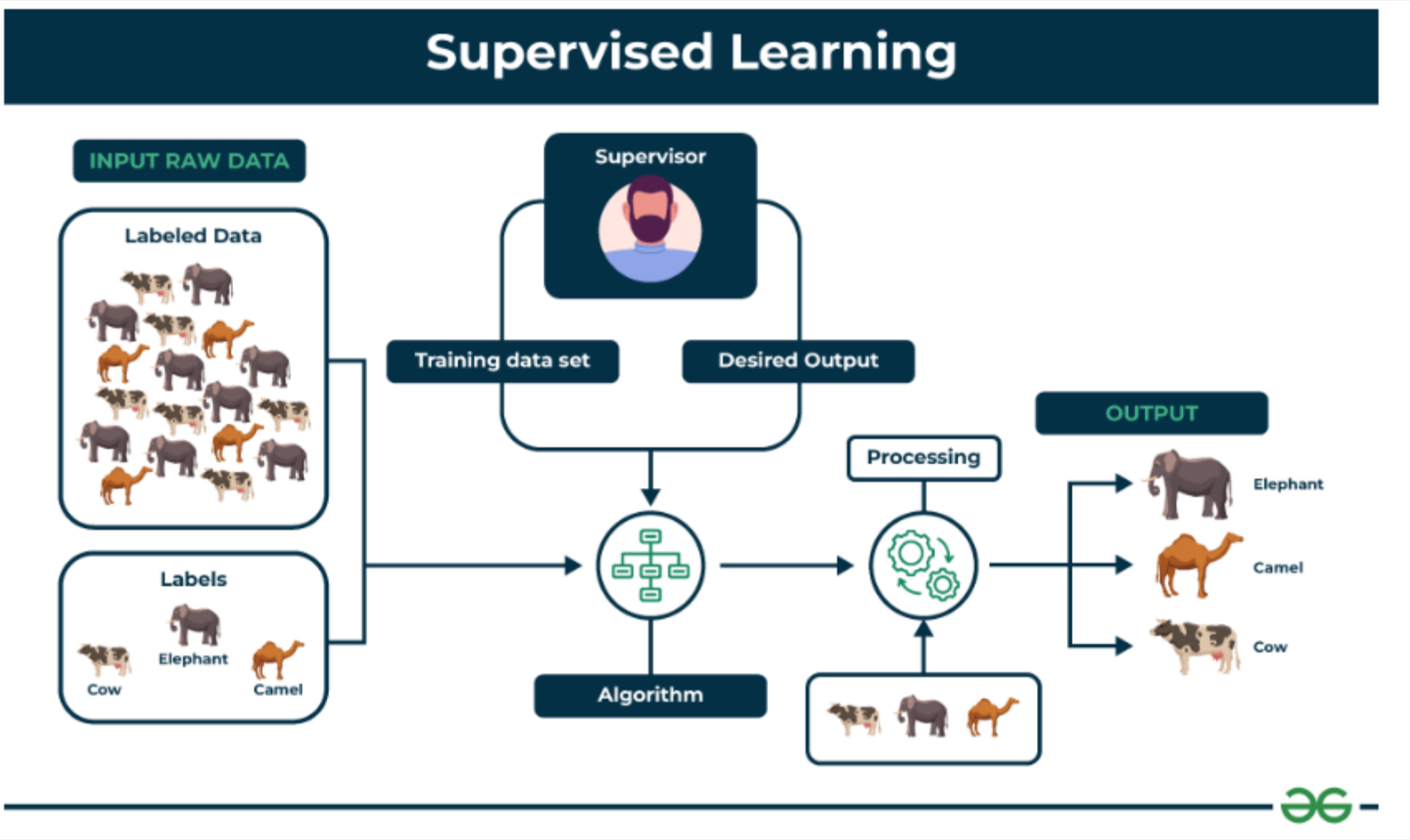

https://www.geeksforgeeks.org/machine-learning/getting-started-with-classification/

# TYPES

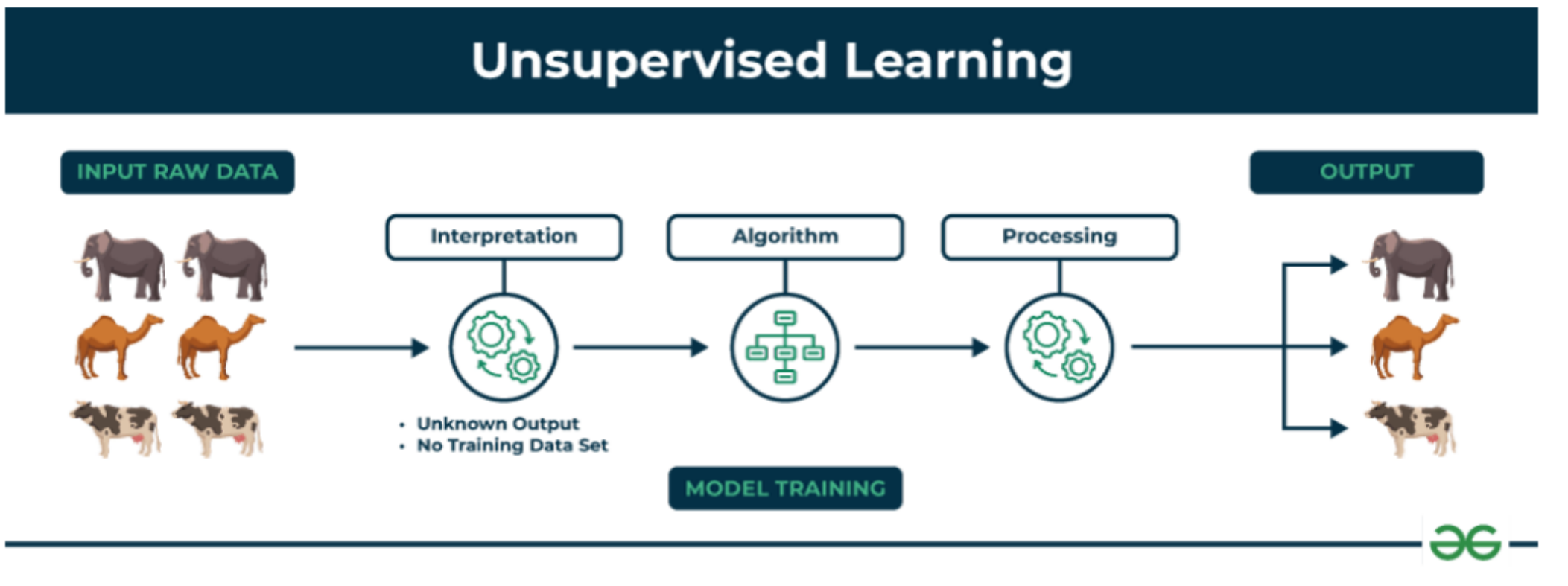


# UNSUPERVISED VS SUPERVISED


ML
Supervised
Unsupervised
Regression
Classification
Clustering
Association rule learning
predict **continuous** values
**group** similar data
find patterns and relationships




https://www.geeksforgeeks.org/machine-learning/supervised-unsupervised-learning/

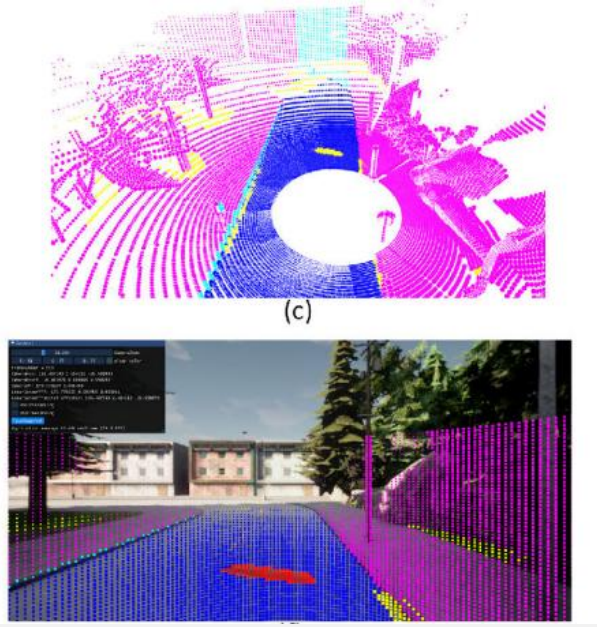



https://www.geeksforgeeks.org/machine-learning/supervised-unsupervised-learning/

# APPLICATIONS

## UNSUPERVISED

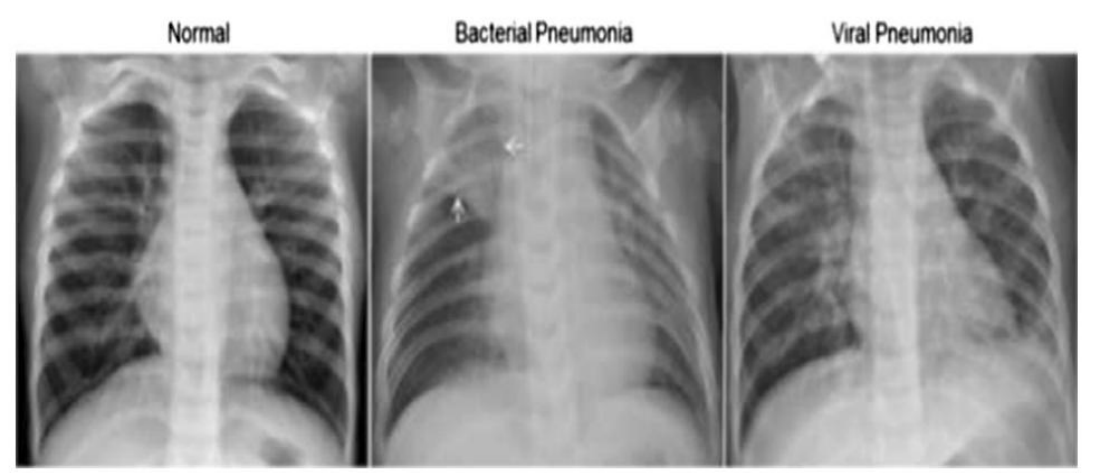
- **Anomaly detection (e.g. pca)**
- **Scientific discovery**
- **Recommendation systems**




https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10311086

## SUPERVISED

- **Image classification**
- **Medical diagnosis**
- **NLP/Fraud detection**




https://link.springer.com/article/10.1186/s40537-019-0276-2

# CLASSIFICATION

# LINEAR CLASSIFIERS

- Logistic Regression
- Support Vector Machines having kernel = 'linear'
- Single-layer Perceptron
- Stochastic Gradient Descent (SGD) Classifier

# NON-LINEAR CLASSIFIERS

- K-Nearest Neighbours
- Kernel SVM
- Naive Bayes
- Decision Tree Classification
- Ensemble learning classifiers: (XGBoost)
- Random Forests,
- AdaBoost,
- Bagging Classifier,
- Voting Classifier,
- Extra Trees Classifier
- Multi-layer Artificial Neural Networks

https://www.geeksforgeeks.org/machine-learning/getting-started-with-classification/

# STEPS

**1.Data Collection**
- Labelled data

**2.Feature Extraction**
- system identifies features (like color, shape, or texture)

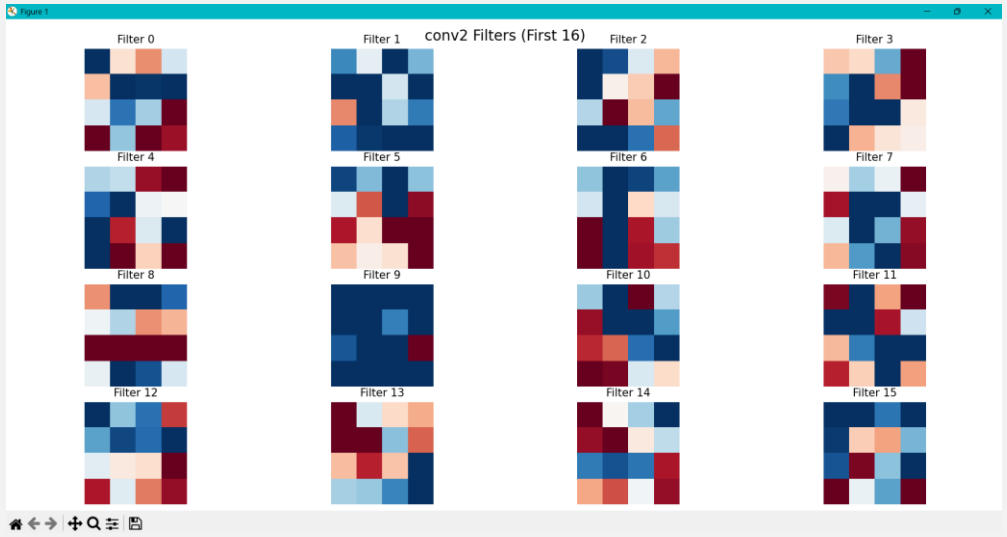
**3.Model Training**

**4.Model Evaluation**

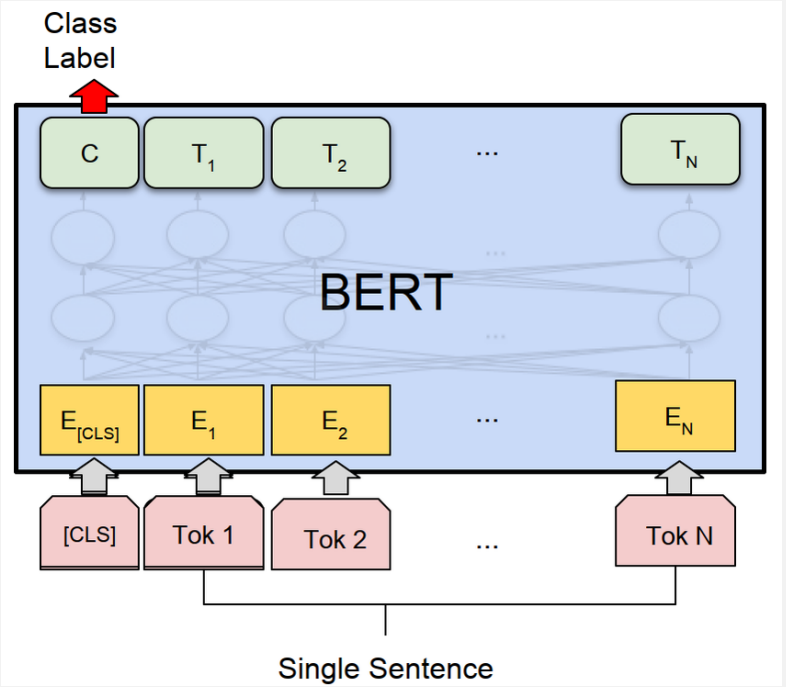
**5.Prediction**

**6.Optional: tune hyperparameters**

# FEATURE EXTRACTION




Filters (Kernels)




dense vector representations

# AGENDA

**Text classification**

- Σύγκριση αλγορίθμων

**Titanic**

- Πρόβλεψη επιβίωσης https://www.kaggle.com/code/alexisbcook/titanic-tutorial

# ALGORITHMS

- ❑ Naive Bayes
- ❑ Logistic Regression
- ❑ Random Forest
- ❑ k-NN
- ❑ Gzip+k-NN
- ❑ SVM(linear,…)
- ❑ **XGBoost**

# TEXT CLASSIFICATION

- Simple classification (TF-IDF) Statistical / frequency-based methods
  - Transform the text into numbers (frequency)

Vs

- BERT and word embeddings (Contextual / semantic methods )
- Hugging face embeddings, word in text => vector depending on meaning

Maybe next meeting? NLP, RAG, BERT, anonymization…

# VECTORIZATION

TF-IDF

# PARSING - LABEL ENCODER:

- **επεξεργασία των δεδομένων εκπαίδευσης: κείμενο και κλάσεις.**
- Οι κλάσεις μετατρέπονται σε ακεραίους και δημιουργείται ο **label encoder** που αντιστοιχίζει τους ακέραιους σε ονόματα κλάσεων και αντίστροφά.
- Ύστερα τα δεδομένα χωρίζονται σε δεδομένα εκπαίδευσης (80%) και δεδομένα testing (20%)

# VECTORIZATION TF-IDF MATRIX

- *Input data:* Λίστα από έγγραφα του training dataset, σε μορφή strings
- Δημιουργείται ένα TfidfVectorizer **(vectorizer object)** το οποίο αναλύει το κείμενο σε λέξεις (analyzer="word") και εξάγει
  - **unigrams** (1 λέξη)
  - και **bigrams** (2 συνεχόμενες λέξεις/δίγραμμα) (ngram_range=(1, 2)).
- Max features επιλογή k=10.000 **σημαντικότερες** λέξεις/διγράμματα (βάσει συχνότητας σε όλα τα έγγραφα και IDF)

***French Revolution, French, Revolution => Διαφορετικοί όροι!***

**TF-IDF (term) = TF (term) × IDF (term)**

**TF (term)** = ( Αριθμός εμφάνισης της λέξης στο **έγγραφο /**

Συνολικός αριθμός λέξεων στο **έγγραφο** )

**IDF (term)** = log ( Συνολικός αριθμός εγγράφων **/**

Αριθμός εγγράφων που περιέχουν τη λέξη )

| | Feature_1 | Feature_2 | … | …. | … | Feature_10_000 |
|---|---|---|---|---|---|---|
| Text_1 | TF-IDF_11 | TF-IDF_21 | … | … | …. | TF-IDF_10k1 |
| Text_2 | TF-IDF_12 | TF-IDF_22 | … | … | … | TF-IDF _10k2 |
| … | … | … | … | … | … | … |
| Text_N | TF-IDF_1N | TF-IDF_2N | … | … | … | TF-IDF _10kN |

# TF-IDF

- **Αποτέλεσμα:** Το TF-IDF είναι υψηλό για λέξεις που εμφανίζονται **συχνά σε ένα συγκεκριμένο έγγραφο** (υψηλό TF) **και** είναι **σπάνιες στο corpus** (υψηλό IDF). Άρα αυτές οι λέξεις έχουν μεγαλύτερη σημασία στην ταξινόμηση του εγγράφου.
- Εν συντομία:
- Οι 10 χιλιάδες συχνότερες λέξεις ή διγράμματα αποθηκεύονται στον πίνακα.
- Κάθε στήλη αντιπροσωπεύει μία λέξη/δίγραμμα.
- Κάθε σειρά αντιστοιχεί σε ένα έγγραφο.
- Κάθε κελί περιέχει την TF-IDF τιμή της λέξης/διγράμματος της στήλης που προκύπτει από το έγγραφο της σειράς.
- Οι 10_000 στήλες είναι τα **features** που χρησιμοποιούνται στην εφαρμογή ταξινόμησης κειμένου.

# TF-IDF VS BERT

- TF-IDF => SIMPLE CASES IS ENOUGH
- COMPLEX TEXTS => REQUIRE BERT

# ALGORITHMS FOR NEWS CLASSIFICATION

# SUGGESTED AGENDA

- Naïve bayes theory
- Results
- Xgboost theory
- Results

- Accuracy measures : “f1–score”
- Confusion matrix:“quality of learning”

# NAIVE BAYES CLASSIFIERS

- https://www.geeksforgeeks.org/machine-learning/naive-bayes-classifiers/
- ✨ python
- X_train_vec and X_test_vec = TF-IDF
- y_train = training labels.

```
from sklearn.naive_bayes import MultinomialNB

clf = MultinomialNB()  # Instantiate the classifier
clf.fit(X_train_vec, y_train)  # Train on the training data
y_pred = clf.predict(X_test_vec)  # Predict on the test data
```

# NAIVE BAYES CLASSIFIERS

## Bayes Theorem Formula

For any two events A and B, **Bayes's** formula for the Bayes theorem is given by:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

# NAIVE ASSUMPTION

- assumption that all features are independent given the class
- (θεώρημα ανεξαρτησίας)

$$P(x_1, x_2, ..., x_n|y) = P(x_1|y) \cdot P(x_2|y) \cdots P(x_n|y)$$

- Πώς επιλέγουμε; ΚΑΝΟΝΑΣ ΑΠΟΦΑΣΗΣ ΚΑΤΆ BAYES
  - επιλέγεται η κλάση που μεγιστοποιεί την εκ των υστέρων πιθανότητα

$$\hat{C} = \arg\max_{k=1}^{K} P(C_k|x)$$

# DECISION RULE

$$P(C_k|x) = \frac{P(x|C_k) \cdot P(C_k)}{P(x)}$$

•P(C_k): Η **Εκ των Προτέρων Πιθανότητα** (Prior) της κλάσης C_k

•P(x | C_k): Η **Πιθανοφάνεια** (Likelihood), η πιθανότητα να παρατηρηθεί το x αν ανήκει στην κλάση C_k.

•P(x): Η **Πιθανότητα Μάρτυρας** (Evidence), η πιθανότητα να παρατηρηθεί το x. (Αυτό συνήθως παραλείπεται στον κανόνα απόφασης, καθώς είναι το ίδιο για όλες τις κλάσεις).

$$\hat{C} = \arg\max_{k=1}^{K} P(C_k|x)$$

# DECISION RULE

## ΔΥΟ ΚΛΆΣΕΙΣ (K=2)

$$P(C_1|d) > P(C_2|d)$$

$$\log\left(\frac{P(C_1|d)}{P(C_2|d)}\right) > 0$$

## ΠΟΛΛΈΣ ΚΛΆΣΕΙΣ (K>2)

- Δεν αρκεί ο λόγος. Πρέπει να **υπολογίσουμε την πιθανότητα για κάθε κλάση** ξεχωριστά και στη συνέχεια να συγκρίνουμε τις τιμές για να βρούμε τη μέγιστη.
- $P(C_1|x), P(C_2|x), \ldots, P(C_K|x)$

# ΓΙΑΤΙ ΧΡΕΙΑΖΕΤΑΙ ΝΕΥΡΩΝΙΚΟ;

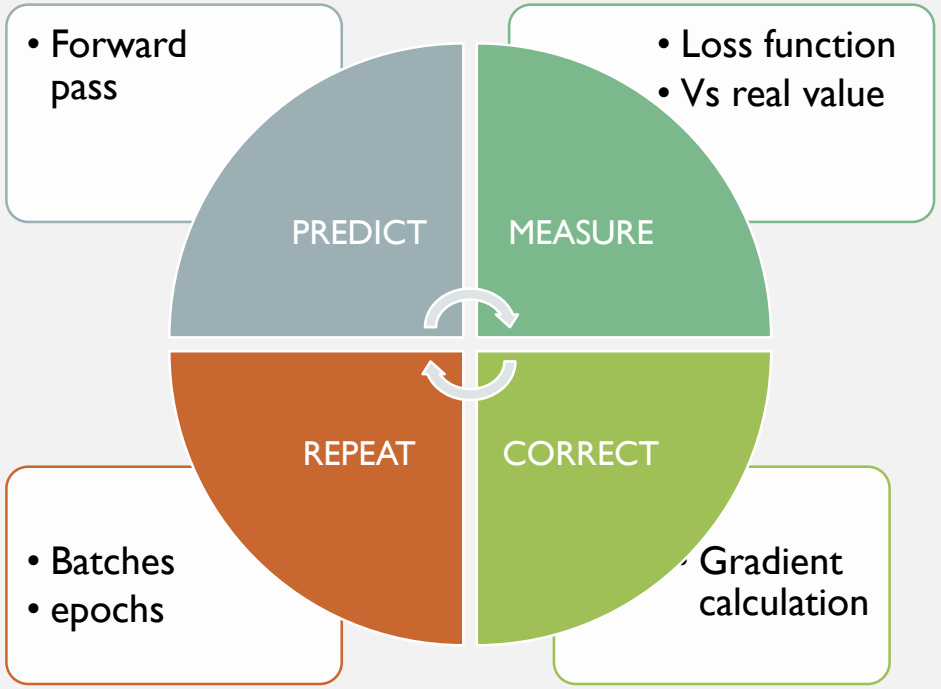
- δυσκολία υπολογισμού της Πιθανοφάνειας P(x | C_k) για σύνθετα δεδομένα
- Παράδειγμα: Αναγνώριση Εικόνας
  - C1: Γατα
  - C2: όχι γατα (σκυλος)
- Είσοδος εικόνα, μεγέθους 10*10
- Πιθανοφάνεια: $P(x|\text{Cat}) = P(x_1, x_2, \ldots, x_{100} \mid \text{Cat})$

**Γιατί είναι αδύνατο;**

- Κάθε pixel μπορεί να λάβει 256 τιμές.
- Ο συνολικός αριθμός πιθανών εικόνων σε κλίμακα του γκρι είναι 256^100.
- Για να υπολογίσουμε σωστά την P(x | Cat), θα έπρεπε να δούμε πώς κατανέμεται η πιθανότητα σε αυτόν τον τεράστιο χώρο για όλες τις εικόνες "Γάτας" στον κόσμο.

# NEURAL NETWORK

- Το νευρωνικό δίκτυο είναι μια παραμετρική συνάρτηση **υ(x,θ)** που με βελτιστοποίηση προσπαθεί να προσεγγίσει τον Bayes. (Bayes Optimal Classifier)



# 20 NEWS GROUP

- The 20 newsgroups dataset comprises around
  - 18000 newsgroups posts
  - on 20 topics
- split in two subsets:
  - one for training (or development)
  - one for testing (or for performance evaluation).
  - (The split between the train and test set is based upon a messages posted before and after a specific date.)

```
from sklearn.datasets import fetch_20newsgroups
```

# NAÏVE BAYES ON 20NEWSGROUP

AI_SG\1\bayes_train_20newsgroups.py:

| | | | | |
|---|---|---|---|---|
| accuracy | | | **0.67** | 7532 |
| macro avg | 0.70 | 0.66 | 0.65 | 7532 |
| weighted avg | 0.70 | 0.67 | 0.66 | 7532 |

# F1-SCORE

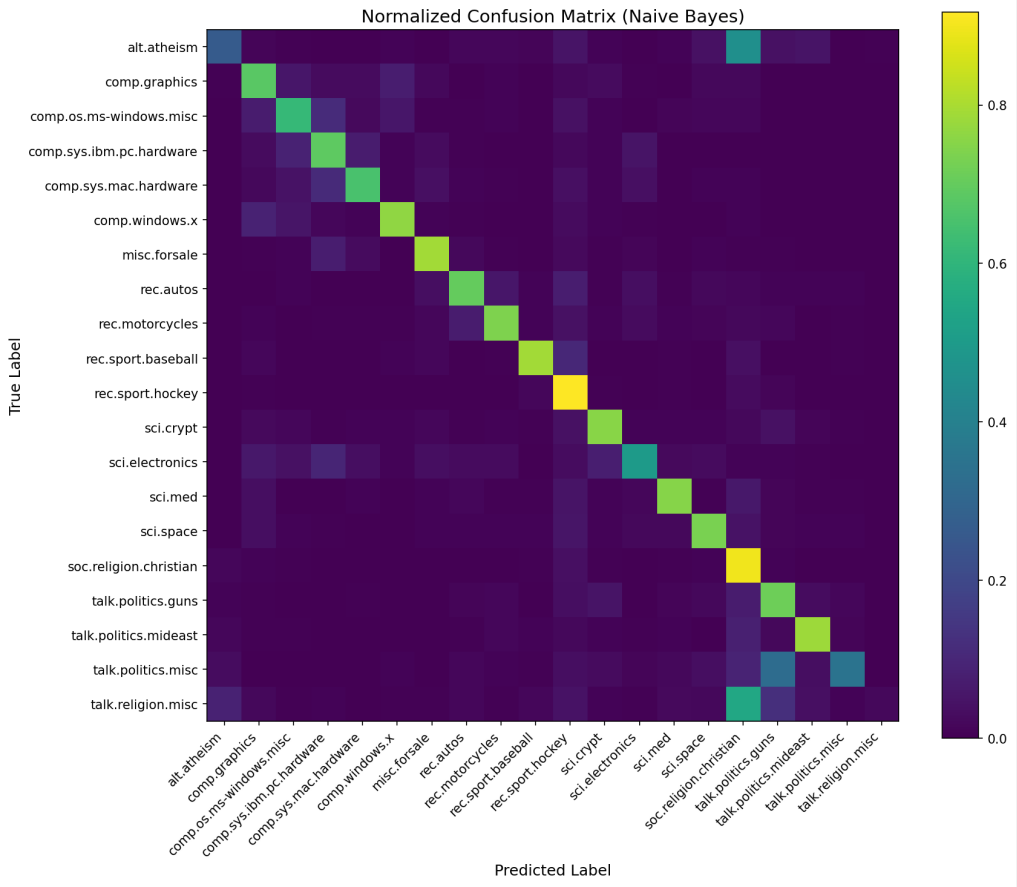
- **F1-score:**
- Το F1-score χρησιμοποιείται για την αξιολόγηση μοντέλων ταξινόμησης και λαμβάνει υπόψιν του τα false positives και τα false negatives. Το F1-score είναι ο συνδυασμός precision και recall, και ο σταθμισμένος - "weighted" μέσος όρος λαμβάνει υπόψη την υποστήριξη (support) κάθε κλάσης.
- **F1_Score = 2 * (Precision * Recall) / (Precision + Recall)**, όπου:
- **Precision = True Positives / True Positives + False Positives**
- **Recall = True Positives / True Positives + False Negatives**
- Καθώς επιλέχθηκε σταθμισμένος μέσος όρος Το F1-score κάθε κλάσης πολλαπλασιάζεται με την υποστήριξη (support) της κλάσης, δηλαδή τον αριθμό των δειγμάτων που ανήκουν σε αυτήν την κλάση. Ο σταθμισμένος μέσος όρος υπολογίζεται ως:
- **Support:** αριθμός των δειγμάτων που ανήκουν σε αυτήν την κλάση
- **Weighted_F1 = (F1_1 * Support_1 + F1_2 * Support_2 + ... + F1_n * Support_n) / Total_Support**

# EVALUATION

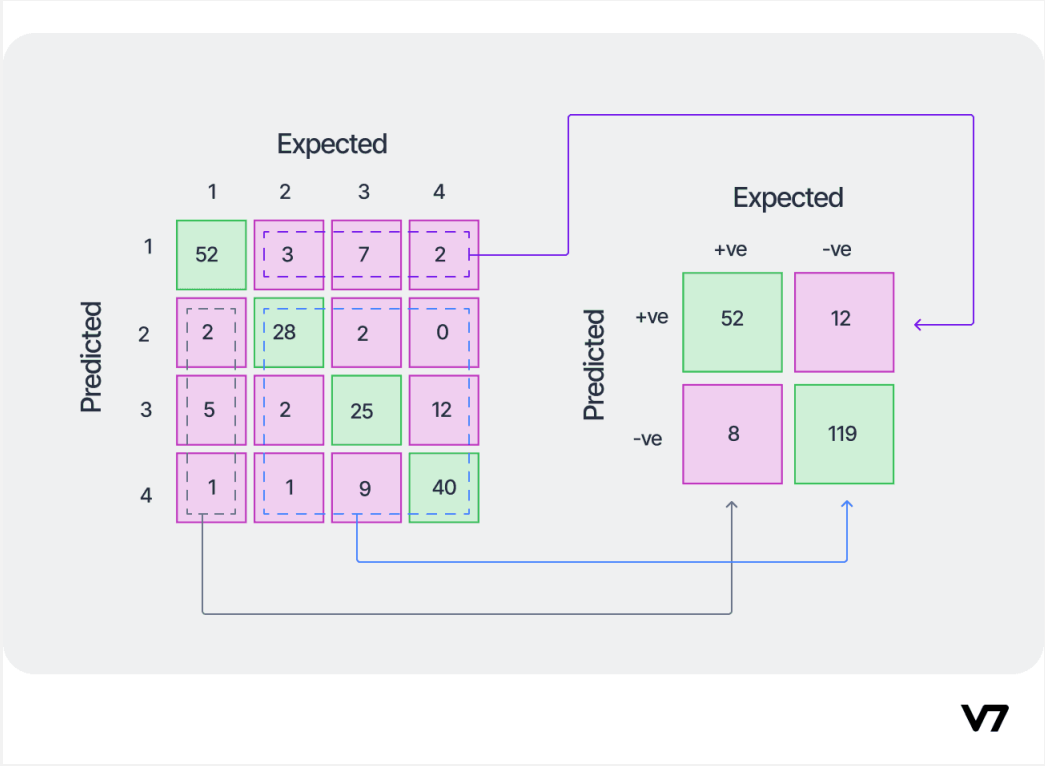
Classification report (Naive Bayes):

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| alt.atheism | 0.61 | 0.26 | 0.37 | 319 |
| comp.graphics | 0.61 | 0.68 | 0.65 | 389 |
| comp.os.ms-windows.misc | 0.64 | 0.61 | 0.63 | 394 |
| comp.sys.ibm.pc.hardware | 0.60 | 0.69 | 0.64 | 392 |
| comp.sys.mac.hardware | 0.74 | 0.65 | 0.70 | 385 |
| comp.windows.x | 0.79 | 0.76 | 0.78 | 395 |
| misc.forsale | 0.76 | 0.79 | 0.77 | 390 |
| rec.autos | 0.75 | 0.70 | 0.73 | 396 |
| rec.motorcycles | 0.79 | 0.74 | 0.77 | 398 |
| rec.sport.baseball | 0.88 | 0.79 | 0.84 | 397 |
| rec.sport.hockey | 0.56 | 0.92 | 0.70 | 399 |
| sci.crypt | 0.74 | 0.75 | 0.74 | 396 |
| sci.electronics | 0.67 | 0.50 | 0.57 | 393 |
| sci.med | 0.83 | 0.75 | 0.79 | 396 |
| sci.space | 0.76 | 0.73 | 0.75 | 394 |
| soc.religion.christian | 0.42 | 0.90 | 0.58 | 398 |
| talk.politics.guns | 0.56 | 0.71 | 0.63 | 364 |
| talk.politics.mideast | 0.82 | 0.78 | 0.80 | 376 |
| talk.politics.misc | 0.79 | 0.35 | 0.48 | 310 |
| talk.religion.misc | 0.62 | 0.02 | 0.04 | 251 |
| accuracy | | | 0.67 | 7532 |
| macro avg | 0.70 | 0.66 | 0.65 | 7532 |
| weighted avg | 0.70 | 0.67 | 0.66 | 7532 |

# CONFUSION MATRIX

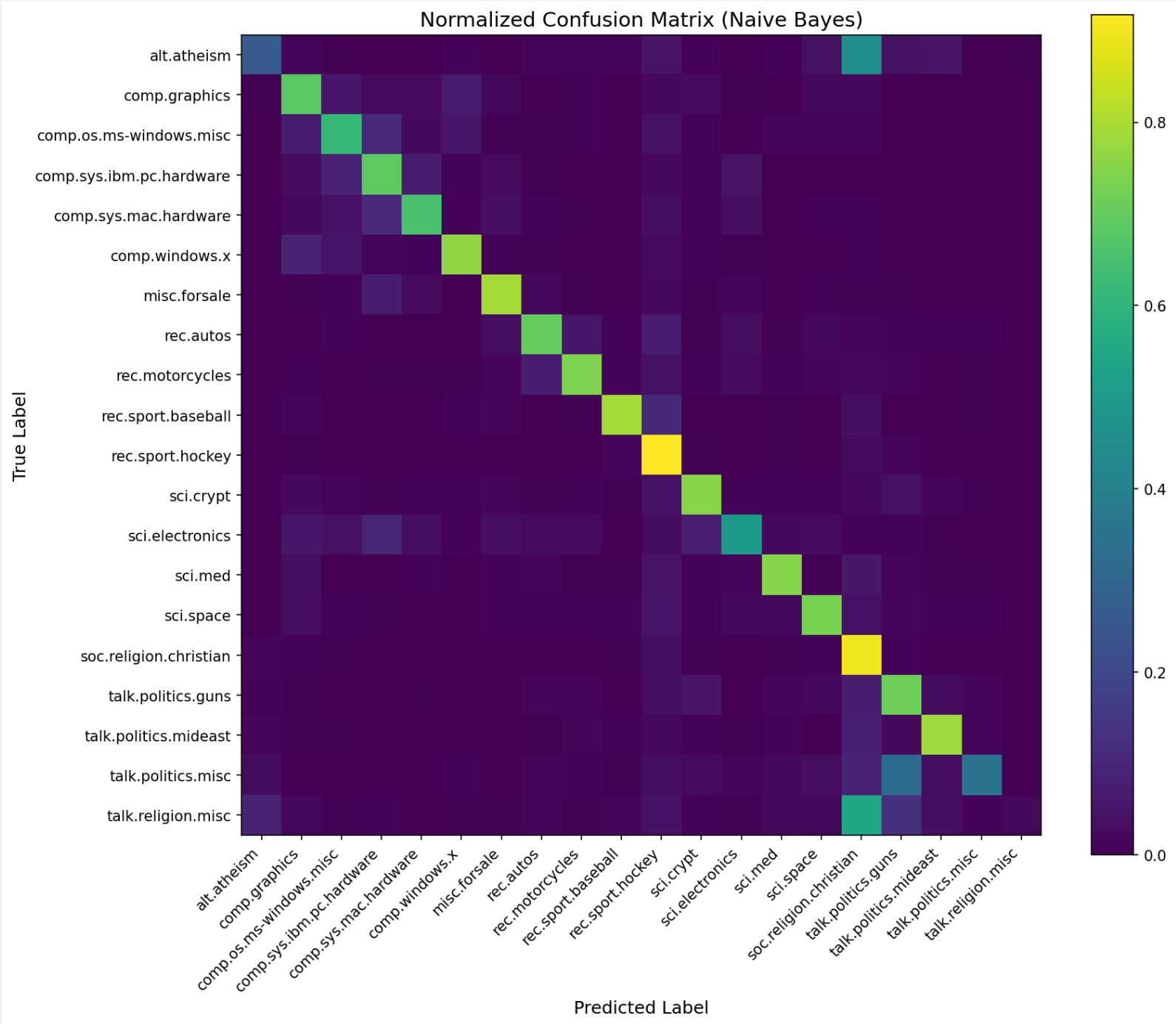


# CONFUSION MATRIX




## Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | True Positives (TPs) | False Positives (FPs) |
| Predicted Negative (0) | False Negatives (FNs) | True Negatives (TNs) |


Normalized Confusion Matrix (Naive Bayes)
True Label
Predicted Label
alt.atheism
comp.graphics
comp.os.ms-windows.misc
comp.sys.ibm.pc.hardware
comp.sys.mac.hardware
comp.windows.x
misc.forsale
rec.autos
rec.motorcycles
rec.sport.baseball
rec.sport.hockey
sci.crypt
sci.electronics
sci.med
sci.space
soc.religion.christian
talk.politics.guns
talk.politics.mideast
talk.politics.misc
talk.religion.misc
0.0
0.2
0.4
0.6
0.8

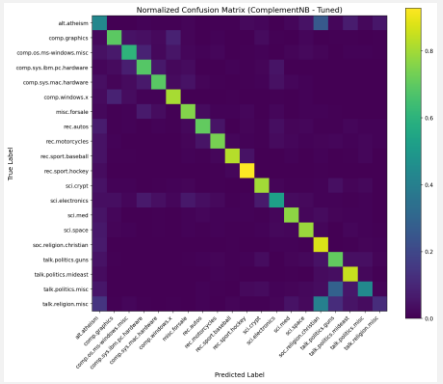
# ΒΕΛΤΙΩΣΗ ΑΛΓΟΡΙΘΜΟΥ

- **Δοκιμή παραλλαγών** (complementNB)
- The Complement Naive Bayes classifier was designed to correct the "severe assumptions" made by the standard Multinomial Naive Bayes classifier. It is particularly suited for imbalanced data sets.

| | | | | |
|---|---|---|---|---|
| accuracy | | | 0.70 | 7532 |
| macro avg | 0.70 | 0.69 | 0.68 | 7532 |
| weighted avg | 0.71 | 0.70 | 0.70 | 7532 |

# ΒΕΛΤΙΩΣΗ ΑΛΓΟΡΙΘΜΟΥ

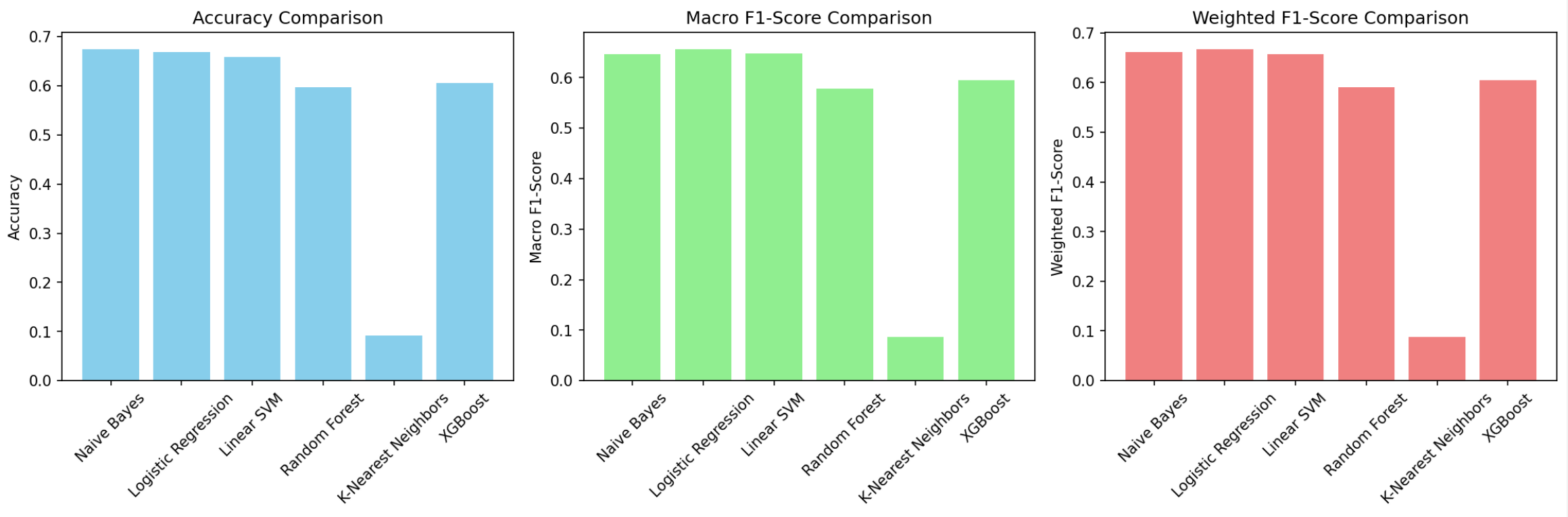
- Hyperparameter tuning
- Gird search
- Best parameters: {'classifier__alpha': 0.1, 'vectorizer__max_df': 0.8, 'vectorizer__max_features': 30000, 'vectorizer__min_df': 2, 'vectorizer__ngram_range': (1, 1)}
- Best cross-validation score: 0.7471
- Test set accuracy: 0.7007 (όχι και πολύ καλυτερα)



# ΒΕΛΤΙΩΣΗ ΣΥΣΤΗΜΑΤΟΣ

- Δοκιμή άλλων αλγορίθμων

# EXAMPLE RESULTS



# ΠΡΟΕΤΟΙΜΑΣΙΑ ΓΙΑ ΤΙΤΑΝΙΚΟ

- Naïve Bayes (fast)
- Logistic Regression (baseline for binary classification)
- More complex:
  - Random forest
  - XGBoost

- Idea: Να δειξω τωρα XGBoost
- Και να ξεκινησουμε τον τιτανικο με LR για baseline συγκριση μετα πιο περιπλοκων

# XGBOOST

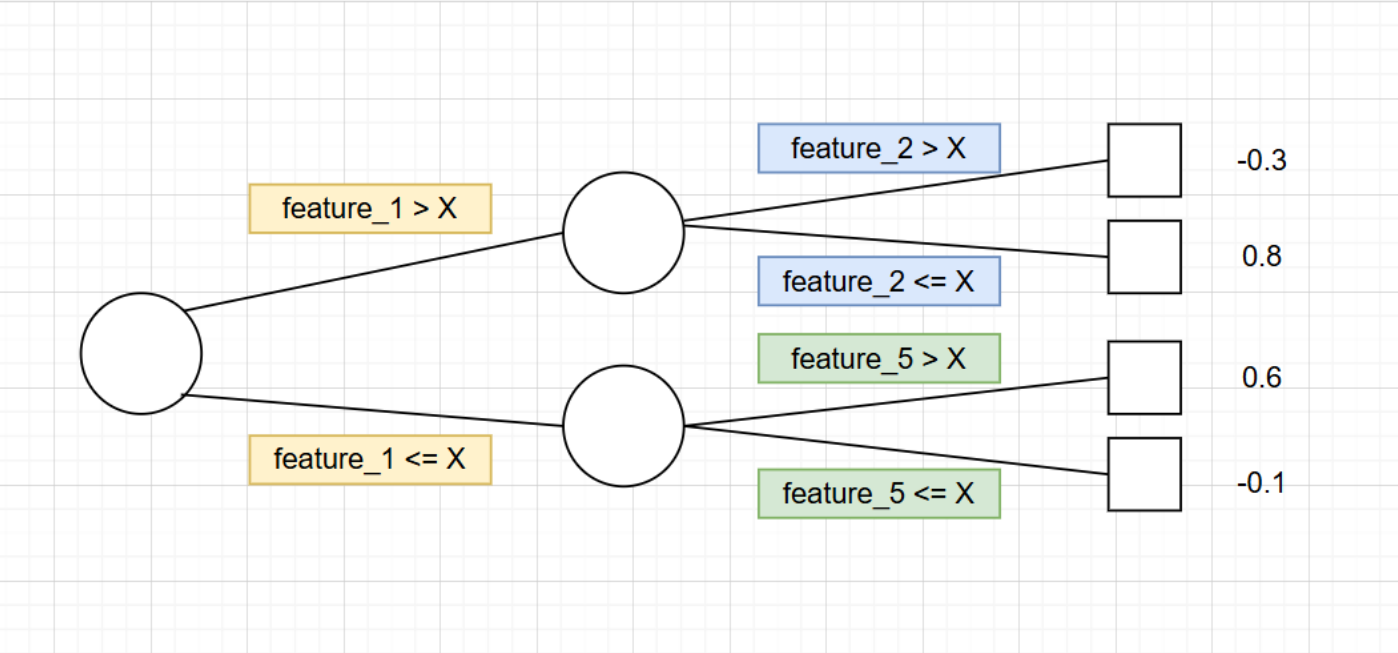
# XGBOOST

- **XGBoost** is an optimized distributed gradient boosting library designed to be highly **efficient**, **flexible** and **portable**.
  - C++, Java, Python,[3] R,[4] Julia,[5] Perl,[6] and Scala. It works on Linux, Microsoft Windows,[7] and macOS.[8]
- https://xgboost.readthedocs.io/en/stable/tutorials/model.html

# XGBOOST

- **Αναγνώριση κλάσεων:**
- K μοναδικές κλάσεις στο y_train
- K **ξεχωριστά ensembles** (σύνολα από δέντρα ) – ένα για κάθε κλάση.
- Στο τέλος της εκπαίδευσης, το XGBoost παράγει **K raw scores** (ένα για κάθε κλάση), τα οποία μετατρέπονται σε πιθανότητες μέσω της συνάρτησης ***softmax***.

# ΔΕΝΤΡΟ ΑΠΟΦΑΣΗΣ

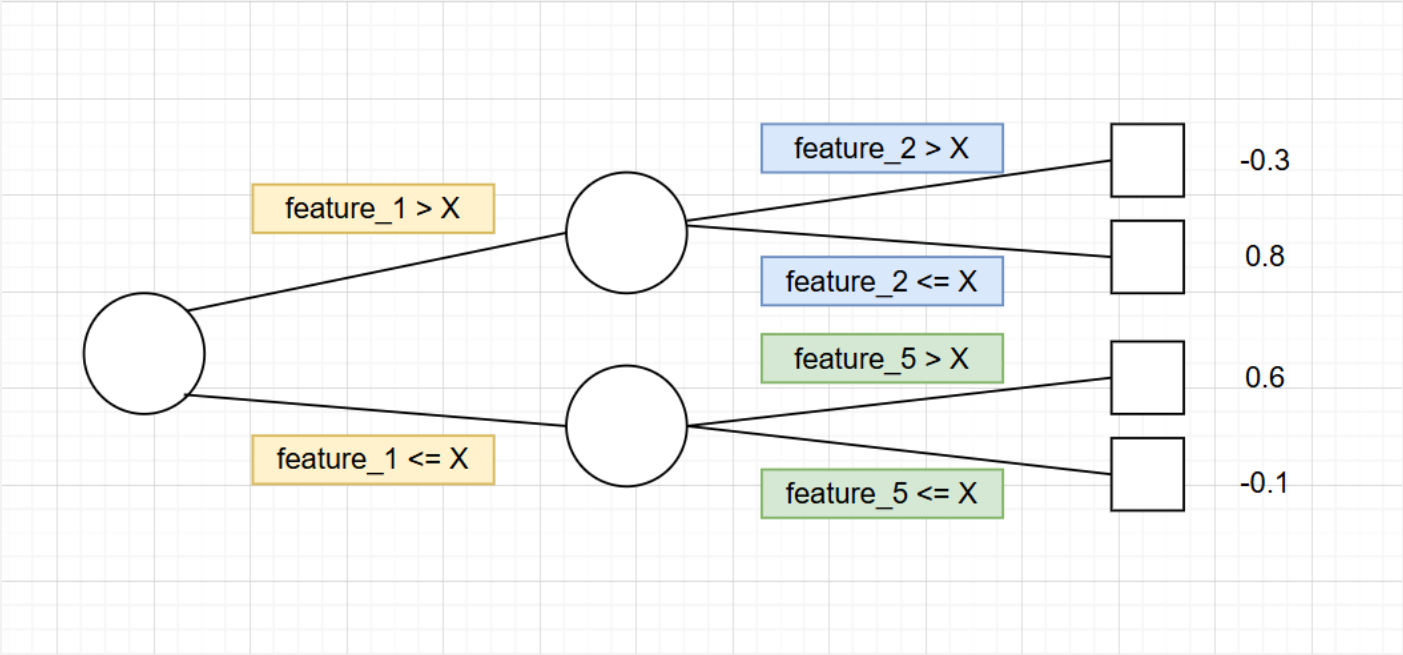


# ΔΕΝΤΡΟ ΑΠΟΦΑΣΗΣ

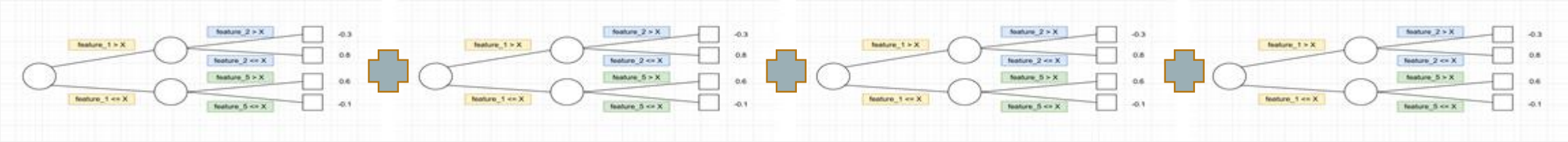
- **Ρίζα (Root Node)**: Ο αρχικός κόμβος που περιλαμβάνει όλα τα δεδομένα και ξεκινά τη διαδικασία διαχωρισμού.
- **Κόμβοι Απόφασης (Decision Nodes)**: Ενδιάμεσοι κόμβοι που διαχωρίζουν τα δεδομένα με βάση συγκεκριμένα χαρακτηριστικά.
- **Φύλλα (Leaf Nodes)**: Τελικοί κόμβοι που περιέχουν την πρόβλεψη (π.χ. ετικέτα κατηγορίας ή τιμή παλινδρόμησης).
- **Κριτήριο Διαχωρισμού (Splitting Criteria)**: Ο τρόπος με τον οποίο αποφασίζεται ο διαχωρισμός των δεδομένων. (log loss για ταξινόμηση)

Τα **πιθανά features** αντιστοιχούν στις 10 χιλιάδες στήλες του **TF-IDF matrix**, δηλαδή στα 10_000 συχνότερα μοναδικά διγράμματα/λέξεις που εμφανίζονται στα έγγραφα.

Οι τιμές των **Leaf Nodes** είναι raw scores/logits, δηλαδή δεν αντιπροσωπεύουν ακόμα πιθανότητα, αλλά είναι απλά τι υπολογίστηκε από την αντικειμενική συνάρτηση / objective function που χρησιμοποιεί ο XGBoost.



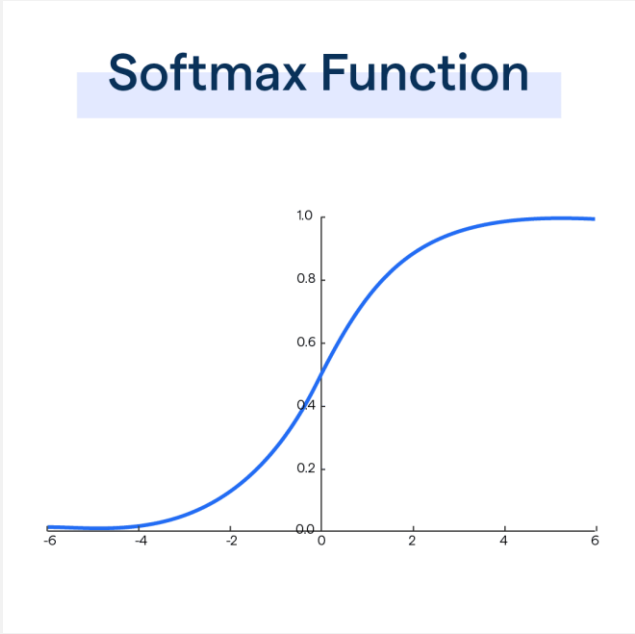
# ΣΥΝΔΥΑΣΜΟΣ ΔΕΝΤΡΩΝ



raw score

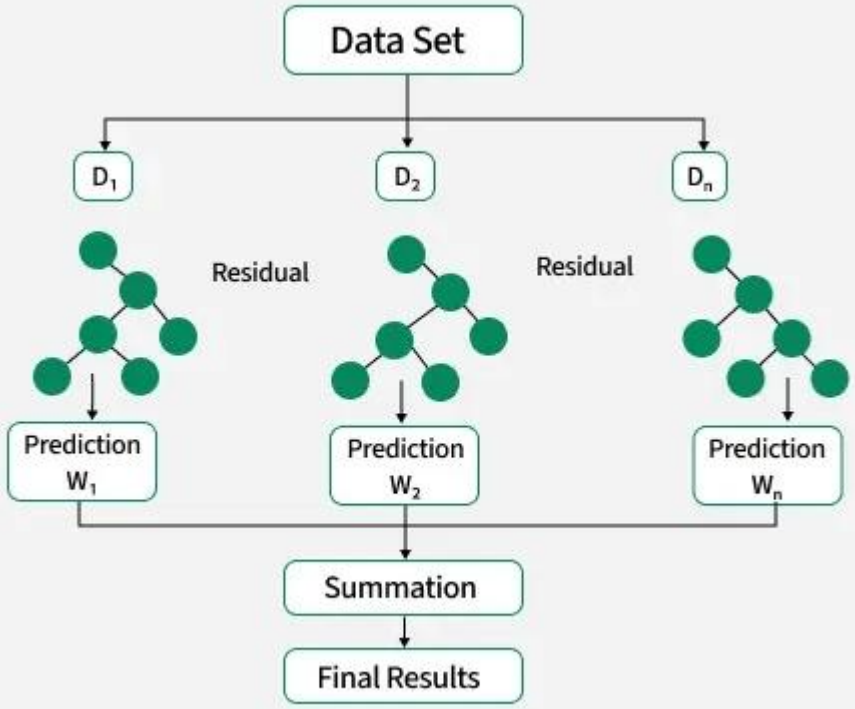
*K κλάσεις

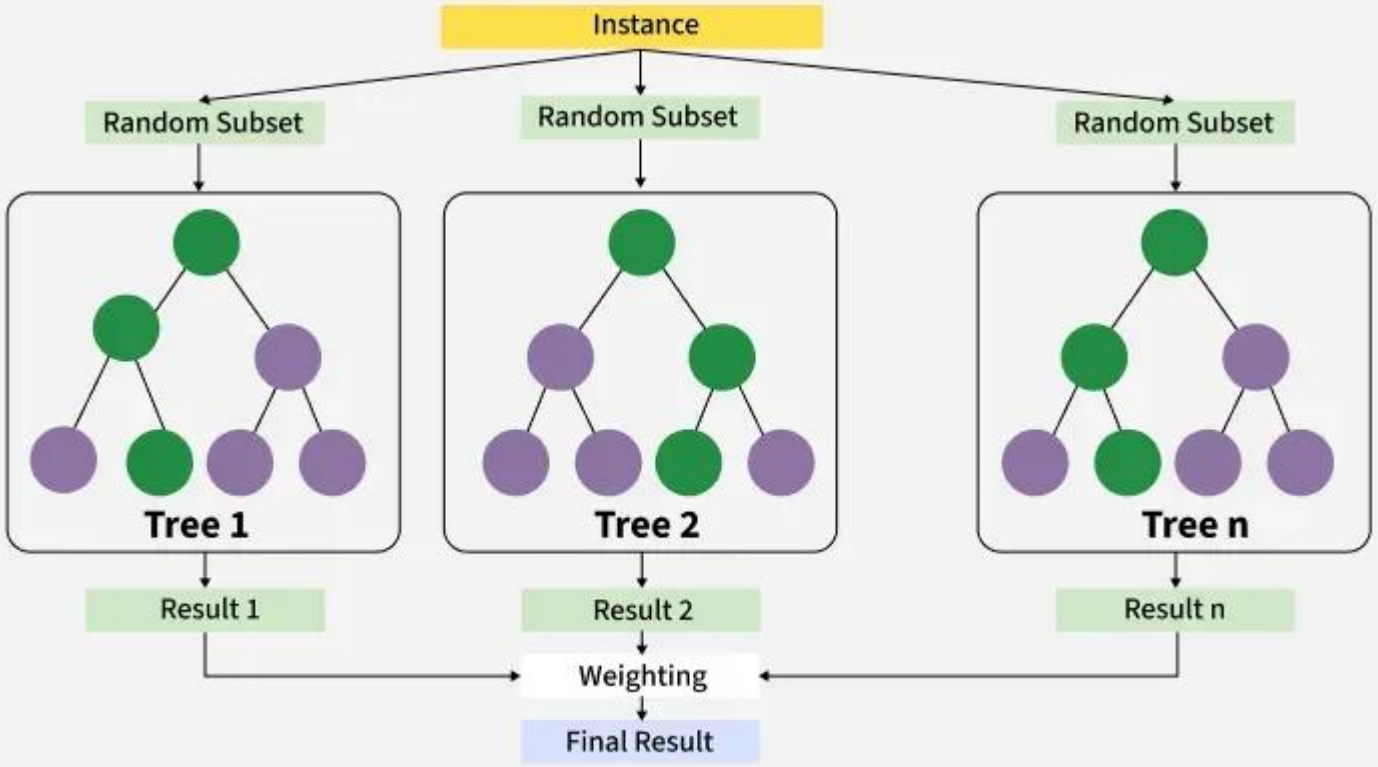
Raw score




probabilty

# ΕΙΚΟΝΕΣ

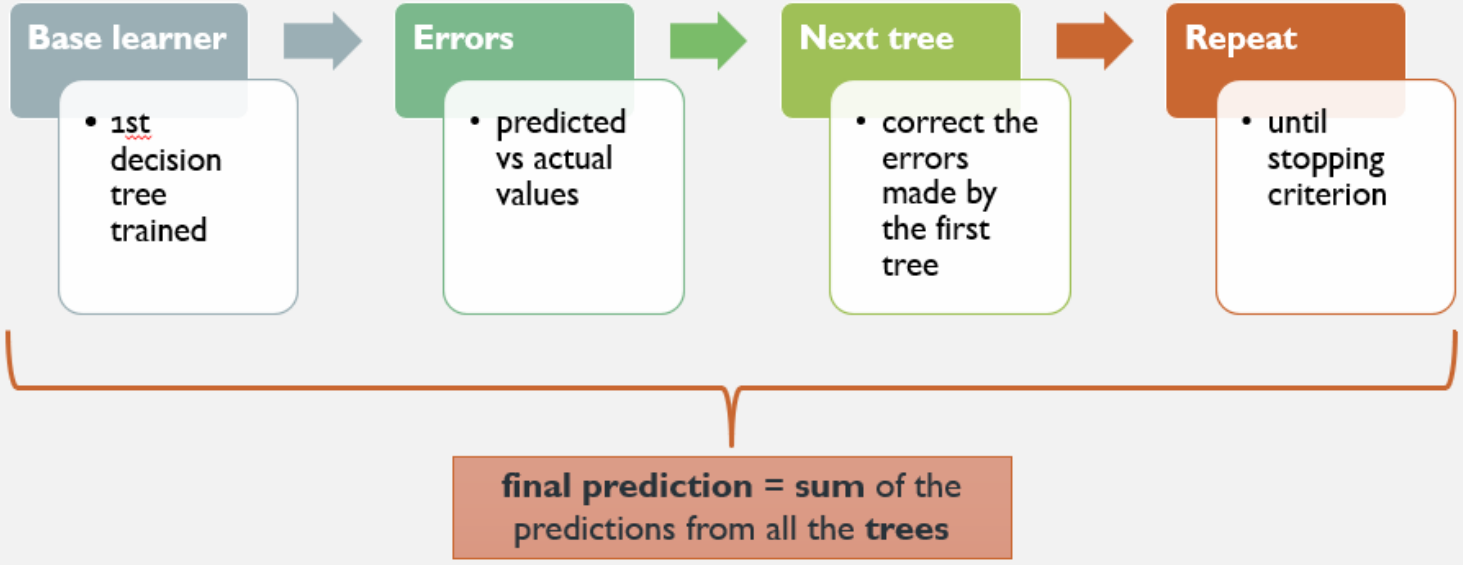

Data Set
D1
D2
Dn
Residual
Residual
Prediction W1
Prediction W2
Prediction Wn
Summation
Final Results


Instance
Random Subset
Random Subset
Random Subset
Tree 1
Tree 2
Tree n
Result 1
Result 2
Result n
Weighting
Final Result

# ΑΡΧΙΤΕΚΤΟΝΙΚΗ – ΣΥΝΤΟΜΗ ΕΞΗΓΗΣΗ

- Τα δέντρα είναι weak learners, δηλαδή απλές δομές που συνδυάζονται για να δημιουργήσουν ένα ισχυρό μοντέλο.
- Μικρό βάθος
- **Κάθε νέο δέντρο εκπαιδεύεται για να διορθώνει τα σφάλματα που έκανε το προηγούμενο δέντρο !!!**

# TRAINING


Base learner
• 1st decision tree trained
Errors
• predicted vs actual values
Next tree
• correct the errors made by the first tree
Repeat
• until stopping criterion
final prediction = sum of the predictions from all the trees

# ΔΗΜΙΟΥΡΓΙΑ ΔΙΑΚΛΑΔΩΣΕΩΝ - ΕΠΙΛΟΓΗ FEATURES ΔΕΝΤΡΩΝ ΑΠΌ ΤΟΝ XGBOOST

**1. Αντικειμενική Συνάρτηση του XGBoost:**

Objective Function = Training Loss + Regularization = **Λάθος Πρόβλεψης + Όρος Κανονικοποίησης**

$$obj = \sum_{i}^{n} l(y_i, y_i') + \sum_{k=1}^{t} \omega(f_k)$$

Όπου n τα training samples, l(y_i,y_I') = log loss function(true_label, predicted), δηλαδή η συνάρτηση απώλειας/λάθους και ω η πολυπλοκότητα του δέντρου και t το πλήθος των δέντρων.

Η συνάρτηση απώλειας/**loss function** που χρησιμοποιεί ο **XGBoost** είναι η **cross-entropy loss:**

**Loss = - [ y * log(y') +(1-y) * log(1-y')],** όπου y η σωστή ετικέτα και y' η προβλεπόμενη πιθανότητα να είναι η ετικέτα 1.

https://xgboost.readthedocs.io/en/stable/tutorials/model.html

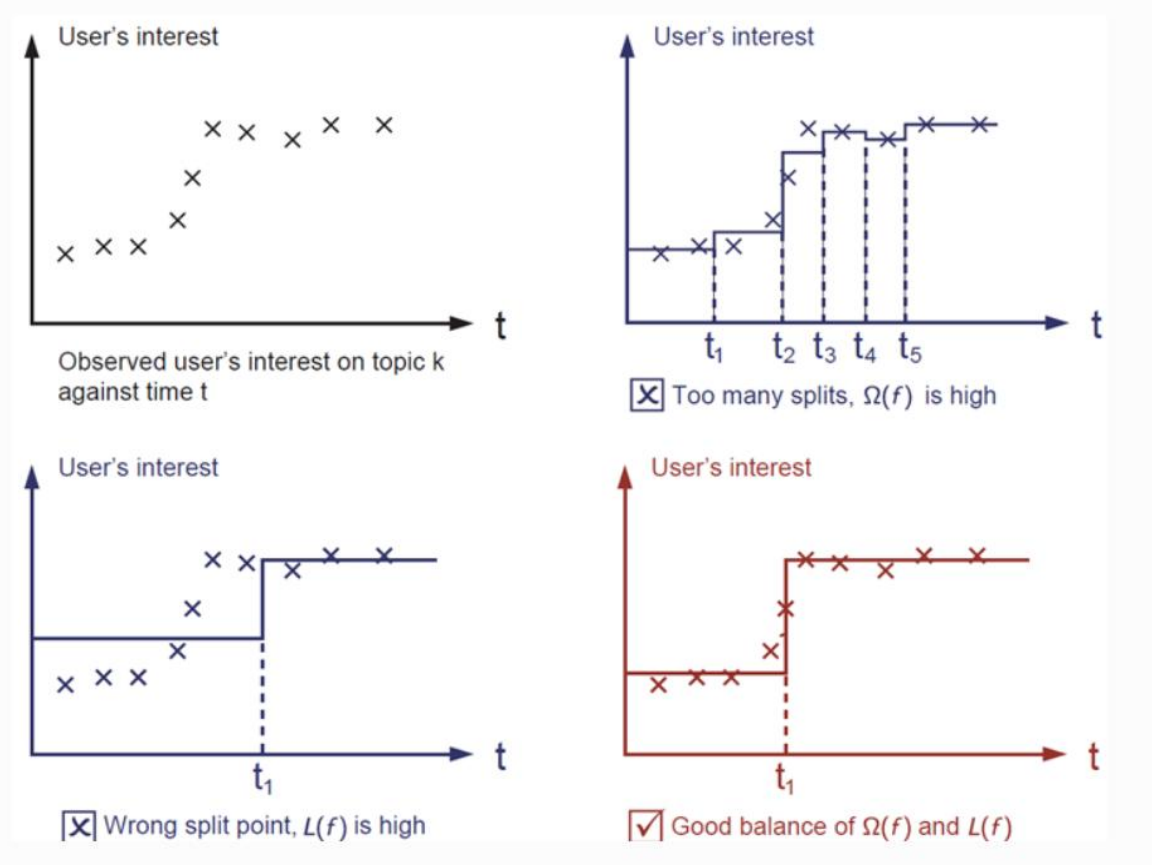
# ΚΑΝΟΝΙΚΟΠΟΙΗΣΗ

- Η περιπλοκότητα του δέντρου υπολογίζεται ως: **γ*Πλήθος Φύλλων + ½*λ*Άθροισμα Σκορ των φύλλων:**

$$\omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Στόχος είναι να **ελαχιστοποιηθούν** και οι **δύο όροι**, δηλαδή τα **predictions** να είναι σωστά, ενώ ταυτόχρονα το δέντρο να **μην** γίνεται υπερβολικά **περίπλοκο (πολλά φύλλα με μεγάλα σκορ)**. Το σκορ του φύλλου είναι ένα μέτρο που θα αναλυθεί παρακάτω.

# ΚΑΝΟΝΙΚΟΠΟΙΗΣΗ

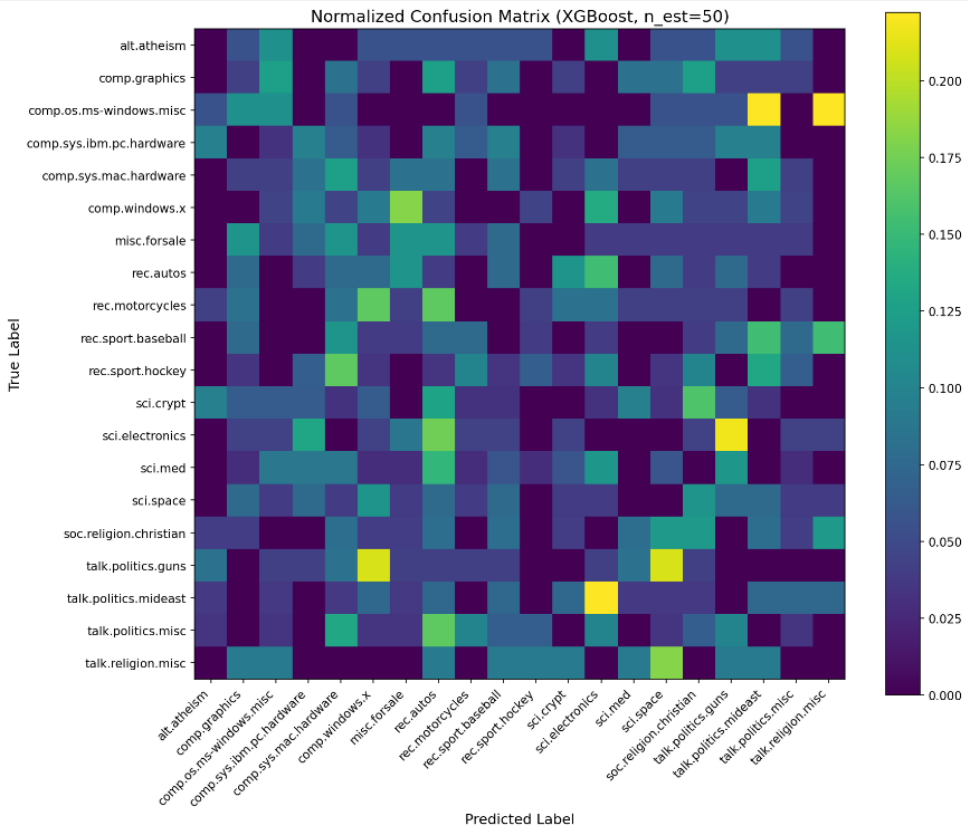


# HYPER-PARAMETERS

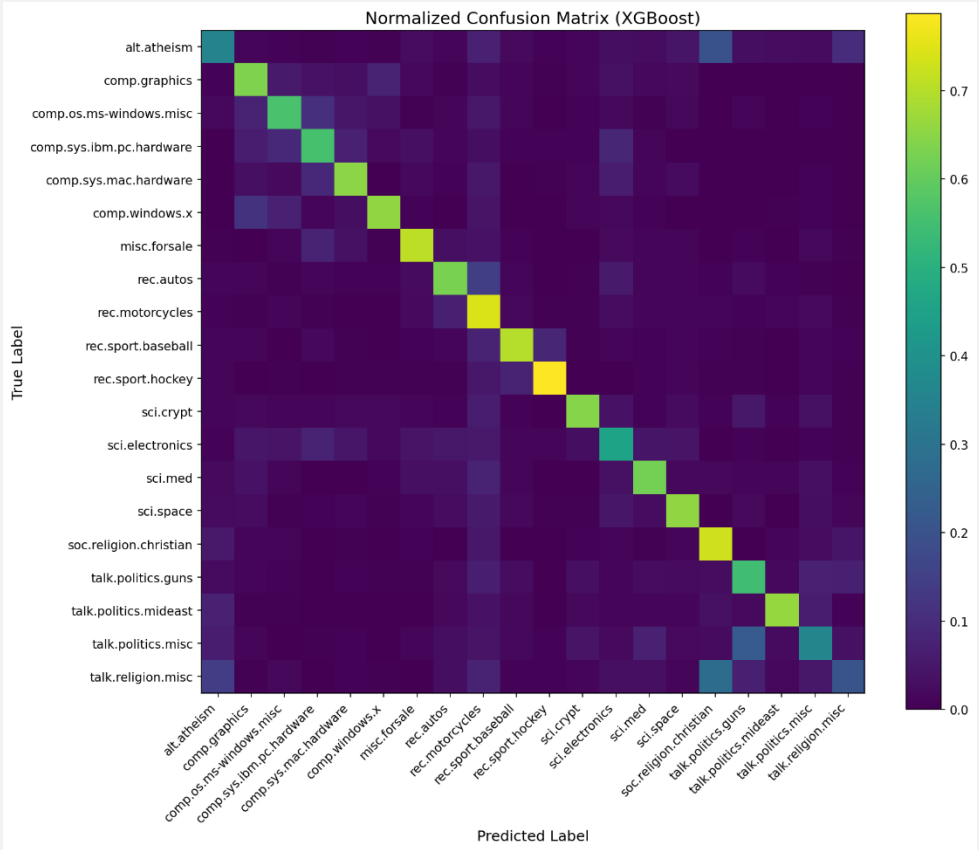
```
# Train XGBoost classifier
clf = XGBClassifier(
    n_estimators=100,
    max_depth=6,
    learning_rate=0.3,
    random_state=42,
    eval_metric='mlogloss',
    use_label_encoder=False
)
```

# CONFUSION MATRIX

N=50 depth=8 f=10000




N=100 depth=6 f=10000

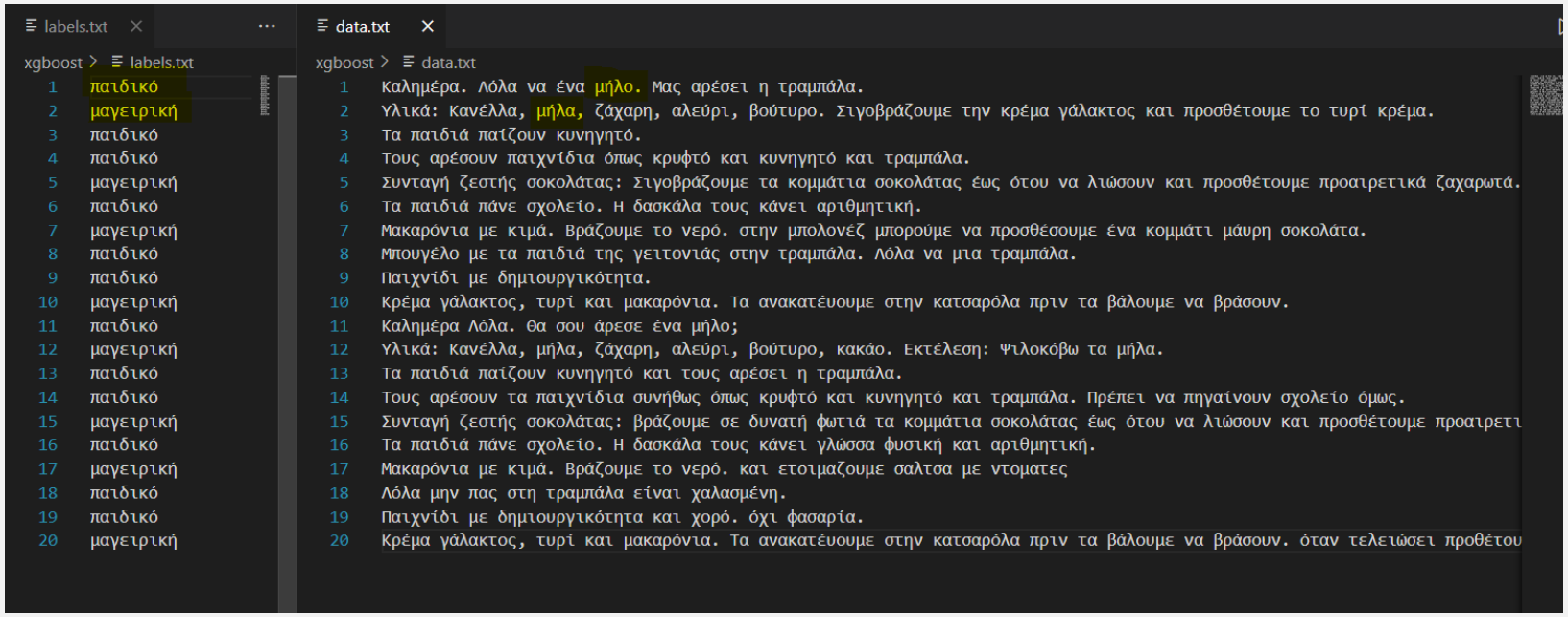


```
================================================================================
ARTICLE ANALYSIS (Index: 45)
================================================================================
TRUE LABEL: comp.graphics
PREDICTED: comp.graphics
CONFIDENCE: 0.184
CORRECT: ✓ YES

TOP 5 PREDICTIONS:
★ 1. comp.graphics            : 0.1842
   2. misc.forsale            : 0.1443
   3. sci.electronics         : 0.0911
   4. comp.windows.x          : 0.0904
   5. rec.sport.baseball      : 0.0789

ARTICLE TEXT:
--------------------------------------------------------------------------------
I've done a bit of looking, and havn't been able to
come up with a mailing list or newsgroup for users
of Adobe Photoshop.  Assuming I've just not missed
it, I'll go ahead and see if there is enough interest
to start a mailing list (and/or alt. newsgroup).

Drop me a note if  you might be interested in subscribing.

THANKS!

--Bob Wier (NOT of the Grateful Dead :-)
--------------------------------------------------------------------------------
```

# EXTRA CONSIDERATION

- Conda – Virtual environments
- Terminal cheatsheet

# INTUITION TF-IDF+XGBOOST



# TITANIC

# LINKS

- https://www.kaggle.com/competitions/titanic/overview
- https://www.kaggle.com/code/alexisbcook/titanic-tutorial

# LOGISTIC REGRESSION