Project για το μάθημα της Αναγνώρισης Προτύπων και Μηχανικής Μάθησης

Κιζιρίδης Δημήτριος ΑΕΜ: 10539
Μπίλλας Θωμάς Αχιλλέας ΑΕΜ: 10366
Ομάδα 33

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών ΑΠΘ

Τετάρτη 08 Ιανουαρίου 2025

Πίνακας Περιεχομένων

Μέρος Α - Εισαγωγή

Στόχος του μέρους Α είναι να διαπιστώσουμε κατά πόσο ο δείκτης – αριθμός x, που σχετίζεται με τα μοτίβα συχνότητας και δύναμης πίεσης των πλήκτρων της κονσόλας μπορεί να χρησιμοποιηθεί σε ένα σύστημα ταξινόμησης σχετικά με το στρες των παικτών βιντεοπαιχνιδιών. Έτσι, με γνωστή κατανομή πυκνότητας πιθανότητας, καλούμαστε να υλοποιήσουμε έναν ταξινομητή με τη **Μέθοδο της Μέγιστης Πιθανοφάνειας** για να αξιολογήσουμε το δείκτη x, με βάση δεδομένα που έχουμε αναφορικά με το αν νιώθουν στρες ή όχι οι χρήστες.
Στο ερώτημα 1 θα εκτιμηθούν άγνωστες παράμετροι και θα απεικονιστούν τα ζητούμενα διαγράμματα.
Στο ερώτημα 2 θα χρησιμοποιηθεί δοθείσα συνάρτηση διάκρισης, θα ταξινομηθούν τα δείγματα που έχουμε και θα διατυπωθούν παρατηρήσεις.

Μέρος Α - Δεδομένα

Η κατανομή πυκνότητας πιθανότητας που ακολουθεί αυτός ο δείκτης x για τις δύο κλάσεις (χωρίς στρες = κλάση ω_1, με στρες = κλάση ω_2) είναι:

$$p(x|\theta) = \frac{1}{\pi} \frac{1}{1 + (x - \theta)^2}$$

όπου η παράμετρος θ είναι άγνωστη.
Έχοντας ένα δείγμα 12 ατόμων και γνωρίζοντας αν δήλωσαν ότι αισθάνθηκαν στρες ή όχι (7 δήλωσαν όχι, 5 δήλωσαν ναι), παίρνουμε τους δείκτες:
$D_1 = [2.8, -0.4, -0.8, 2.3, -0.3, 3.6, 4.1]$, για την κλάση ω_1
$D_2 = [-4.5, -3.4, -3.1, -3.0, -2.3]$, για την κλάση ω_2.
Δινεται για το ερώτημα 2 η συνάρτηση διάκρισης:

$$g(x) = \log P(x|\hat{\theta}_1) - \log P(x|\hat{\theta}_2) + \log P(\omega_1) - \log P(\omega_2)$$

Μέρος Α - Ερώτημα 1 - Θεωρητική ανάλυση

Για να εκτιμήσουμε τις $\hat{\theta}_1, \hat{\theta}_2$ για την παράμετρο θ μας ζητείται να χρησιμοποιήσουμε τη Μέθοδο Μεγίστης Πιθανοφάνειας, άρα με βάση τη θεωρία πρέπει να βρεθούν οι τιμές που θα μεγιστοποιήσουν τις συναρτήσεις πιθανοφάνειας $p(D_1|\theta)$ και $p(D_2|\theta)$ αντίστοιχα.
Γνωρίζουμε γενικά πως:

$$p(D|\theta) = p(x_1, x_2, \ldots, x_N|\theta) = \prod_{n=1}^{N} p(x_n|\theta) \tag{1}$$

όπου $p(x|\theta) = \frac{1}{\pi} \frac{1}{1+(x-\theta)^2}$ και τα $x_1, x_2, \ldots, x_N \in D$.
Έτσι βρίσκουμε τις ζητούμενες τιμές $\hat{\theta}_1, \hat{\theta}_2$ που μεγιστοποιούν τις $p(D_1|\theta)$ και $p(D_2|\theta)$ αντίστοιχα.

Μέρος Α - Ερώτημα 1 - Υλοποίηση

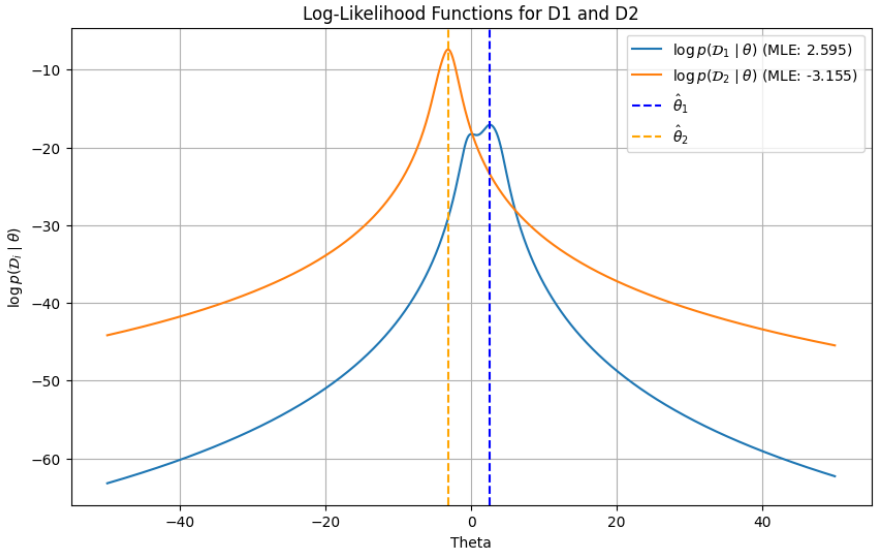
Προχωρόντας στην υλοποίηση λογαριθμίζουμε τη σχέση (1), αφενός γιατί αποτελεί καλή πρακτική για απλοποίηση των υπολογισμών και αφετέρου γιατί μας ζητείται να απεικονίσουμε τις $\log p(\mathcal{D}_1|\theta)$ και $\log p(\mathcal{D}_2|\theta)$.

$$\log p(D|\theta) = \log \prod_{n=1}^{N} p(x_n|\theta)$$

Για να πετύχουμε πολύ καλή ακρίβεια στις εκτιμήσεις μας, στην υλοποιήση χρησιμοποιούμε 10000 τιμές στο διάστημα [-5,+5], έχοντας βέβαια επιβεβαίωσει ότι στην περιοχή αυτή βρίσκονται τα ζητούμενα μέγιστα.

```
theta_values = np.linspace(-5, 5, 1000)
```

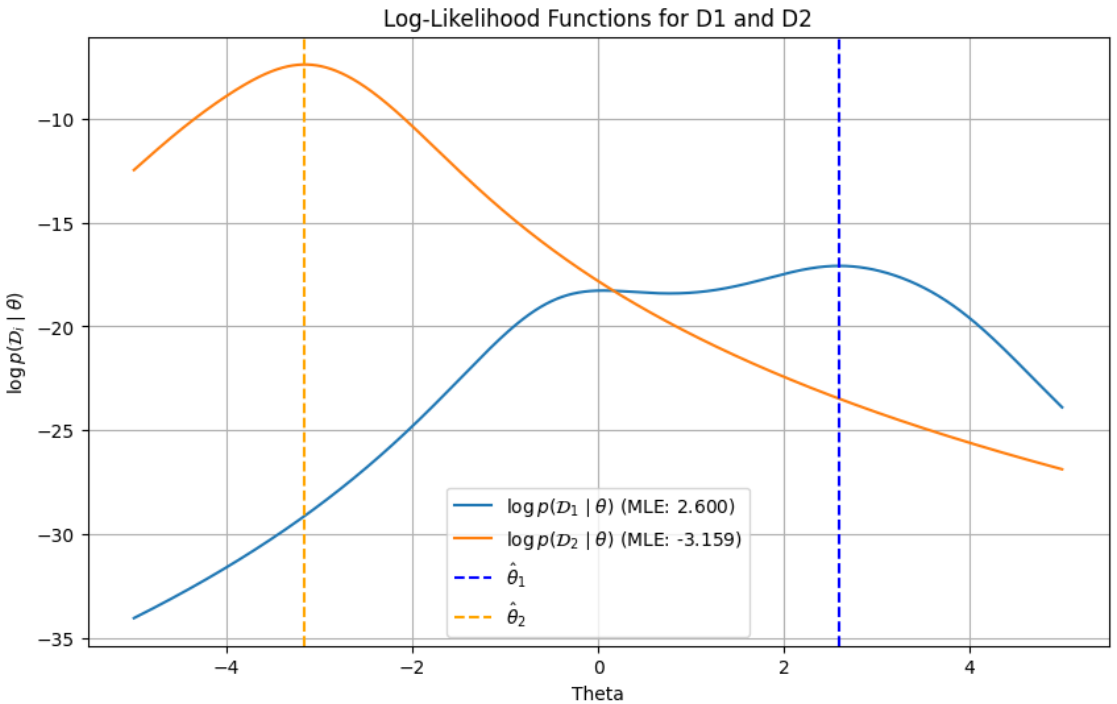
Μέρος Α - Ερώτημα 1 - Υλοποίηση



Επιβεβαιώνουμε οπτικά την περιοχή στην οποία θα βρεθούν τα μέγιστα, και απεικονίζουμε τις ζητούμενες συναρτήσεις λογαριθμικής πιθανοφάνειας $\log p(\mathcal{D}_1|\theta)$ και $\log p(\mathcal{D}_2|\theta)$ συναρτήσει του θ.

Οι ζητούμενες παράμετροι, με ακρίβεια 3 δεκαδικών ψηφίων προκύπτουν:

$$\hat{\theta}_1 = 2.600 \text{ και } \hat{\theta}_1 = -3.159$$



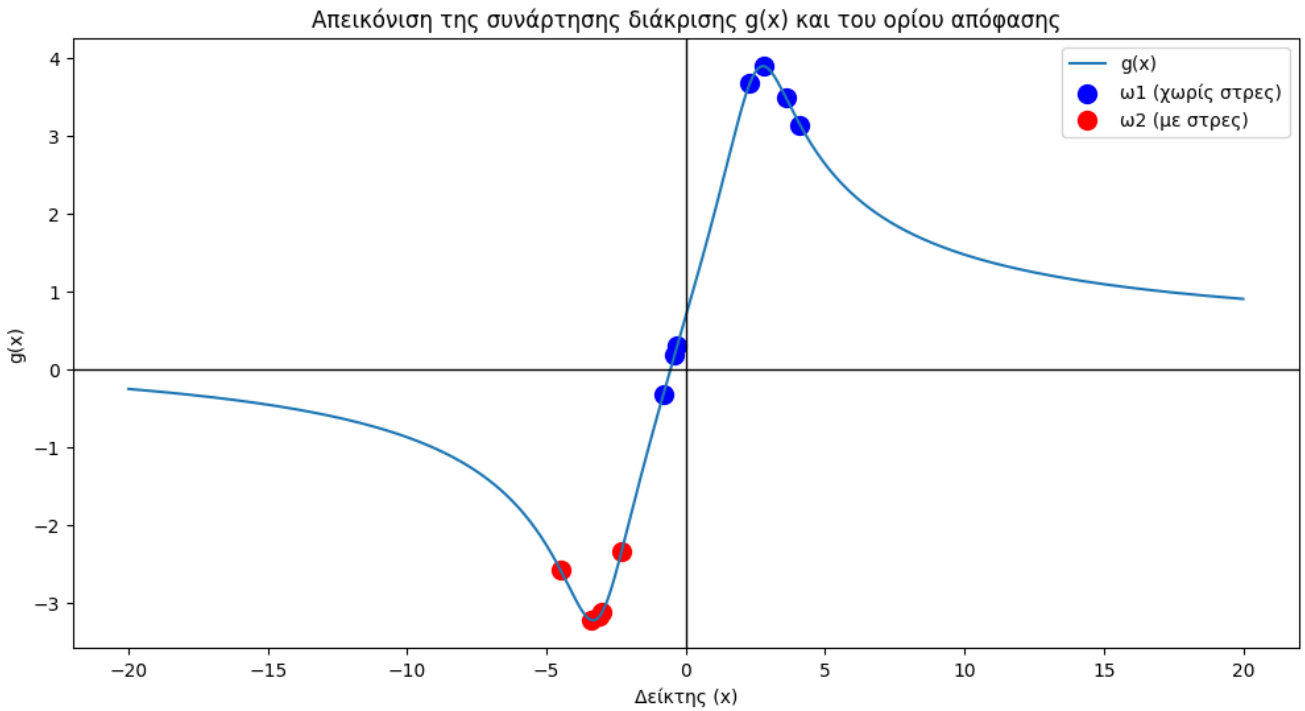
Μέρος Α - Ερώτημα 2 - Θεωρητική ανάλυση

Στη συνέχεια για την ταξινόμηση θα χρησιμοποιήσουμε τη συνάρτηση διάκρισης που μας δίνεται.

$$g(x) = \log P(x|\hat{\theta}_1) - \log P(x|\hat{\theta}_2) + \log P(\omega_1) - \log P(\omega_2)$$

Η συνάρτηση και όροι της μας είναι γνώριμοι από τη θεωρία και το γενικό κανόνα του Bayes και γνωρίζουμε - παρατηρούμε πως το πρόσημο της $g(x)$ καθορίζει σε ποια κλάση θα τοποθετηθούν τα δείγματα, δηλαδή για $g(x) > 0$ το δείγμα τοποθετείται στην κλάση ω_1, ενώ για $g(x) < 0$ το δείγμα τοποθετείται στην κλάση ω_2.
Οι $P(\omega_1)$ και $P(\omega_2)$ προκύπτουν από τον λόγο των δειγμάτων που ανήκουν σε κάθε κλάση προς τον συνολικό αριθμό δειγμάτων που μας δίνονται. Ενώ, οι $\log P(x|\hat{\theta}_1)$ και $\log P(x|\hat{\theta}_2)$ προκύπτουν από τη δοθείσα σχέση της εκφώνησης.

Μέρος Α - Ερώτημα 2 - Υλοποίηση και Παρατηρήσεις



Απεικονίζουμε την $g(x)$ και το όριο απόφασης, καθώς και τα δείγματα που μας δίνονται. Παρατηρούμε πως ένα δείγμα $(x = -0.8)$ ταξινομείται λάθος, ανήκει στην κλάση ω_1 αλλά ταξινομείται στην ω_2.

Υπενθυμίζουμε πως ο κανόνας απόφασης είναι ω_1 αν $g(x) > 0$ και ω_2 αν $g(x) < 0$.

Μέρος Β - Εισαγωγή

Στόχος του μέρους Β είναι να υλοποιήσουμε νέο ταξινομητή για την εκτίμηση της άγνωστης παραμέτρου θ με τη **Μέθοδο Εκτίμησης κατά Bayes**. Η prior συνάρτηση πυκνότητας πιθανότητας είναι γνωστή και μας ζητούνται τα εξής:

- Στο ερώτημα 1 να υπολογιστούν και να απεικονιστούν οι εκ των υστέρων πιθανότητες $p(\theta|D_1)$ και $p(\theta|D_2)$.
- Στο ερώτημα 2 θα χρησιμοποιηθεί δοθείσα συνάρτηση διάκρισης, θα ταξινομηθούν τα δείγματα που έχουμε και θα διατυπωθούν παρατηρήσεις.

Μέρος Β - Δεδομένα

Από το Μέρος Α, είναι ίδια η $p(x|\theta)$, τα δείγματα D_1 και D_2, ενώ έχουμε την επιπλέον πληροφορία για την prior συνάρτηση πυκνότητας πιθανότητας, η οποία είναι:

$$p(\theta) = \frac{1}{10\pi} \frac{1}{1 + \left(\frac{\theta}{10}\right)^2}$$

Δινεται για το ερώτημα 2 η συνάρτηση διάκρισης:

$$g(x) = \log P(x|D_1) - \log P(x|D_2) + \log P(\omega_1) - \log P(\omega_2)$$

Μέρος Β - Ερώτημα 1 - Θεωρητική ανάλυση

Για να εκτιμήσουμε τις $p(\theta|D_1)$ και $p(\theta|D_2)$ έχοντας το γνωστό μοντέλο και με βάση τη θεωρία θα χρησιμοποιήσουμε τη γνωστή από το μάθημα σχέση:

$$p(\theta \mid D) = \frac{p(D \mid \theta)p(\theta)}{\int p(D \mid \theta)p(\theta)\, d\theta}$$

όπου $p(D \mid \theta) = \prod_{n=1}^{N} p(x_n \mid \theta)$, και μπορεί να υπολογιστεί όπως στο Μέρος Α.
Οι $p(\theta)$ υπολογίζονται από τη σχέση της εκφώνησης.

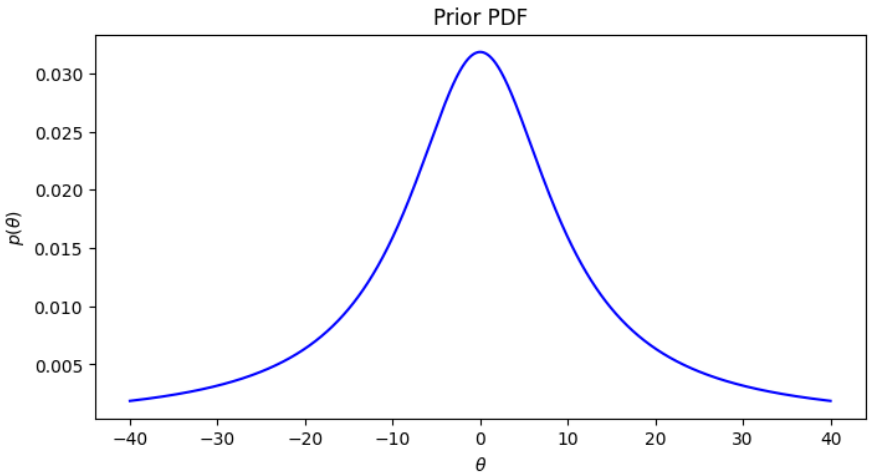
Μέρος Β - Ερώτημα 1 - Υλοποίηση

Προχωρώντας στην υλοποίηση, υπολογίζουμε το ζητούμενο με τον κανόνα του Bayes και καθ' υπόδειξιν της εκφώνησης θα χρησιμοποίησουμε κανόνα τραπεζίου για να υπολογίσουμε το ολοκλήρωμα στην έκφραση των $p(\theta|D)$.

```
def integralTrapezoid(self, f:np.array, dx:float) -> float:
    return np.sum((f[1:] + f[:-1]) / 2 * dx)
```

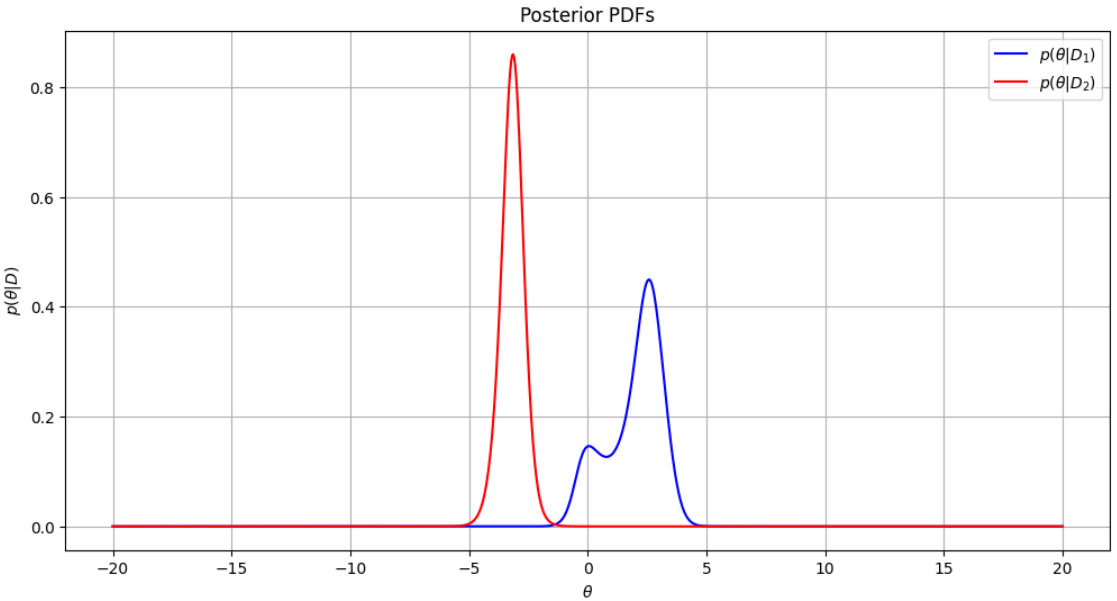
```
def calculate_posterior(self, D, thetas):
    dtheta = thetas[1] - thetas[0]
    probs_x_theta = np.array([self.likelihood(D, theta) for
        theta in thetas])
    probs_D_theta = np.prod(probs_x_theta, axis=1)
    probs_theta = self.prior(thetas)
    integral = self.integralTrapezoid(probs_D_theta *
        probs_theta, dtheta)
    posterior = (probs_D_theta * probs_theta) / integral
    return posterior
```

Μέρος Β - Ερώτημα 1 - Υλοποίηση



Απεικονίζουμε την prior $p(\theta)$, αφού είναι γνωστή συνάρτηση από την εκφώνηση, στο διάστημα [-40,+40] και χρησιμοποιούμε 1000 τιμές για είναι πλήρως εμφανές το σχήμα της.

Απεικονίζουμε τις ζητούμενες εκ των υστέρων πυκνότητες πιθανότητες $p(\theta|D_1)$ και $p(\theta|D_2)$ στο διάστημα [-20,+20] και χρησιμοποιώντας 1000 δείγματα.



Μέρος Β - Ερώτημα 1 - Υλοποίηση - Παρατηρήσεις

Με βάση τα ζητούμενα διαγράμματα μπορούμε να εξάγουμε τις εξής παρατηρήσεις:

- Οι εκ των υστέρων πυκνότητες πιθανότητας παρουσιάζουν υψηλή συγκέντρωση γύρω από τις πιο πιθανές τιμές των παραμέτρων, με τη μέγιστη τιμή να βρίσκεται κοντά στα σημεία που βρέθηκε και στο μέρος Α, χρησιμοποιώντας τη μέθοδο της Μέγιστης Πιθανοφάνειας. Ειδικότερα για την κλάση ω_2 η εκτίμηση της παραμέτρου είναι με πολύ μεγάλη πιθανότητα.
- Από την εκ των προτέρων συνάρτηση πυκνότητας πιθανότητας δε μπορούμε να εξάγουμε κάποια πληροφορία, καθώς τα δεδομένα δεν την επηρεάζουν.
- Όπως είδαμε και στη θεωρία έχουμε "μετατρέψει" την εκτίμηση μας σε μια πιο ακριβή αφού έχουμε την πληροφορία των δεδομένων μας (a-posteriori) σε σχέση με πριν τα παρατηρήσουμε (a-priori).

Μέρος Β - Ερώτημα 2 - Θεωρητική Ανάλυση

Στη συνέχεια για την ταξινόμηση θα χρησιμοποιήσουμε τη συνάρτηση διάκρισης που μας δίνεται.

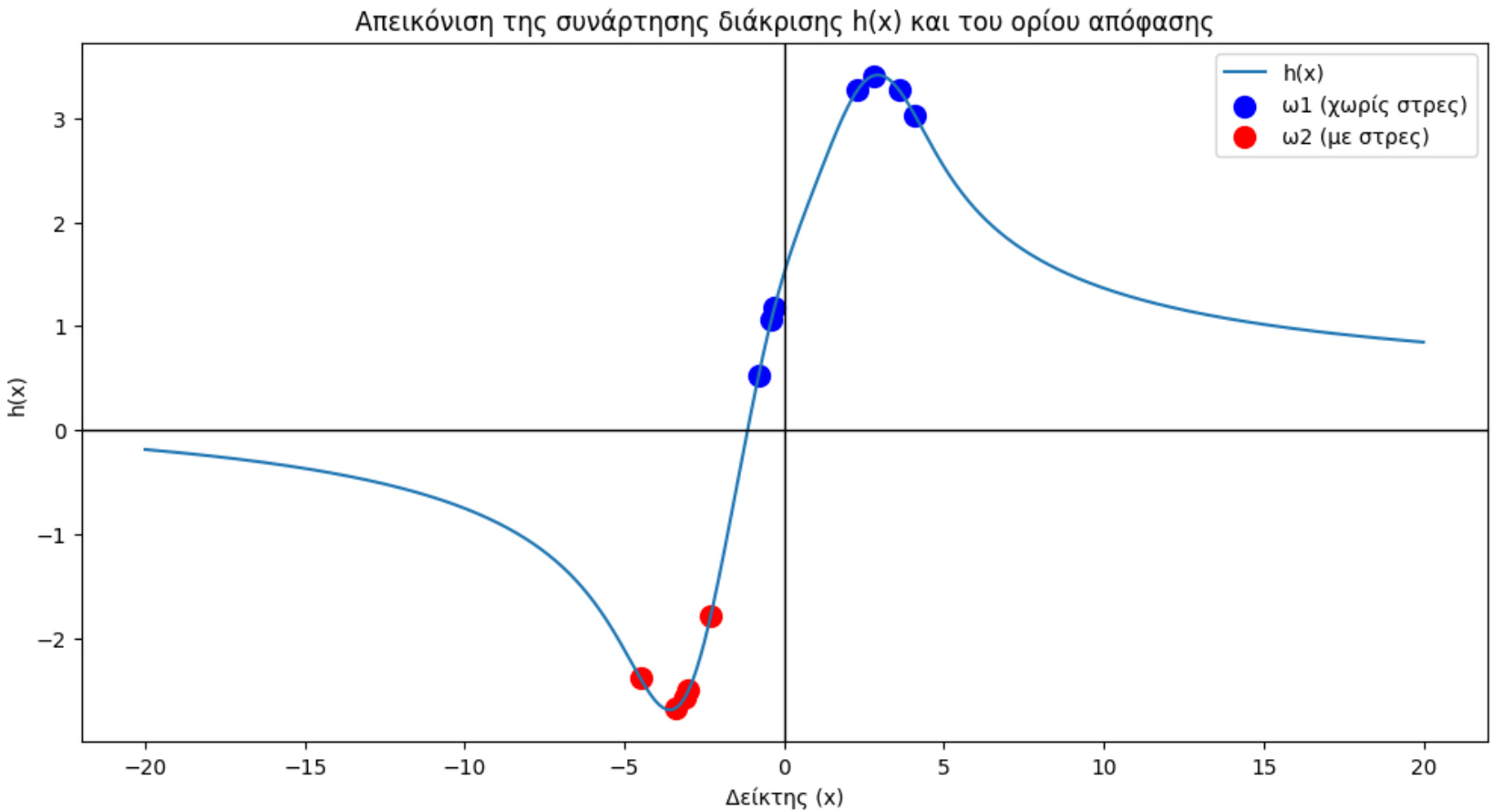
$$g(x) = \log P(x|D_1) - \log P(x|D_2) + \log P(\omega_1) - \log P(\omega_2)$$

Η συνάρτηση και όροι της μας είναι γνώριμοι από τη θεωρία και το γενικό κανόνα του Bayes και έτσι υπολογίζουμε την $P(x|D)$ από τη γνωστή σχέση:

$$p(x|D) = \int p(x|\theta)p(\theta|D)\, d\theta$$

Οι $P(\omega_1)$ και $P(\omega_2)$ προκύπτουν όπως στο μέρος Α από τον λόγο των δειγμάτων που ανήκουν σε κάθε κλάση προς τον συνολικό αριθμό δειγμάτων που μας δίνονται.
Επιπλέον παρατηρούμε - γνωρίζουμε πως το πρόσημο της $h(x)$ καθορίζει σε ποια κλάση θα τοποθετηθούν τα δείγματα, δηλαδή για $h(x) > 0$ το δείγμα τοποθετείται στην κλάση ω_1, ενώ για $h(x) < 0$ το δείγμα τοποθετείται στην κλάση ω_2.

Μέρος Β - Ερώτημα 2 - Υλοποίηση και Παρατηρήσεις



Απεικονίζουμε την $h(x)$ και το όριο απόφασης, καθώς και τα δείγματα που μας δίνονται. Παρατηρούμε πως όλα τα δείγματα ταξινομούνται σωστά. Υπενθυμίζουμε πως ο κανόνας απόφασης είναι ω_1 αν $h(x) > 0$ και ω_2 αν $h(x) < 0$.

Μέρος Β - Σύγκριση Μέγιστης Πιθανοφάνειας και Εκτίμησης κατά Bayes

Έχοντας απεικονίσει τα όρια απόφασης και για τους δύο ταξινομητές, αλλά και τις $g(x)$ και $h(x)$ μπορούμε να εξάγουμε μερικά κρίσιμα συμπεράσματα:

- Τα όρια απόφασης έχουν την ίδια μορφολογία και μοιάζουν αρκετά χωρίς όμως να είναι ακριβώς ίδια.
- Η εκτίμηση κατά Bayes παρουσιάζει καλύτερα αποτέλεσματα στο συγκεκριμένο πρόβλημα αφού ταξινομεί ορθά όλα τα δείγματα, σε αντίθεση με τον ταξινομητή Μέγιστης Πιθανοφάνειας που ταξινομεί ένα λάθος.
- Όπως γνωρίζουμε ο εκτιμητής κατά Bayes λόγω του ολοκληρώματος έχει μεγαλύτερη υπολογιστική πολυπλοκότητα κάτι που ενδέχεται να μας προβληματίσει σε περιπτώσεις με περισσότερες διαστάσεις όμως μπορεί να οδηγήσει σε πιο ακριβή αποτελέσματα όπως φαίνεται και στο συγκεκριμένο ερώτημα, λόγω της εκμετάλλευσης της γνώσης της a-priori πιθανότητας.

Μέρος Γ - Εισαγωγή

Το μέρος Γ χωρίζεται σε 2 διακριτές ενότητες.
Στην πρώτη ενότητα ασχολούμαστε με Δέντρα Απόφασης. Έτσι, μας ζητείται να κατεβάσουμε το dataset Iris της βιβλιοθήκης sklearn και απομονώνοντας κάποια χαρακτηριστικά αυτού να χρησιμοποιήσουμε τον αλγόριθμο DecisionTreeClassifier της ίδια βιβλιοθήκης για να εκπαιδεύσουμε το μοντέλο, να ταξινομήσουμε ορισμένα δείγματα και να εξάγουμε συμπεράσματα.
Στη δεύτερη ενότητα μας ζητείται να δημιουργήσουμε ένα ταξινομητή Random Forest με τη τεχνική Bootstrap στο ίδιο dataset, να ταξινομήσουμε ορισμένα δείγματα, να αναλύσουμε την επίδραση του γ στα αποτελέσματα μας και να εξάγουμε ορισμένα συμπεράσματα.

Μέρος Γ - Ενότητα 1 - Υλοποίηση - Ερώτημα 1

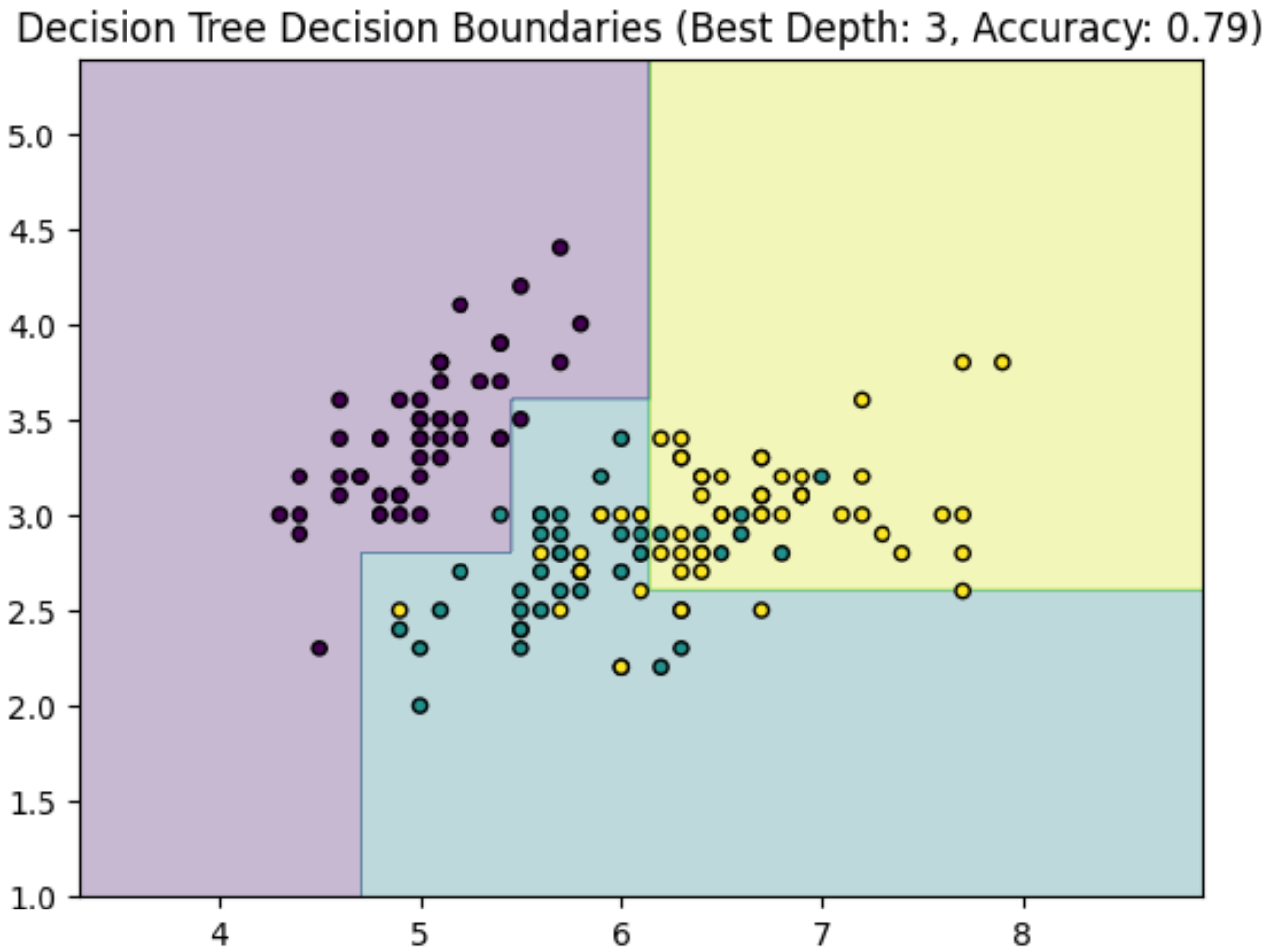
Το βάθος του δέντρου αποτελεί μία υπερπαράμετρο του προβλήματος. Χρησιμοποιούμε τον έτοιμο αλγόριθμο DecisionTreeClassifier της sklearn και χρησιμοποιώντας σαν μετρική το accuracy βρίσκουμε το καλύτερο βάθος για το Δέντρο Απόφασής μας, έχοντας αναζητήσει σε εύρος τιμών από 1 έως 12.

```
best_depth = 0
best_accuracy = 0
for depth in range(1, 12):
    clf = DecisionTreeClassifier(max_depth=depth,
        random_state=42)
    clf.fit(X_train, y_train)
    accuracy = clf.score(X_test, y_test)
    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_depth = depth
```

Καταλήγουμε πως την καλύτερη ακρίβεια (0.79) την επιτυγχάνουμε για βάθος 3.

Μέρος Γ - Ενότητα 1 - Υλοποίηση - Ερώτημα 2

Χρησιμοποιώντας όπως μας ζητείται τη συνάρτηση contourf της matplotlib.pyplot απεικονίζουμε τα ζητούμενα όρια απόφασης.



Μέρος Γ - Ενότητα 2 - Υλοποίηση - Ερώτημα 1

Δημιουργούμε ταξινομητή 100 δέντρων με $\gamma = 50\%$.

```
n_trees = 100
gamma = 0.5  #50% of the training set
n_samples = int(gamma * len(X_train))
```

Το βάθος του κάθε Δέντρου Απόφασης αποτελεί και πάλι μία υπερπαράμετρο του προβλήματος.

```
best_rf_depth = 0
best_rf_accuracy = 0
for depth in range(1, 12):
    rf_clf = RandomForestClassifier(n_estimators=n_trees,
        max_depth=depth, bootstrap=True, random_state=42,
        max_samples=gamma)
    rf_clf.fit(X_train, y_train)
    accuracy = rf_clf.score(X_test, y_test)
    if accuracy > best_rf_accuracy:
        best_rf_accuracy = accuracy
        best_rf_depth = depth
```

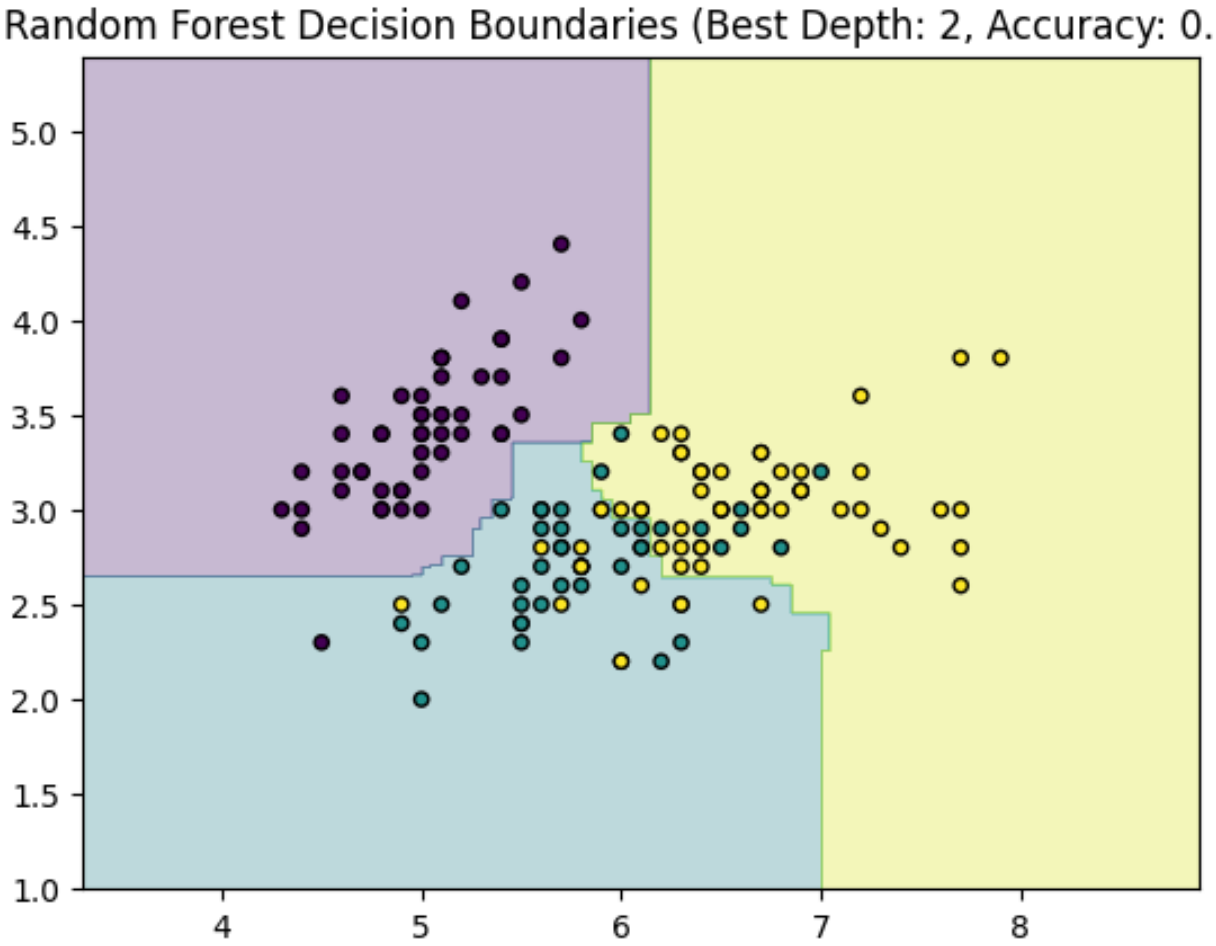
Μέρος Γ - Ενότητα 2 - Υλοποίηση - Ερώτημα 1

Υλοποιούμε δικό μας ταξινομητή χρησιμοποιώντας την RandomForestClassifier της sklearn και πάλι χρησιμοποιώντας σαν μετρική το accuracy βρίσκουμε το καλύτερο βάθος για το κάθε Δέντρο Απόφασης στο Random Forest μας, έχοντας αναζητήσει σε εύρος τιμών από 1 έως 12.
Σε αυτήν την περίπτωση καταλήγουμε πως την καλύτερη ακρίβεια (0.83) την επιτυγχάνουμε για μέγιστο 2.

Σχόλιο: Θα μπορούσαμε να χρησιμοποιήσουμε τη συνάρτηση GridSearchCV της sklearn.model_selection, την οποία μελετήσαμε και στο εργαστήριο στα πλαίσια του μαθήματος, αφού όμως η συνάρτηση αυτή θα χρησιμοποιηθεί σε μεγάλο βαθμό στο Μέρος Δ, επιλέξαμε να δείξουμε μία διαφορετική λύση στο μέρος αυτό.

Μέρος Γ - Ενότητα 2 - Υλοποίηση - Ερώτημα 2

Χρησιμοποιώντας όπως μας ζητείται τη συνάρτηση contourf της matplotlib.pyplot απεικονίζουμε τα ζητούμενα όρια απόφασης.



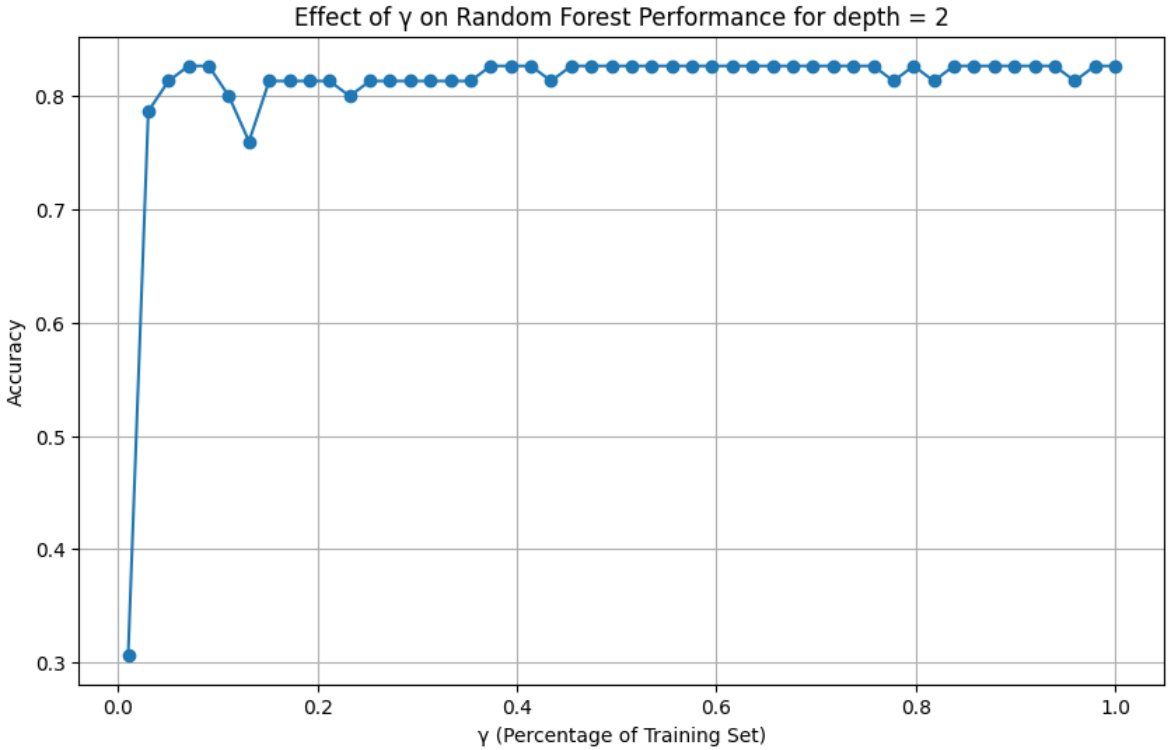
Παρατηρούμε πως τα όρια απόφασης έχουν γίνει πιο σύνθετα, και προσαρμόζονται καλύτερα στα δεδομένα, γεγονός που δικαιολογεί και την υψηλότερη ακρίβεια, ίσως όμως ταυτόχρονα να αυξάνεται ο κίνδυνος για overfitting.

Μέρος Γ - Ενότητα 2 - Υλοποίηση - Ερώτημα 3

Για να ελέγξουμε την επίδραση του γ στην απόδοση του αλγορίθμου, τον εκτελούμε για διάφορες τιμές του. Πιο συγκεκριμένα δοκιμάζουμε 50 τιμές ομοιόμορφα κατανεμημένες στο διάστημα [0.01,1].

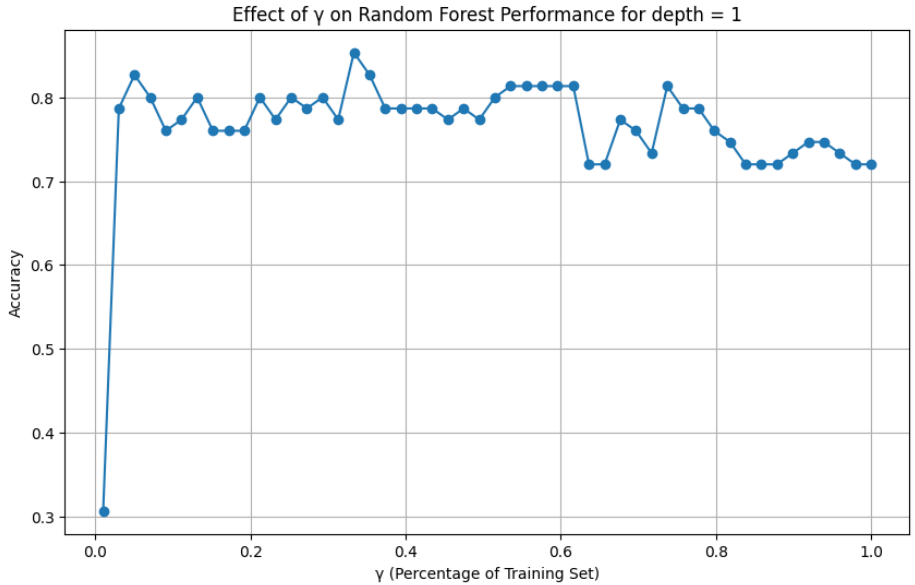
```
gamma_values = np.linspace(0.01, 1, 50)
```

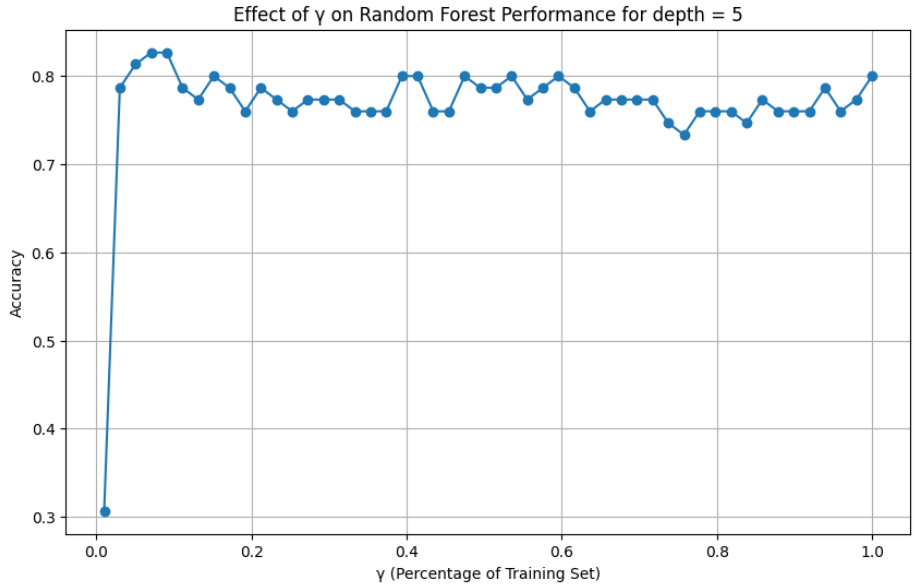
Για τα παραπάνω γ και για το βέλτιστο βάθος δέντρων παίρνουμε το διάγραμμα που δείχνει την επίδραση του γ στη μετρική του accuracy.

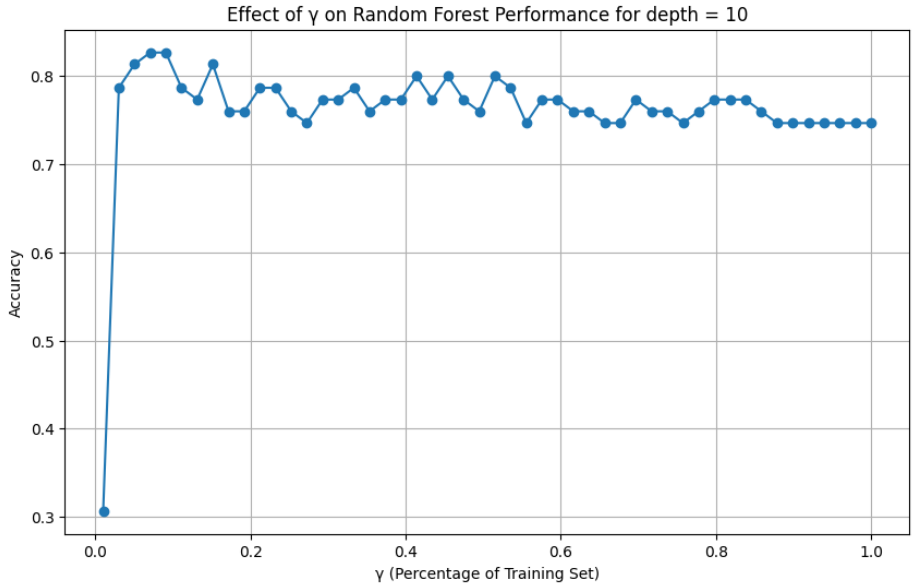


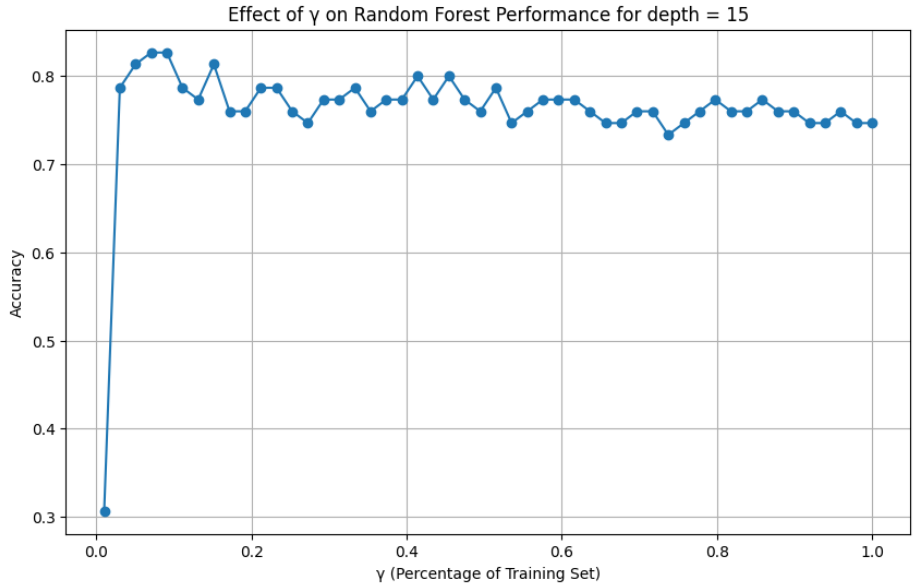
Παρατηρούμε πως το γ στη συγκεκριμένη περίπτωση δεν επηρεάζει την ακρίβεια πέρα από τιμές που βρίσκονται πολύ κοντά στο 0 (< 0.03).

Μέρος Γ - Ενότητα 2 - Υλοποίηση - Ερώτημα 3









Μέρος Γ - Ενότητα 2 - Ερώτημα 3 - Παρατηρήσεις

Με βάση τις δύο τελευταίες διαφάνειες μπορούμε να εξάγουμε ορισμένα συμπεράσματα για την επίδραση του γ:

- Παρατηρούμε πως σε κάθε περίπτωση κοντά στο 0 η ακρίβεια πέφτει σημαντικά.
- Για μέγιστο βάθος δέντρου 1 υπάρχει μια αρκετά σημαντική επίδραση του γ με την υψηλότερη ακρίβεια να επιτυγχάνεται γύρω στο $\gamma = 0.35$, με τιμή ακρίβειας κοντά στο 0.9.
- Σε κάθε περίπτωση η ακρίβεια δεν πέφτει κάτω από 0.7 (εξαιρουμένης της περιοχής κοντά στο 0 που αναφέρθηκε προηγουμένως).
- Για μεγαλύτερα βάθη συναντάμε την καλύτερη ακρίβεια σε πιο μικρά γ γύρω στο 0.1.

Συνοψίζοντας η υπερπαράμετρος του γ αν και δεν είναι μείζονος σημασίας για τον αλγόριθμο, σίγουρα επηρεάζει σε μικρό βαθμό την ακρίβεια του και απαιτεί μία σύντομη ανάλυση για την εύρεση της βέλτιστης τιμής του, ειδικά σε περιπτώσεις όπου το μέγιστο βάθος των δέντρων δεν είναι το βέλτιστο.

Μέρος Δ - Εισαγωγή

Στο μέρος Δ καλούμαστε να αναπτύξουμε αλγόριθμο ταξινόμησης με μέθοδο της επιλογής μας. Χρησιμοποιούμε το δοθέν datasetTV.csv ως training set, το οποίο περιέχει 8743 δείγματα με 224 χαρακτηριστικά ανά δείγμα μαζί με τις ετικέτες τους. Επιπλέον, χρησιμοποιούμε το δοθέν datasetTest.csv ως test set για να εφαρμόσουμε το μοντέλο μας και να παράξουμε το ζητούμενο αρχείο - διάνυσμα labels33 με τις προβλέψεις του μοντέλου μας.

Μέρος Δ - Θεωρητική Ανάλυση

Η προσέγγιση που επιλέγουμε είναι η εξής:

- Δοκιμάζουμε πολλούς και διαφορετικούς αλγορίθμους που προκύπτουν από διάφορες έτοιμες υλοποιήσεις των βιβλιοθηκών sklearn, xgboost και lightgbm. Δοκιμάζουμε τους:
  - Random Forest (RF)
  - K-Nearest Neighbors (KNN)
  - Support Vector Machine (SVM)
  - Multi-Layer Perceptron (MLP)
  - Adaptive Boosting (AdaBoost)
  - Extreme Gradient Boosting (XGBoost)
  - Light Gradient Boosting Machine (LightGBM)
- Χρησιμοποιούμε την GridSearchCV της sklearn για να καταλήξουμε στο βέλτιστο σετ υπερπαραμέτρων για κάθε μοντέλο που επιτυγχάνουν την καλύτερη ακρίβεια στο validation set. Να σημειώσουμε εδώ τα grids που δίνονται παρακάτω είναι ένα υποσύνολο όλων των δοκιμών που έγιναν, για λόγους συντομίας όμως αναφέρουμε τις πιο ουσιώδεις.

Μέρος Δ - Θεωρητική Ανάλυση

Κάποιες ακόμα επιλογές μας:

- Πιο συγκεκριμένα, όσο αφορά την GridSearchCV χρησιμοποιούμε cross validation με 3, 5 ή 10 folds και επιλέγουμε το training να γίνει στο 80% του datasetTV και το validation στο υπόλοιπο 20%. Οι επιλογές έγιναν με βάση όσα έχουν ειπωθεί στις διαλέξεις και στα εργαστήρια, καθώς και με βάση δική μας έρευνα.
- Επιλέγουμε να μην κανονικοποιήσουμε τα δεδομένα μας κατά την προεπεξεργασία γιατί παρατηρούμε πως η κανονικοποίηση είτε με StandardScaler είτε με MinMaxScaler μειώνει την ακρίβεια των προβλέψεων στο validation set. Επιπλέον, ορίζουμε τη συνάρτηση check_scaling_or_normalization που χρησιμοποιεί δείκτες για το range των features και της τυπικής απόκλισης σε ένα δείγμα για να αποφανθούμε αν κάποιο dataset χρήζει κανονικοποίησης.
- Χρησιμοποιήσαμε της μετρικές Silhouette Score, Davies-Bouldin Index της sklearn για να έχουμε μία διαισθητική εικόνα των ζητούμενων παραγόμενων labels αν και γνωρίζουμε πως χρησιμοποιούνται ορθότερα σε προβλήματα regression.

Μέρος Δ - Random Forest

Αλγόριθμος γνωστός από το μάθημα. Χρησιμοποιήθηκε η RandomForestClassifier της sklearn.ensemble με την τεχνική Bootstrap που αναλύθηκε και στο μέρος Γ της εργασίας, καθώς παρατηρήσαμε ότι η χρήση της έδινε καλύτερα αποτελέσματα.

Πλεονεκτήματα:

- Ανθεκτικότητα σε overfitting, λόγω του τρόπου με τον οποίο δημιουργεί και συνδυάζει πολλαπλά ανεξάρτητα decision tree (ensemble, random feature selection).
- Αρκετά καλή ακρίβεια σε προβλήματα ταξινόμησης και καταλληλότητα για δεδομένα με πολλά χαρακτηριστικά.

Μειονεκτήματα:

- Αργός χρόνος εκπαίδευσης και απαίτηση περισσότερων πόρων λόγω της δημιουργίας πολλών δέντρων.
- Υψηλό υπολογιστικός κόστος για τη διαδικασία του fine tuning και της εύρεσης των βέλτιστων υπερπαραμέτρων.

Μέρος Δ - Random Forest

Δοκιμάστηκαν οι εξής παράμετροι:

```
# Parameter grid for random forest
param_grid_rf = {
    'n_estimators': [100, 200, 250, 300, 350],
    'max_depth': [None, 1, 2, 5, 7, 10, 20, 25, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4, 10]
}
```

Με βέλτιστες επιλογές να είναι οι:

- n_estimators = 300
- max_depth = None (classifier default value)
- min_samples_leaf = 1 (classifier default value)
- min_samples_split = 2 (classifier default value)

Η ακρίβεια που επετεύχθη στο validation set ήταν: 0.819325328759291

Μέρος Δ - K-Nearest Neighbors

Αλγόριθμος γνωστός από το μάθημα. Χρησιμοποιήθηκε η KNeighborsClassifier της sklearn.neighbors.

Πλεονεκτήματα:

- Ευκολία στην εφαρμογή και κατανόηση αφού βασίζεται μόνο στις αποστάσεις μεταξύ των δειγμάτων.
- Μπορεί να χρησιμοποιηθεί πολύ αποτελεσματικά σε μικρά datasets και να διαχειριστεί μη γραμμικά δεδομένα.

Μειονεκτήματα:

- Ευαισθησία σε outliers και σε ενδεχόμενα irrelevant featues.
- Δεν κλιμακώνει καλά σε πολύ μεγάλα σύνολα δεδομένων, ειδικά όταν αυξάνεται ο αριθμός των χαρακτηριστικών ή των δειγμάτων.

Μέρος Δ - K-Nearest Neighbors

Δοκιμάστηκαν οι εξής παράμετροι:

```
# Parameter grid for KNN
param_grid_knn = {
    'n_neighbors': [5, 7, 9, 11, 13, 15],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan', 'minkowski'],
}
```

Ο αριθμός των γειτόνων επιλέγεται ως περιττός αριθμός για να μην υπάρχουν ισοπαλίες σε διαδικασίες classification.
Βέλτιστες επιλογές είναι οι:

- n_neighbors = 7
- weights = 'distance'
- metric = 'euclidean' (classifier default value)

Η ακρίβεια που επετεύχθη στο validation set ήταν: 0.846769582618639 2

Μέρος Δ - Support Vector Machine

Αλγόριθμος γνωστός από το μάθημα. Χρησιμοποιήθηκε η SVC (Support Vector Classifier) της sklearn.svm.

Πλεονεκτήματα:

- Υψηλή αποτελεσματικότητα σε δεδομένα υψηλή διάστασης, λόγω της χρήσης του kernel και της εκ κατασκευής μεγιστοποίησης του περιθωρίου μεταξύ των κλάσεων.
- Πολύ καλή ικανότητα γενίκευσης και δυνατότητα αποφυγής overfitting, ειδικά όταν το μέγεθος του dataset είναι περιορισμένο.

Μειονεκτήματα:

- Αυξημένη υπολογιστική πολυπλοκότητα ειδικά για μεγάλα datasets, και κατά τη διάρκεια αναζήτησης των βέλτιστων υπερπαραμέτρων.
- Υπάρχει μεγάλη ευαισθησία στην επιλογή των υπερπαραμέτρων, αν δεν γίνει σωστή επιλογή του kernel, του C και του gamma ο αλγόριθμος μπορεί να μην αποδώσει.

Μέρος Δ - Support Vector Machine

Δοκιμάστηκαν οι εξής παράμετροι:

```
# Parameter grid for SVM
param_grid_svm = {
    'C': [0.1, 1, 8, 9, 10, 11, 12, 15, 100],
    'gamma': [1, 0.1, 0.03, 0.025, 0.022, 0.02, 0.018,
        0.015, 0.01, 0.001],
    'kernel': ['linear', 'rbf']
}
```

Βέλτιστες επιλογές είναι οι:

- C = 9
- gamma = 0.025
- kernel = 'rbf' (classifier default value)

Η ακρίβεια που επετεύχθη στο validation set ήταν: 0.873642081189251

Μέρος Δ - Multi-Layer Perceptron

Αλγόριθμος ενός feedforward νευρωνικού δικτύου γνωστός από το μάθημα. Χρησιμοποιήθηκε η MLPClassifier της sklearn.neural_network.

Πλεονεκτήματα:

- Universal Approximator, όπως είδαμε και στο μάθημα μπορεί να προσεγγίσει αρκετά καλά οποιαδήποτε συνάρτηση με αρκετούς νευρώνες.
- Προσφέρει μία σχετική ευελιξία αφού υπάρχουν πολλές δυνατότητες επιλογής της αρχιτεκτονικής του, οι οποίες κλιμακώνουν καλά σε μεγαλύτερα datasets αν υπάρχουν πόροι.

Μειονεκτήματα:

- Απαιτείται πολύ μεγάλη υπολογιστική ισχύς ειδικά για μεγάλα datasets.
- Κίνδυνος overfitting αν δεν υπάρχει ικανός αριθμός από δεδομένα.

Μέρος Δ - Multi-Layer Perceptron

Δοκιμάστηκαν οι εξής παράμετροι:

```
# Parameter grid for MLP
param_grid_mlp =
{   'hidden_layer_sizes': [(10), (100), (50,50), (100,50),
    (100,100), (50,50,50), (400, 40), (400,38), (400,42)],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.001, 0.01],
    'learning_rate': ['constant', 'adaptive'],
    'learning_rate_init': [0.001, 0.01],        }
```

Βέλτιστες επιλογές είναι οι:

- hidden_layer_sizes = (400,40)
- activation = 'relu' (classifier default value)
- solver = 'adam' (classifier default value)
- alpha = 0.001 (classifier default value)
- learning_rate = 'constant' (classifier default value)
- learning_rate_init = 0.001 (classifier default value)

Η ακρίβεια που επετεύχθη στο validation set ήταν: 0.855917667238422

Μέρος Δ - Adaptive Boosting

Αλγόριθμος γνωστός από το μάθημα. Χρησιμοποιήθηκε η AdaBoostClassifier της sklearn.ensemble.

Πλεονεκτήματα:

- Δίνει μεγαλύτερη έμφαση στα δείγματα που ταξινομούνται λανθασμένα (Boosting), βελτιώνοντας τη συνολική απόδοση.
- Μπορεί να συνδυαστεί με διάφορους αλγορίθμους (π.χ., decision stumps, decision trees) και είναι σχετικά εύκολο να εφαρμοστεί, καθώς οι βασικοί αλγόριθμοι λειτουργούν αυτόνομα και εστιάζουν μόνο στις αδύναμες περιοχές.

Μειονεκτήματα:

- Ο διαδοχικός τρόπος με τον οποίο εκπαιδεύονται τα weak learners (με ενημέρωση των βαρών σε κάθε βήμα) μπορεί να είναι πιο αργός σε σχέση με άλλους αλγορίθμους, ειδικά σε μεγάλα datasets.
- Ενδέχεται να επηρεαστεί σε μεγάλο βαθμό από outliers, καθώς τους δίνει μεγαλύτερη σημασία στις επόμενες επαναλήψεις.

Μέρος Δ - Adaptive Boosting

Δοκιμάστηκαν οι εξής παράμετροι:

```
# Parameter grid for AdaBoost
param_grid_ab = {
    'n_estimators': [50, 100, 200, 300],
    'learning_rate': [0.01, 0.1, 1, 10]
}
```

Βέλτιστες επιλογές είναι οι:

- n_estimators = 300
- learning_rate = 0.1

Η ακρίβεια που επετεύχθη στο validation set ήταν: 0.6958261863922242

Μέρος Δ - Gradient Boosting

Διαισθητικά ένας Gradient Boosting αλγόριθμος, συνδυάζει δύο τεχνικές γνωστές από το μάθημα, αυτή του Boosting, δηλαδή τον συνδυασμό πολλών weak learners σε έναν πιο "ισχυρό" learner, στον οποίο σε κάθε επάναληψη διορθώνονται τα "λάθη" των προγόνων, με αυτή του Gradient Desecent. Ο αλγόριθμος συνοπτικά:

Input: training set $\{(x_i, y_i)\}_{i=1}^n$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
   1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

   2. Fit a base learner (or weak learner, e.g. tree) closed under scaling $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^n$.
   3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

   4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

Μέρος Δ - Extreme Gradient Boosting

Ο XGBoost αποτελεί μια βελτιωμένη υλοποίηση του Gradient Boosting, με τεχνικές κανονικοποίησης, παραλληλισμού, και βελτιστοποιήσεις για ταχύτητα και ακρίβεια, καθιστώντας τον ιδανικό για μεγάλα και σύνθετα datasets. Χρησιμοποιήθηκε η XGBClassifier της xgboost.

Πλεονεκτήματα:

- Υψηλή ακρίβεια γιατί συνδυάζει τεχνικές κανονικοποίησης, early stopping, και παραμετροποίηση, καθιστώντας τον ιδιαίτερα αποδοτικό για datasets με σύνθετα μοτίβα.
- Πλήρης υποστήριξη παραλληλισμού κατά την εκπαίδευση, επιταχύνοντας τη διαδικασία σε μεγάλα datasets, ενώ μπορεί να διαχείριστει με υψηλή ακρίβεια task παλινδρόμησης, ταξινόμησης και βαθμολόγησης.

Μειονεκτήματα:

- Απαίτηση για μεγάλη υπολογιστική ισχύς και μνήμη, ιδιαίτερα σε μεγάλα datasets.
- Πολύ μεγάλος αριθμός υπερπαραμέτρων και έτσι η διαδικασία εύρεσης των βέλτιστων γίνεται πολύ σύνθετη και χρονοβόρα.

Μέρος Δ - Extreme Gradient Boosting

Δοκιμάστηκαν οι εξής παράμετροι:

```
# Parameter grid for XGB
param_grid_xgb = {
    'n_estimators': [50, 100, 200, 300],
    'learning_rate': [0.01, 0.1, 0.2, 0.3],
    'max_depth': [3, 5, 7, 10],
    'subsample': [0.7, 0.8, 0.9, 1.0],
    'colsample_bytree': [0.7, 0.8, 0.9, 1.0]
}
```

Βέλτιστες επιλογές είναι οι:

- n_estimators = 300
- learning_rate = 0.1
- max_depth = 7
- subsample = 0.7
- colsample_bytree = 0.7

Η ακρίβεια που επετεύχθη στο validation set ήταν: 0.8502001143510578

Μέρος Δ - Light Gradient Boosting Machine

Ο LightGBM είναι ένας ελαφρύς και εξαιρετικά αποδοτικός αλγόριθμος Gradient Boosting που χρησιμοποιεί histogram-based προσέγγιση (διακριτοποίηση των συνεχών χαρακτηριστικών σε ομάδες (bins) πριν από την αναζήτηση των βέλτιστων splits) και τεχνικές όπως το leaf-wise growth, βελτιώνοντας την απόδοση σε μεγάλα σύνολα δεδομένων με πολλές διαστάσεις. Χρησιμοποιήθηκε η LGBMClassifier της lightgbm.

Πλεονεκτήματα:

- Η histogram-based προσέγγιση για την κατασκευή δέντρων απόφασης μειώνει σημαντικά τον χρόνο εκπαίδευσης σε σύγκριση με άλλους αλγορίθμους Gradient Boosting.
- Η ανάπτυξη δέντρων με βάση τα φύλλα (leaf-wise) ενδέχεται να οδηγήσει σε μεγαλύτερη ακρίβεια σε σύνθετα datasets, σε σχέση με άλλους αλγορίθμους.

Μειονεκτήματα:

- Η ανάπτυξη δέντρων με βάση τα φύλλα ίσως προκαλεί overfitting.
- Πολύ μεγάλος αριθμός υπερπαραμέτρων και έτσι η διαδικασία εύρεσης των βέλτιστων γίνεται πολύ σύνθετη και χρονοβόρα.

Μέρος Δ - Light Gradient Boosting Machine

Δοκιμάστηκαν οι εξής παράμετροι:

```
# Parameter grid for lightGBM
param_grid_lgbm = {
    'num_leaves': [31, 50, 70, 100, 150, 300],
    'max_depth': [-1, 10, 20],
    'learning_rate': [0.01, 0.05, 0.1, 0.15, 0.2],
    'n_estimators': [100, 150, 200, 250, 300, 500],
    'min_child_samples': [10, 20, 30],
    'subsample': [0.8, 1.0]    }
```

Βέλτιστες επιλογές είναι οι:

- num_leaves =31 (classifier default value)
- max_depth =-1 (classifier default value)
- learning_rate = 0.1 (classifier default value)
- n_estimators = 200
- min_child_samples = 20 (classifier default value)
- subsample = 0.8

Η ακρίβεια που επετεύχθη στο validation set ήταν: 0.8444825614636935

Μέρος Δ - Σύνοψη των αποτελεσμάτων

Παρουσιάζονται σε μορφή πίνακα η ακρίβεια στο validation set για όλα τα μοντέλα που δοκιμάστηκαν:

Αλγόριθμος	**Accuracy**
Random Forest (RF)	0.819325328759291
K-Nearest Neighbors (KNN)	0.846769582618639 2
Support Vector Machine (SVM)	**0.873642081189251**
Multi-Layer Perceptron (MLP)	0.855917667238422
Adaptive Boosting (AdaBoost)	0.695826186392242
Extreme Gradient Boosting (XGBoost)	0.850200114351057 8
Light Gradient Boosting Machine (LightGBM)	0.844482561463693 5

Μέρος Δ - Το τελικό Μοντέλο

Η λύση που προτείνουμε με βάση τις παραπάνω μετρικές είναι η εξής:

- **Αλγόριθμος:** Support Vector Machine (SVM)
- **Υπερπαράμετροι:**
  - C = 9
  - gamma = 0.025
  - kernel = 'rbf'
- **Accuracy στο validation set:** 0.873642081189251

Τρέχοντας τα 3 πρώτα κελιά του αρχείου Team33-D.ipynb παράγεται το αρχείο labels33.npy με τα ζητούμενα labels.

Πηγές

- https://en.wikipedia.org/wiki/Gradient_boosting
- https://web.archive.org/web/20191101082737/http://statweb.stanford.edu/~jhf/ftp/trebst.pdf
- https://lightgbm.readthedocs.io/
- https://xgboost.readthedocs.io/
- https://scikit-learn.org/