

# Εργασία - Αναγνώριση Προτύπων και Μηχανική Μάθηση

Όμαδα 22

Χρήστος - Αλέξανδρος Δαρδαμπούνης 10335

Χρήστος Κούνσολας 10345

# Μέρος Α

- Σκοπός του πρώτου μέρους είναι η εκτίμηση των παραμέτρων  $\hat{\theta}$  που μεγιστοποιούν την συνάρτηση πιθανοφάνειας η οποία είναι η εξής :

$$p(D_i|\theta) = p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N|\boldsymbol{\theta}) = \prod_{n=1}^N p(\mathbf{x}_n|\theta), i = 1,2$$

Όπου

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

η συνάρτηση πυκνότητας πιθανότητας του δείκτη και για τις δύο κλάσεις που μας δείχνει άμα κάποιος έχει στρες ( $\omega_1$ ) ή όχι ( $\omega_2$ )

# Μέρος Α

- Για τον υπολογισμό της συνάρτησης πιθανοφάνειας του  $\theta$  χρησιμοποιήσαμε τα δεδομένα της εκφώνησης

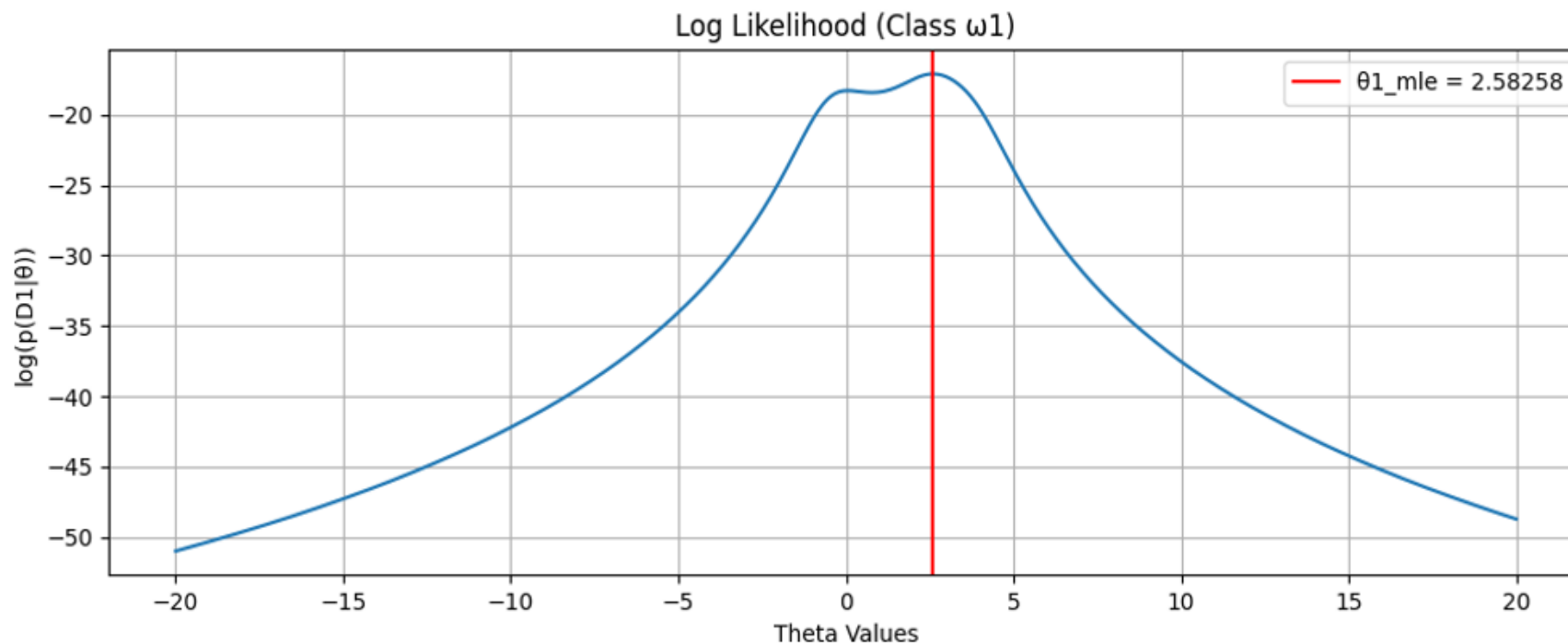
$D_1 = [2.8, -0.4, -0.8, 2.3, -0.3, 3.6, 4.1]$  για την κλάση  $\omega_1$  και το σετ  $D_2 = [-4.5, -3.4, -3.1, -3.0, -2.3]$  για την κλάση  $\omega_2$

Επειδή ο λογάριθμος είναι μια γνησίως αύξουσα συνάρτηση η μεγιστοποίηση της συνάρτησης πιθανοφάνειας μεγιστοποιεί επίσης την συνάρτηση log-likelihood. Συνεπώς επιλέγουμε να μεγιστοποιήσουμε τις  $\log(p(D_1|\theta))$  και  $\log(p(D_2|\theta))$

# Μέρος Α

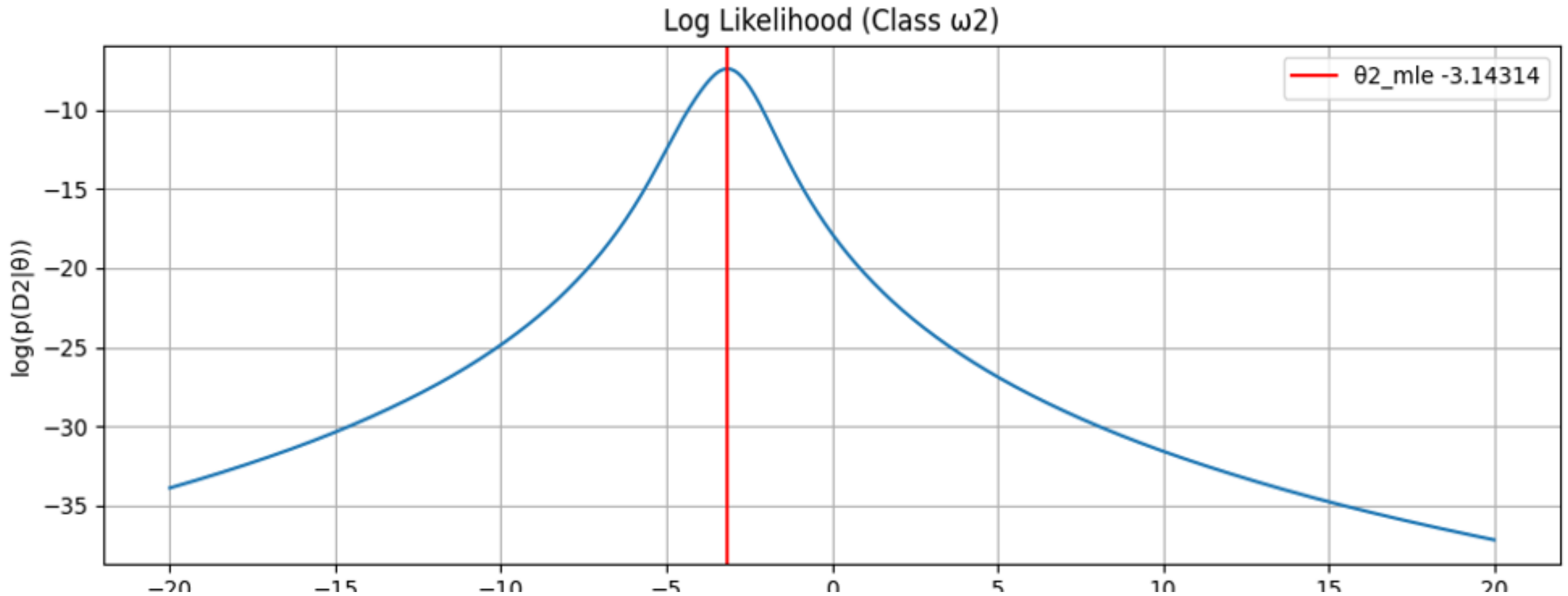
- Για να βρούμε που μεγιστοποιούνται αυτές οι συναρτήσεις δίνουμε ένα εύρος πιθανών τιμών του  $\theta$  και στην συνέχεια για τον προσδιορισμό του εκτιμητή του  $\theta$  βρίσκουμε εκείνο το  $\theta$  που μεγιστοποιεί την συνάρτησή μας. Δηλαδή  $\hat{\theta}_{MLE} = \underset{\theta}{\operatorname{argmax}}\{\log(p(D|\theta))\}$
- Παρακάτω φαίνονται οι αντίστοιχες γραφικές παραστάσεις όπου βλέπουμε το  $\theta$  που τις μεγιστοποιεί.

# Μέρος Α



- Στο παραπάνω σχήμα φαίνεται η συνάρτηση πιθανοφάνειας για την κλάση  $\omega_1$  καθώς επίσης και η τιμή  $\hat{\theta}_{MLE}$  που την μεγιστοποιεί.

# Μέρος Α



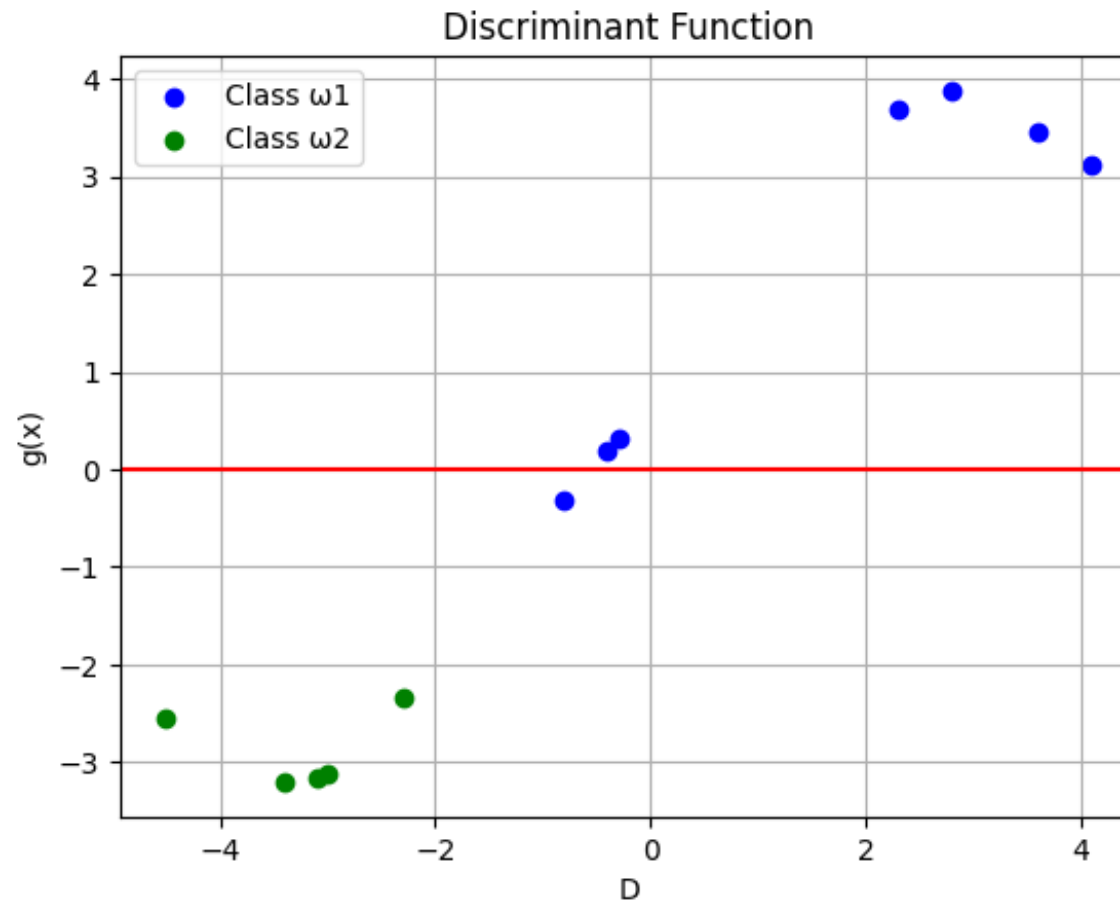
- Στο παραπάνω σχήμα φαίνεται η συνάρτηση πιθανοφάνειας για την κλάση  $\omega_2$  καθώς επίσης και η τιμή  $\hat{\theta}_{MLE}$  που την μεγιστοποιεί.

# Μέρος Α

- Στην συνέχεια χρησιμοποιώντας την συνάρτηση διάκρισης  $g(x) = \log(P(x|\hat{\theta}_1)) - \log(P(x|\hat{\theta}_2)) + \log(P(\omega_1)) - \log(P(\omega_2))$  πραγματοποιούμε ταξινόμηση των δύο συνόλων τιμών
- Αυτό που παρατηρούμε είναι πως για τα δεδομένα της πρώτης κλάσης  $D_1$  οι τιμές της συνάρτησης είναι κυρίως θετικές ενώ για δεδομένα της κλάσης  $D_2$  οι τιμές της συνάρτησης είναι κυρίως αρνητικές

# Μέρος Α

Παρακάτω φαίνεται και η γραφική παράσταση της συνάρτησης διάκρισης.





# Μέρος Α

Όπως μπορούμε να δούμε αυτή η συνάρτηση διάκρισης  $g(\mathbf{x})$  μας βοηθάει να διακρίνουμε τις δύο κλάσεις μεταξύ τους. Πιο συγκεκριμένα ο κανόνας απόφασης είναι ο εξής :

$$\begin{cases} x \text{ to } \omega_1, g(x) > 0 \\ x \text{ to } \omega_2, g(x) < 0 \end{cases}$$

Ο παρακάτω κανόνας απόφασης προκύπτει από το θεώρημα Bayes:  $P(\omega_i|x) = \frac{P(x|\hat{\theta}_i)P(\omega_i)}{P(x)}$  και στην συνέχεια λογαριθμίζοντας και παίρνοντας την διαφορά τους προκύπτει  $g(x) = \log(P(\omega_1|x)) - \log(P(\omega_2|x))$  που είναι η συνάρτηση διάκρισης που μας δίνεται στην εκφώνηση.

# Μέρος Α

Με βάση τον κανόνα ταξινόμησης που περιγράψαμε προηγουμένως και με βάση το γράφημα βλέπουμε πως η μέθοδος μέγιστης πιθανοφάνειας δεν αποδίδει τόσο καλά καθώς δεν καταφέρνει να κατηγοριοποιήσει σωστά όλα τα δείγματα που δίνονται. Αυτό φαίνεται καθώς ένα δείγμα της κλάσης  $\omega_1$  κατηγοριοποιείται λανθασμένα στην κλάση  $\omega_2$ .

# Μέρος Β

- Σε αυτό το μέρος σκοπός μας είναι να εκτιμήσουμε πάλι την παράμετρο  $\theta$  χρησιμοποιώντας όμως αυτή την φορά την μέθοδο εκτίμησης κατά Bayes.
- Σε αυτό το σημείο λαμβάνουμε υπόψη μας και την όποια προηγούμενη πληροφορία έχουμε για την άγνωστη παράμετρο  $\theta$  η οποία στην συγκεκριμένη περίπτωση μπορεί να μοντελοποιηθεί από την συνάρτηση :

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

# Μέρος Β

- Σκοπός μας είναι να βρούμε την *a – priori* συνάρτηση πυκνότητας πιθανότητας

$$p(\theta|D_i) = \frac{p(D_i|\theta)p(\theta)}{\int p(D_i|\theta)p(\theta)d\theta}, i = 1,2 \text{ όπου}$$

$$p(D_i|\theta) = p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N|\theta) = \prod_{n=1}^N p(\mathbf{x}_n|\theta), i = 1,2$$

όπως και προηγουμένως. Επιπλέον για να μπορέσουμε να βρούμε την πιθανότητα το δείγμα να ανήκει στην κλάση θα πρέπει να προσδιορίσουμε και την

$$p(\mathbf{x}|D_i) = \int p(\mathbf{x}|\theta)p(\theta|D_i)d\theta, i = 1,2$$

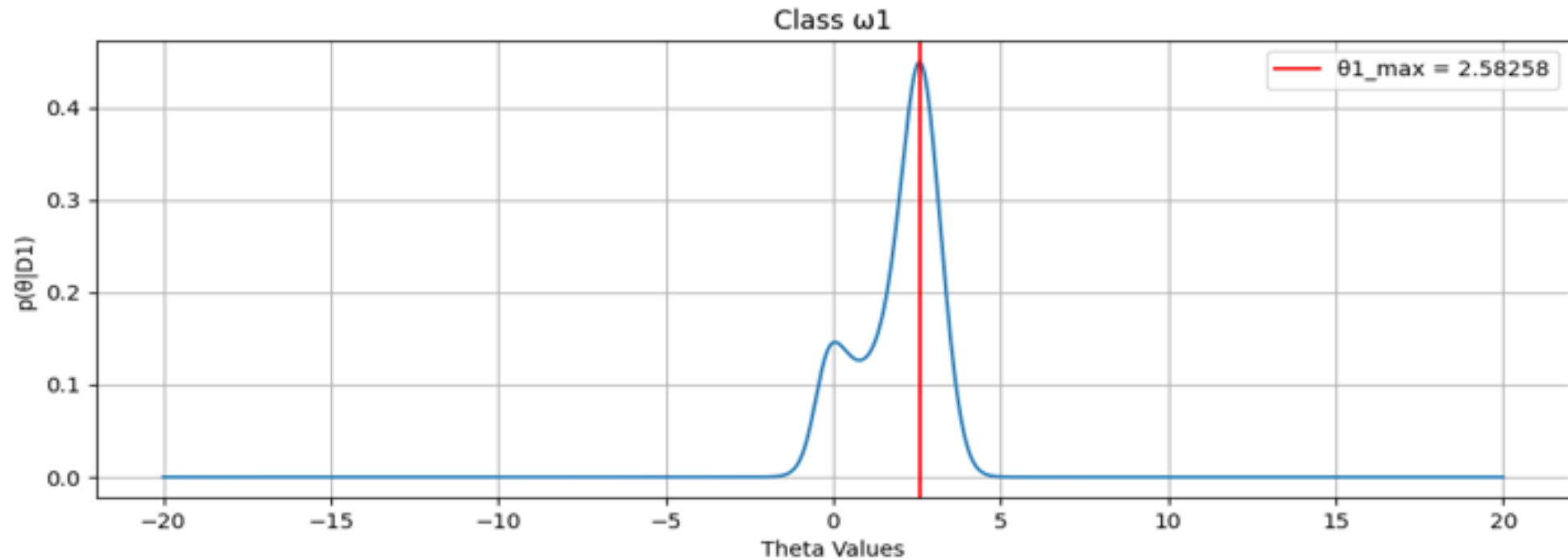
# Μέρος Β

- Έχοντας όλα αυτά μπορούμε να βρούμε την:

$$P(\omega_i|\mathbf{x}, D) = \frac{p(\mathbf{x}|D)P(\omega_i)}{\sum_{j=1}^c p(\mathbf{x}|D)P(\omega_j)}$$

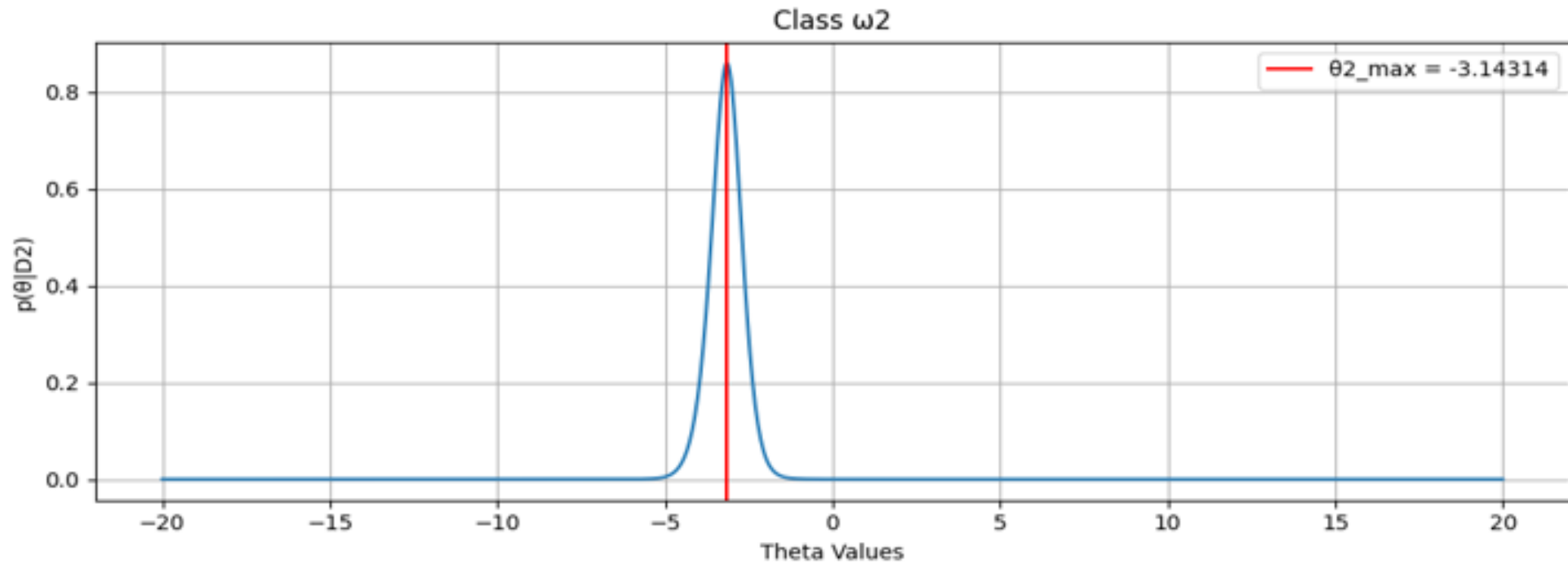
Δεδομένου όλων αυτών μπορούμε τώρα να εκτιμήσουμε την παράμετρο  $\theta$  και να απεικονίσουμε τις εκ των υστέρων πυκνότητες πιθανότητας  $p(\theta|D_1)$  και  $p(\theta|D_2)$ .

# Μέρος Β



- Στο παραπάνω σχήμα φαίνεται η *a-posterior* για την κλάση  $\omega_1$  καθώς επίσης και η τιμή  $\hat{\theta}$  που την μεγιστοποιεί.

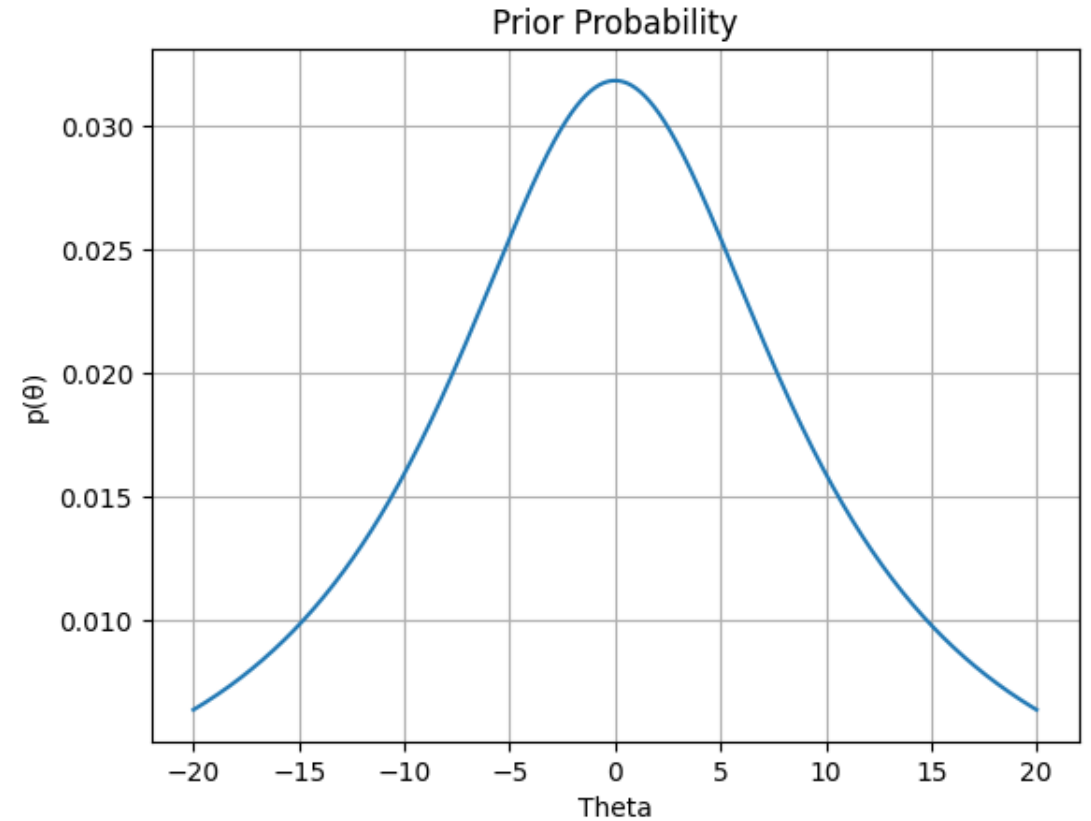
# Μέρος Β



- Στο παραπάνω σχήμα φαίνεται η *a-posterior* για την κλάση  $\omega_2$  καθώς επίσης και η τιμή  $\hat{\theta}$  που την μεγιστοποιεί.

# Μέρος Β

- Τέλος, σχηματίζουμε και την γραφική παράσταση της *prior* από την οποία μπορούμε να παρατηρήσουμε πως οι τιμές των  $\theta$  οι οποίες μεγιστοποιούν τις συναρτήσεις  $p(\theta|D_1)$  και  $p(\theta|D_2)$  είναι τιμές οι οποίες μεγιστοποιούν και την *prior* και άρα έχουν πολύ μεγάλη πιθανότητα.



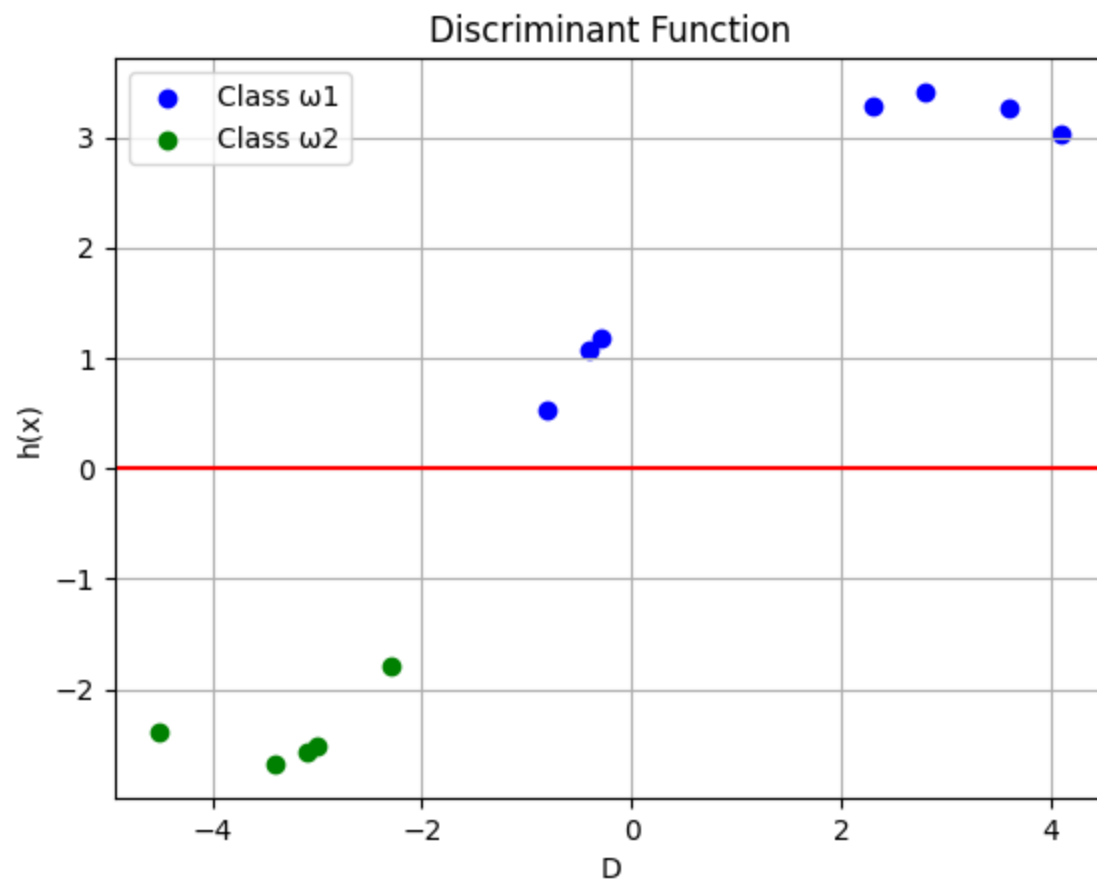


# Μέρος Β

- Στην συνέχεια χρησιμοποιώντας την συνάρτηση διάκρισης  $\mathbf{h}(x) = \mathbf{log}\left(P(x|\hat{\theta}_1)\right) - \mathbf{log}\left(P(x|\hat{\theta}_2)\right) + \mathbf{log}(P(\omega_1)) - \mathbf{log}(P(\omega_2))$  πραγματοποιούμε ταξινόμηση των δύο συνόλων τιμών
- Αυτό που παρατηρούμε είναι πως για τα δεδομένα της πρώτης κλάσης  $D_1$  οι τιμές της συνάρτησης είναι θετικές ενώ για δεδομένα της κλάσης  $D_2$  οι τιμές της συνάρτησης είναι αρνητικές

# Μέρος Β

- Παρακάτω φαίνεται και η γραφική παράσταση της συνάρτησης διάκρισης.



# Μέρος Β

- Σε σχέση με προηγουμένως παρατηρούμε πως η μέθοδος Bayes έχει κατορθώσει να κατηγοριοποιήσει καλύτερα τα δείγματα στις σωστές κλάσεις σε αντίθεση με την μέθοδο μέγιστης πιθανοφάνειας η οποία είχε κατηγοριοποιήσει λανθασμένα ένα δείγμα της κλάσης  $\omega_1$  στην κλάση  $\omega_2$
- Συνεπώς, μπορούμε να πούμε πως η μέθοδος Bayes είναι σαφώς καλύτερη από ότι η μέθοδος μέγιστης πιθανοφάνειας καθότι λαμβάνει υπόψη της την προηγούμενη γνώση για την παράμετρο  $\theta$ .

# Μέρος Β

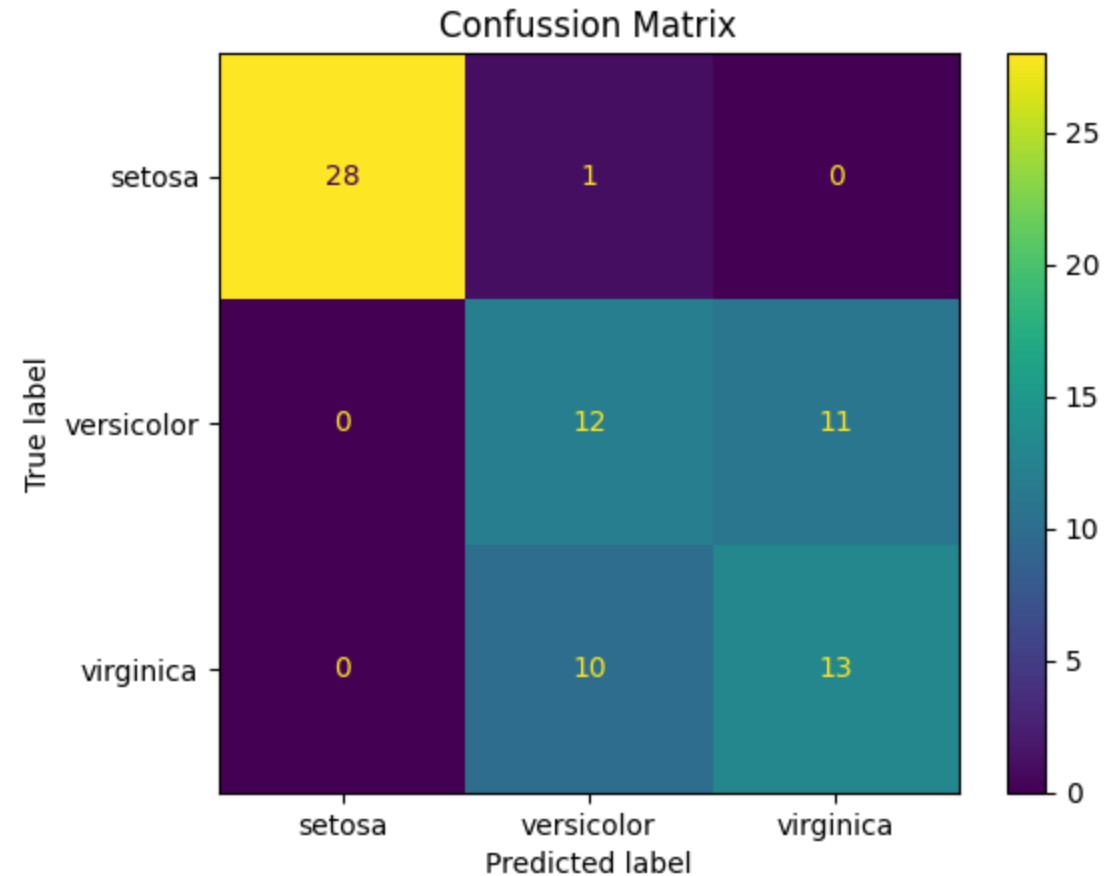
- Οι διαφορές που υπάρχουν με το μοντέλο που υλοποιήσαμε στο πρώτο μέρος θεωρούμε πως οφείλονται κυρίως στο γεγονός πως στην Bayesian προσέγγιση λαμβάνουμε υπόψη και την οποιαδήποτε προηγούμενη πληροφορία για το  $\theta$  το οποίο μπορεί να μας οδηγήσει σε πολύ καλύτερες λύσεις

# Μέρος Γ

- Σε αυτό το μέρος σκοπός μας είναι αρχικά να βρούμε το ποσοστό σωστής ταξινόμησης για ένα Δέντρο Απόφασης και να αναζητήσουμε το βέλτιστο βάθος δέντρου που μας δίνει το καλύτερο ποσοστό.
- Πρέπει να αναφέρουμε ότι από όλο το σύνολο (iris dataset) απομονώσαμε μόνο τα δύο πρώτα χαρακτηριστικά και χωρίσαμε το σύνολο σε 50% δείγματα εκπαίδευσης και 50% δείγματα ελέγχου.

# Μέρος Γ

- Χρησιμοποιώντας ένα Δέντρο Απόφασης με τις προκαθορισμένες παραμέτρους, επιτυγχάνουμε 70.6667% ποσοστό ακρίβειας. Παρακάτω φαίνεται επίσης και το confusion matrix που μας δείχνει την κατανομή των δειγμάτων στις κλάσεις.

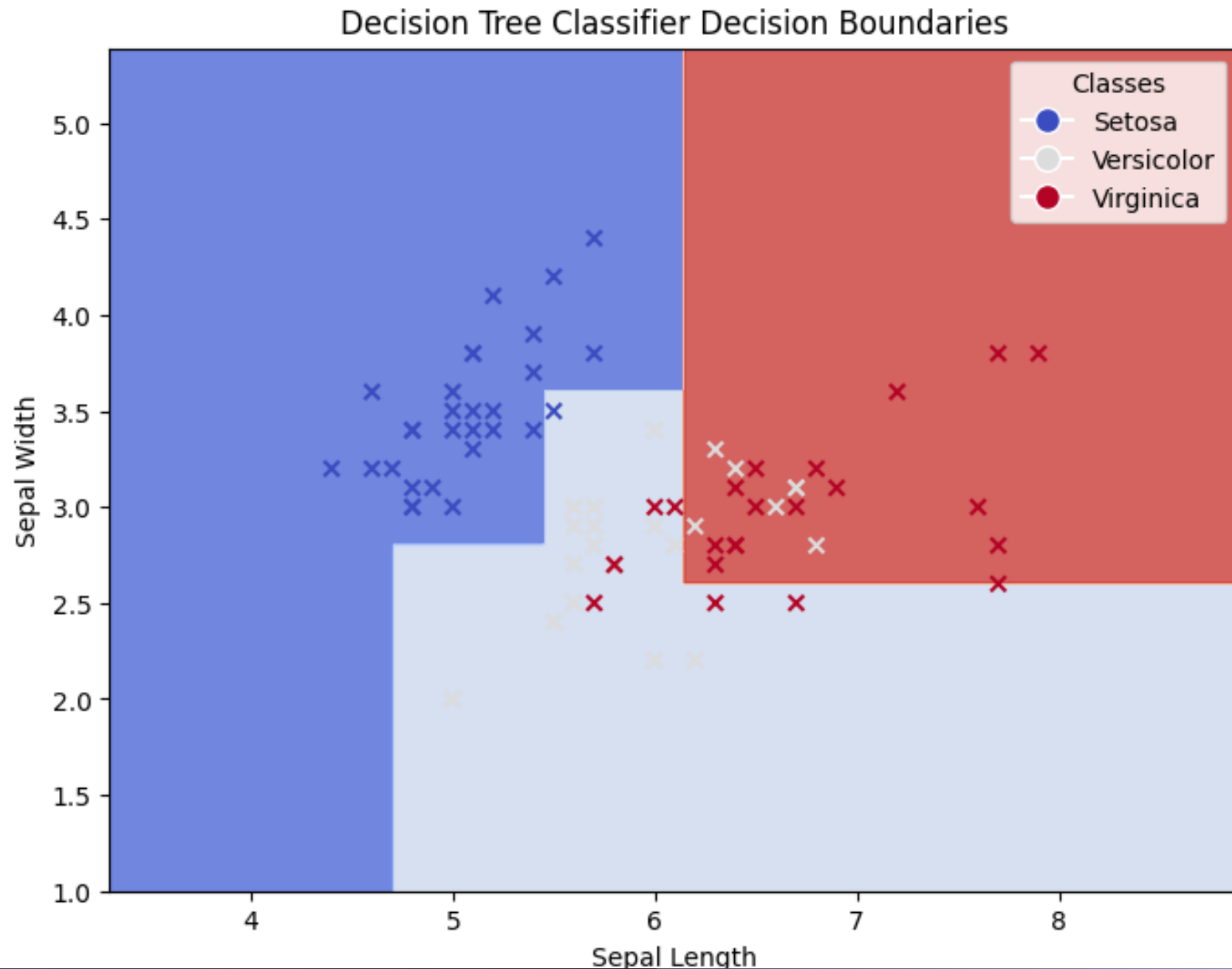


# Μέρος Γ

- Για την εύρεση του βέλτιστου βάθους δέντρου, το οποίο μας δίνει το καλύτερο ποσοστό εξετάζουμε διάφορα βάθη και με την βοήθεια της συνάρτησης `GridSearchCV()`, παίρνουμε το βέλτιστο βάθος δέντρου, το οποίο προκύπτει ότι είναι το 3.
- Με το Δεντρο Απόφασης το οποίο έχει το βέλτιστο βάθος επιτυγχάνουμε ακρίβεια 78.6667%

# Μέρος Γ

- Παραπάνω φαίνονται τα όρια απόφασης που έχουν προκύψει από το δέντρο απόφασης που δίνει το καλύτερο ποσοστό.





# Μέρος Γ

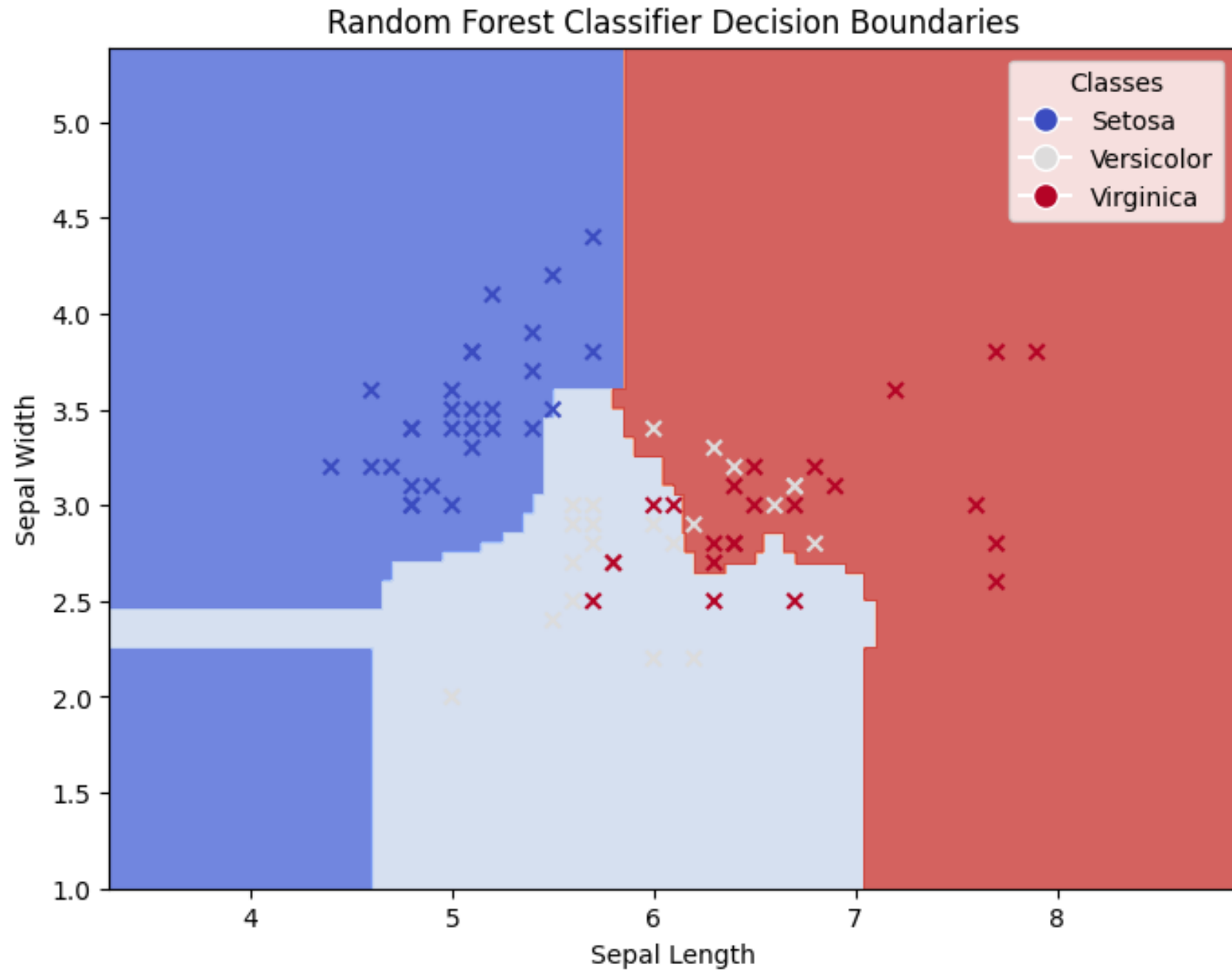
- Στο 2ο μέρος της 1ης ενότητας σκοπός μας είναι να αποκονήσουμε τα όρια απόφασης του ταξινομητή για το καλύτερο αποτέλεσμα.
- Από το γράφημα βλέπουμε ότι για την κλάση *Setosa*, υπάρχει μεγάλη ακρίβεια στην ταξινόμηση, καθώς μόνο 1 δείγμα δεν έχει ταξινομηθεί σωστά.
- Αντιθέτως για τα δείγματα της κλάσης *Versicolor* και για την *Virginica* έχουμε πολλές σωστές αλλά και αρκετές λάθος ταξινομήσεις.

# Μέρος Γ

- Για το 2ο μέρος μας ζητείται να δημιουργήσουμε ένα RandomForest ταξινομητή 100 δέντρων με την τεχνική Bootstrap. Χρησιμοποιούμε για κάθε εκτιμητή ένα διαφορετικό 50% του δείγματος εκπαίδευσης και το ίδιο σύνολο ελέγχου.
- Επιτυγχάνουμε 80% ακρίβεια και λειτουργώντας όπως παραπάνω βρίσκουμε ότι το καλύτερο βάθος είναι 3, το οποίο μας δίνει 80% ακρίβεια.

# Μέρος Γ

- Παραπάνω φαίνονται τα όρια απόφασης που έχουν προκύψει από το Random Forest που δίνει το καλύτερο ποσοστό.

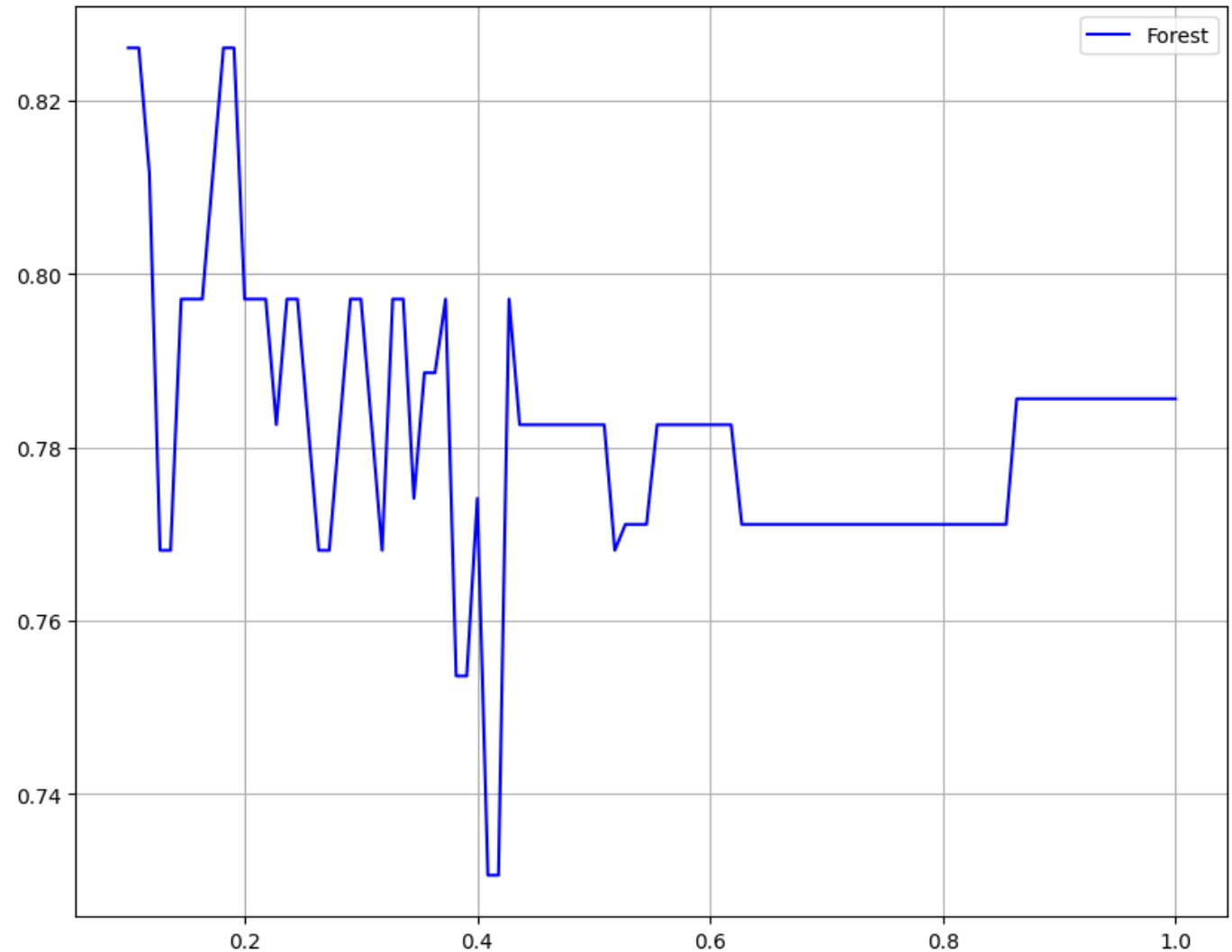


# Μέρος Γ

- Σε σχέση με τον απλό ταξινομητή παρατηρούμε πως τα όρια απόφασης είναι πιο λεία καθώς προκύπτουν από τον μέσο όρο όλων των δέντρων που χρησιμοποιεί ο Random Forest classifier.
- Επίσης, βλέπουμε πως έχει επιτύχει καλύτερη ταξινόμηση των δεδομένων στις κατάλληλες κλάσεις και έχει γενικεύσει καλύτερα σε σχέση με το απλό δέντρο απόφασης οδηγώντας έτσι σε μεγαλύτερο ποσοστό ακρίβειας.

# Μέρος Γ

- Παραπάνω φαίνεται το γράφημα που δείχνει την μεταβολή του ποσοστού ακρίβειας συναρτήση του ποσοστού των δειγμάτων που χρησιμοποιούνται για την bootstrap δειγματοληψία.



# Μέρος Γ

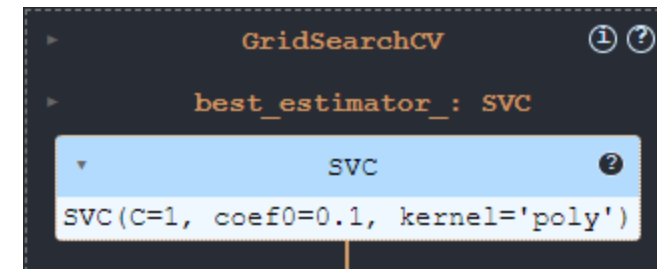
- Από το παραπάνω γράφημα μπορούμε να πούμε ότι μικρό  $\gamma$  προκαλεί μεγάλες διακυμάνσεις, καθώς ο πολύ μικρός αριθμός των δειγμάτων δεν παρέχει αρκετή ποικιλία για την σωστή εκπαίδευση των δέντρων.
- Επιπλέον παρατηρούμε ότι υπάρχει μια σταθεροποίηση στην ακρίβεια για μεγαλύτερες τιμές του  $\gamma$ , καθώς χρησιμοποιείται ένα επαρκές ποσοστό δειγμάτων.
- Τέλος για πολύ υψηλές τιμές, παρατηρείται μια μικρή βελτίωση στην ακρίβεια, πιθανότατα επειδή οι περισσότερες πληροφορίες από το dataset ενσωματώνονται στον ταξινομητή

# Μέρος Δ

- Σε αυτό το μέρος σκοπός μας είναι να αναπτύξουμε έναν αλγόριθμο ταξινόμησης για τα δεδομένα που μας δίνονται. Δοκιμάσαμε διάφορες αλγορίθμους ταξινόμησης στην προσπάθεια μας να επιτύχουμε την καλύτερη δυνατή ακρίβεια.
- Αρχικά χωρίσαμε το σύνολο εκπαίδευσης σε 80% για εκπαίδευση και 20 % για αξιολόγηση των αλγορίθμων μας.

# Μέρος Δ

- Το 1ο μοντέλο που υλοποιήσαμε ήταν ένα Support Vector Machine (SVM).
- Για την επιλογή των κατάλληλων υπερπαραμέτρων πραγματοποιήσαμε ένα GridSearchCV, από το οποίο προέκυψε ότι το καλύτερο SVM ήταν το εξής:



```
GridSearchCV
best_estimator_: SVC
SVC(C=1, coef0=0.1, kernel='poly')
```

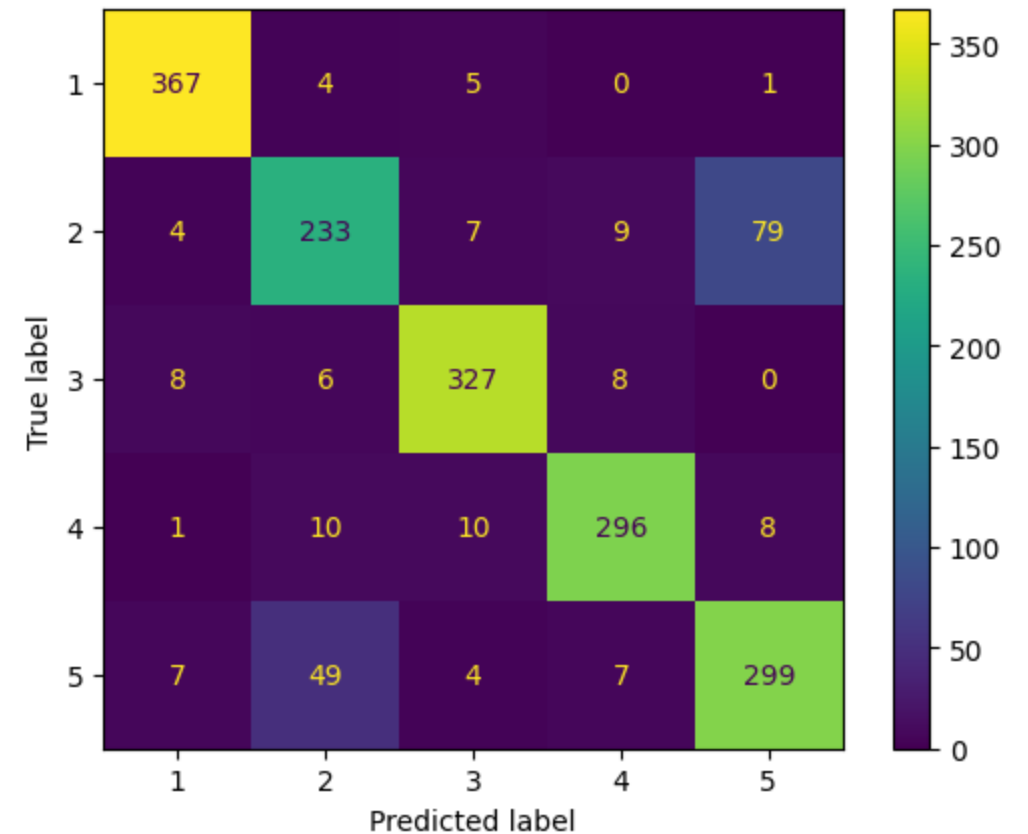
The screenshot shows a Jupyter Notebook interface. It displays the output of a GridSearchCV object, specifically the 'best\_estimator\_' attribute, which is an SVC (Support Vector Classification) model. The parameters of the SVC are shown as C=1, coef0=0.1, and kernel='poly'.

- Με το συγκεκριμένο SVM επιτύχαμε 87.0212% ακρίβεια



# Μέρος Δ

- Χρησιμοποιώντας ένα SVM με τις συγκεκριμένες παραμέτρους, επιτυγχάνουμε 87.02012% ποσοστό ακρίβειας. Παρακάτω φαίνεται και το confusion matrix που μας δείχνει την κατανομή των δειγμάτων στις κλάσεις. Παρατηρούμε ότι ο ταξινομητής είναι αρκετά καλός, με εξαίρεση τις κλάσεις 2 και 5, στις οποίες εμφανίζονται οι περισσότερες λανθασμένες ταξινομήσεις.

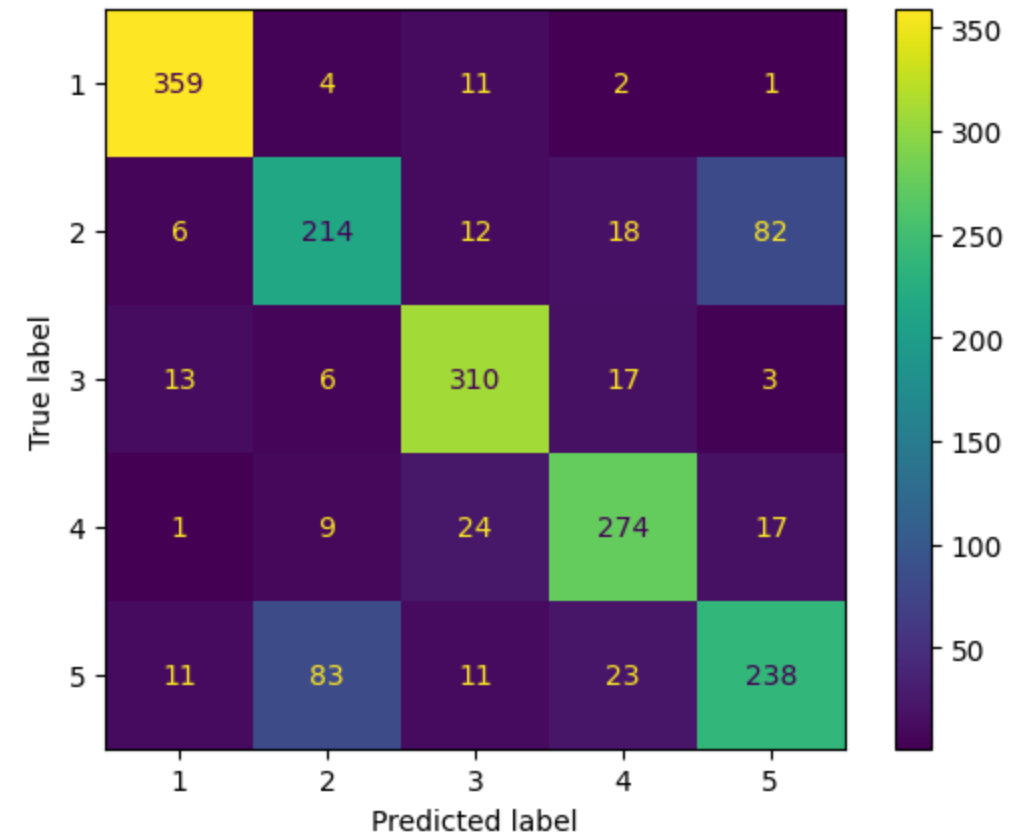


# Μέρος Δ

- Το 2ο μοντέλο που υλοποιήσαμε ήταν ένας Bagging Classifier που είχε 1000 Δέντρα Απόφασης για εκτιμητές και πραγματοποιούσε δειγματοληψία με επανάθεση (bootstrap) στα δεδομένα εκπαίδευσης.
- Τα Δέντρα Απόφασης που χρησιμοποιήθηκαν είχαν τις προκαθορισμένες παραμέτρους

# Μέρος Δ

- Με το συγκεκριμένο μοντέλο επιτυγχάνουμε 79.7599% ποσοστό ακρίβειας. Παρακάτω φαίνεται και το confusion matrix που μας δείχνει την κατανομή των δειγμάτων στις κλάσεις, όπου πάλι παρατηρείται πρόβλημα στις κλάσεις 2 και 5.

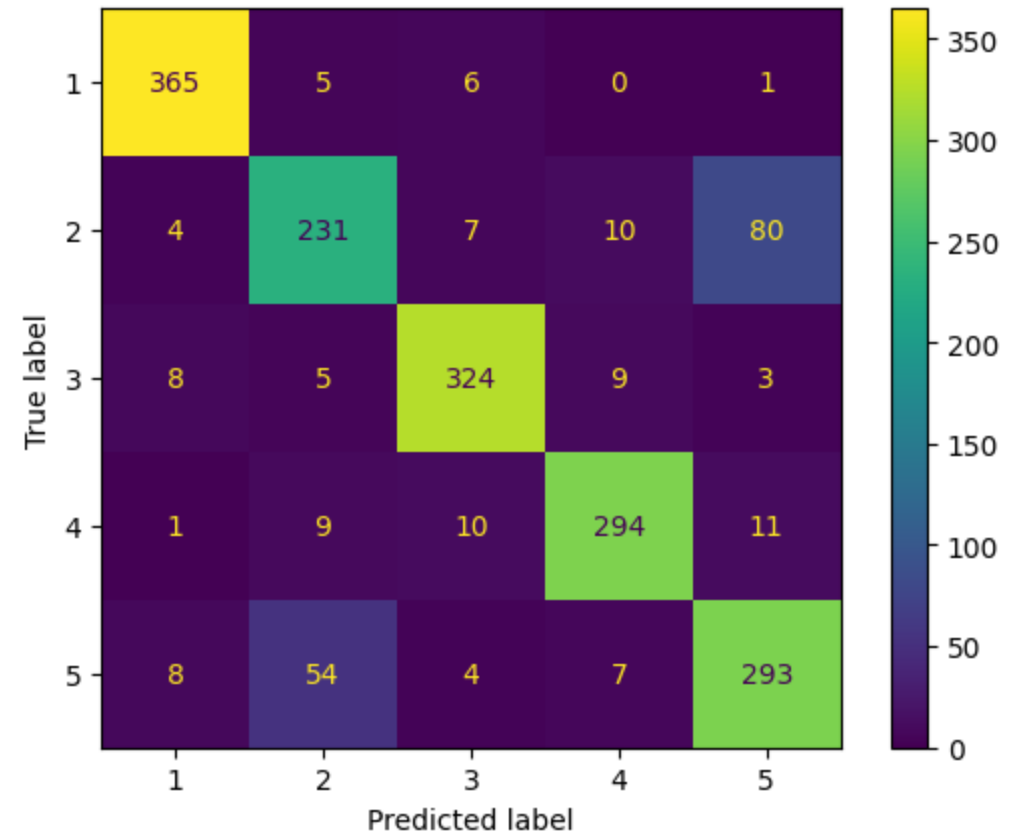


# Μέρος Δ

- Για το 3ο μοντέλο συνεχίζουμε στην λογική του προηγούμενου χρησιμοποιώντας πάλι έναν Bagging Classifier εφαρμόζοντας δειγματοληψία με επανάθεση (bootstrap) στα δείγματά μας, ο οποίος έχει 300 SVM για εκτιμητές με τις καλύτερες παραμέτρους που είχαμε βρει για το 1ο μοντελό. Ωστόσο το αποτέλεσμα που παίρνουμε δεν είναι τόσο καλό διότι

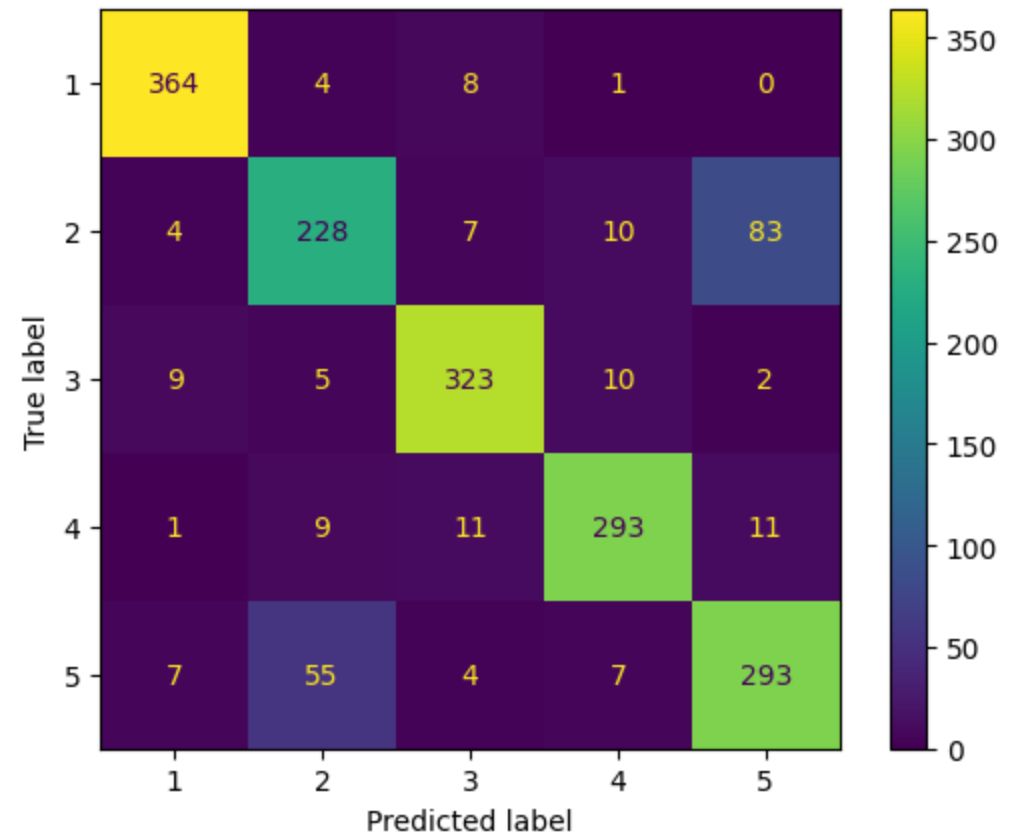
# Μέρος Δ

- Επιτυγχάνουμε 86.1635% ποσοστό ακρίβειας. Παρακάτω παρουσιάζουμε το Confusion Matrix, το οποίο όπως βλέπουμε είναι σαφώς καλύτερο απο το εκείνο του προηγούμενου μοντέλου, διατηρώντας όμως το πρόβλημα στην ταξινόμηση των κλάσεων 2 και 5.



# Μέρος Δ

- Βλέποντας ότι επιτυγχάνουμε αρκετά υψηλή ακρίβεια με το προηγούμενο μοντέλο, διατηρώντας για εκτιμητή το καλύτερο SVM που έχουμε βρει, πραγματοποιούμε ένα GridSearchCV() στο Bagging μοντέλο που προσδιορίσαμε. Ωστόσο, τα αποτελέσματα που παίρνουμε δεν είναι τόσο καλά όπως και προηγουμένως γεγονός που ίσως να οφείλεται στο ότι δεν βάλαμε αρκετά δείγματα Bootstrap.



# Μέρος Δ

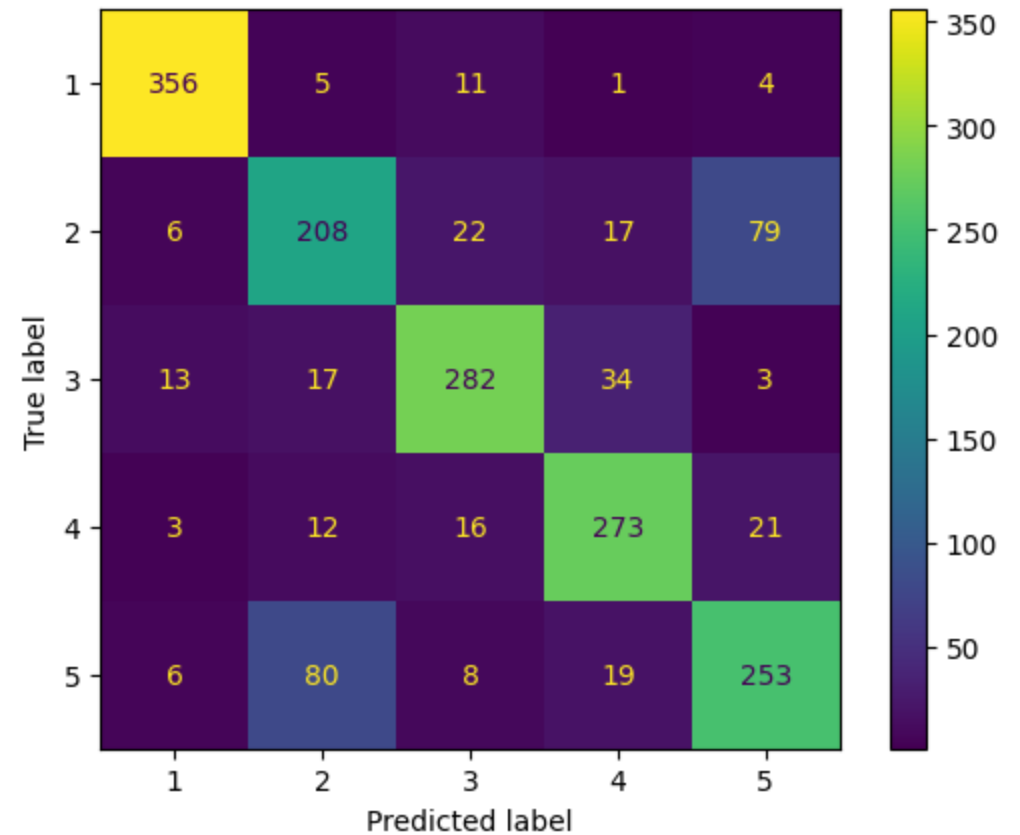
- Στη συνέχεια δοκιμάζουμε ένα Logistic Regression μοντέλο, αρχικά με τις προκαθορισμένες παραμέτρους, έπειτα εφαρμόζοντας δειγματοληψία με επανάθεση στα δειγματα και τέλος πραγματοποιώντας ένα GridSearchCV για να βρούμε τις καλύτερες παραμέτρους.
- Απο τις 3 παραπάνω παραλλαγές, εκείνη που μας έδωσε την μεγαλύτερη ακρίβεια ήταν η τελευταία.

# Μέρος Δ

- Επιτυγχάνουμε 78.4448% ποσοστό ακρίβειας και παρουσιάζουμε το confusion matrix για το συγκεκριμένο μοντέλο με τους παρακάτω παραμέτρους:

LogisticRegression

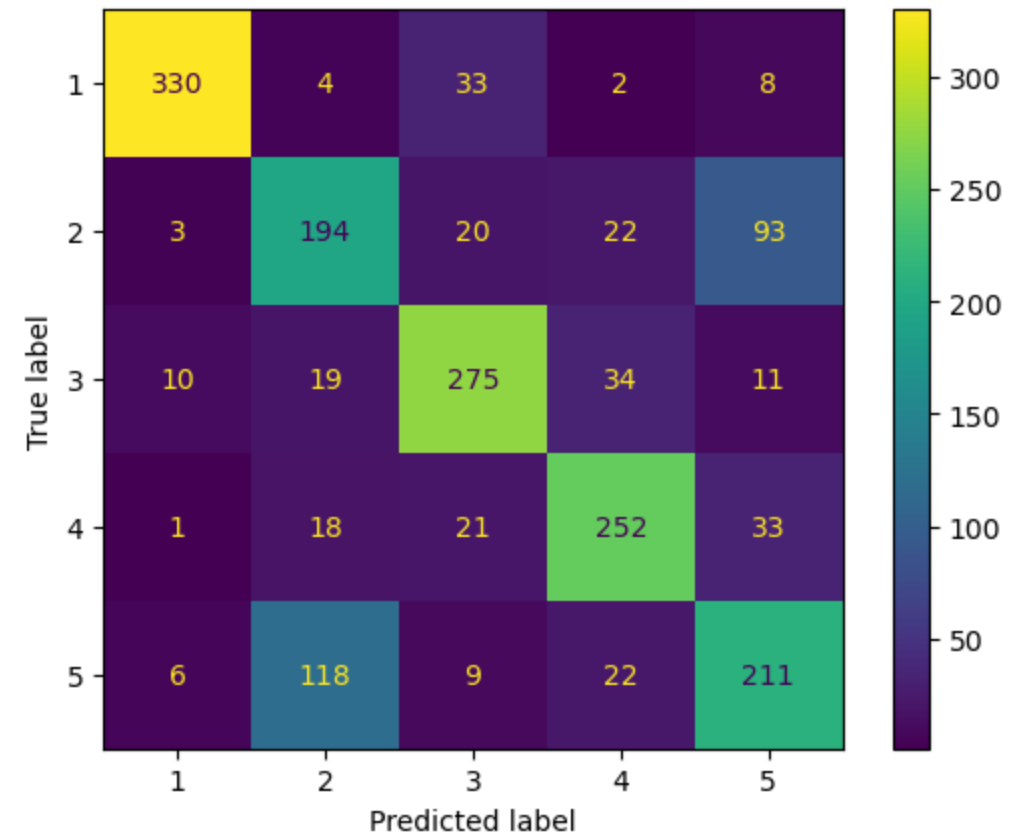
```
LogisticRegression(C=0.01, max_iter=200, random_state=42, solver='saga')
```





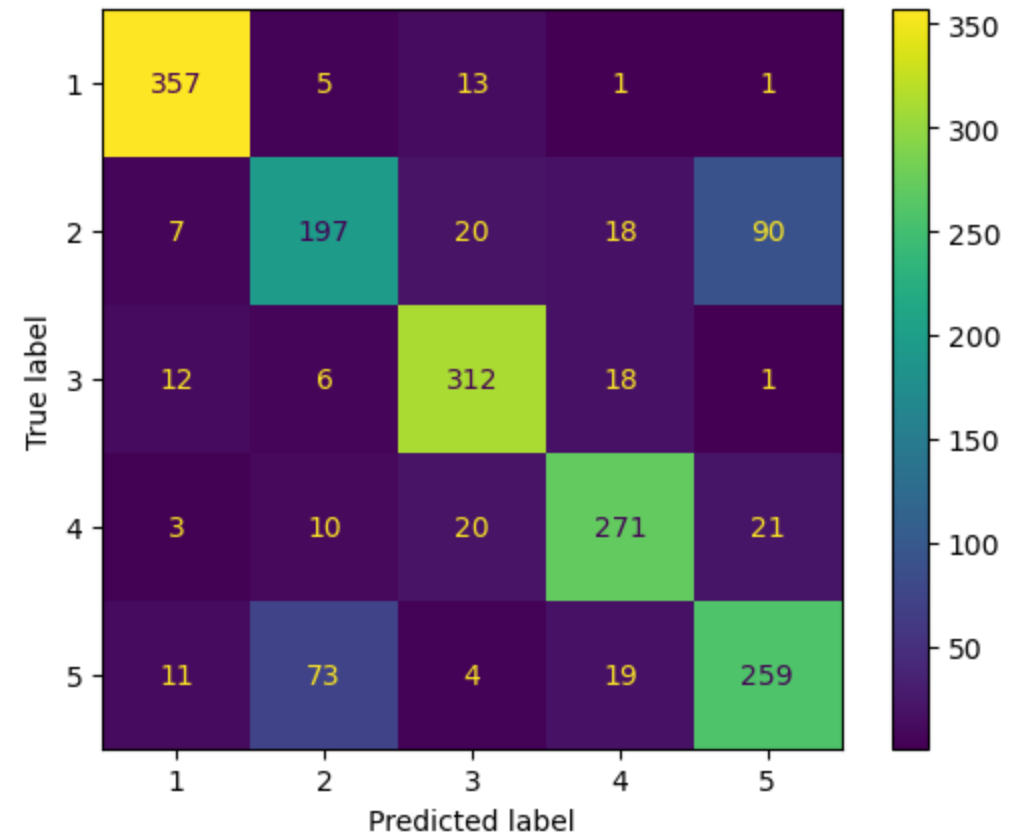
# Μέρος Δ

- Το επόμενο μοντέλο που θα δοκιμάσουμε είναι ένα μοντέλο boosting χρησιμοποιώντας ως βασικό μοντέλο ένα δέντρο απόφασης το οποίο θεωρείται γενικά αδύναμο μοντέλο και άρα ενδείκνυται για την εφαρμογή του boosting . Προκειμένου να βρούμε τις καλύτερες παραμέτρους για το Boosting πραγματοποιούμε παράλληλα με αυτό και ένα Grid Search. Ωστόσο, τα αποτελέσματα που παίρνουμε δεν είναι ικανοποιητικά αφού φτάνουμε μόλις 72.16% σε ακρίβεια κατηγοριοποίησης.
- Για την εφαρμογή του παραπάνω πραγματοποιήσαμε πρώτα scaling των δεδομένων μας και στην συνέχεια εφαρμόσαμε PCA προκειμένου να μειώσουμε την διάσταση και να επιταχύνουμε την διαδικασία του Grid Search



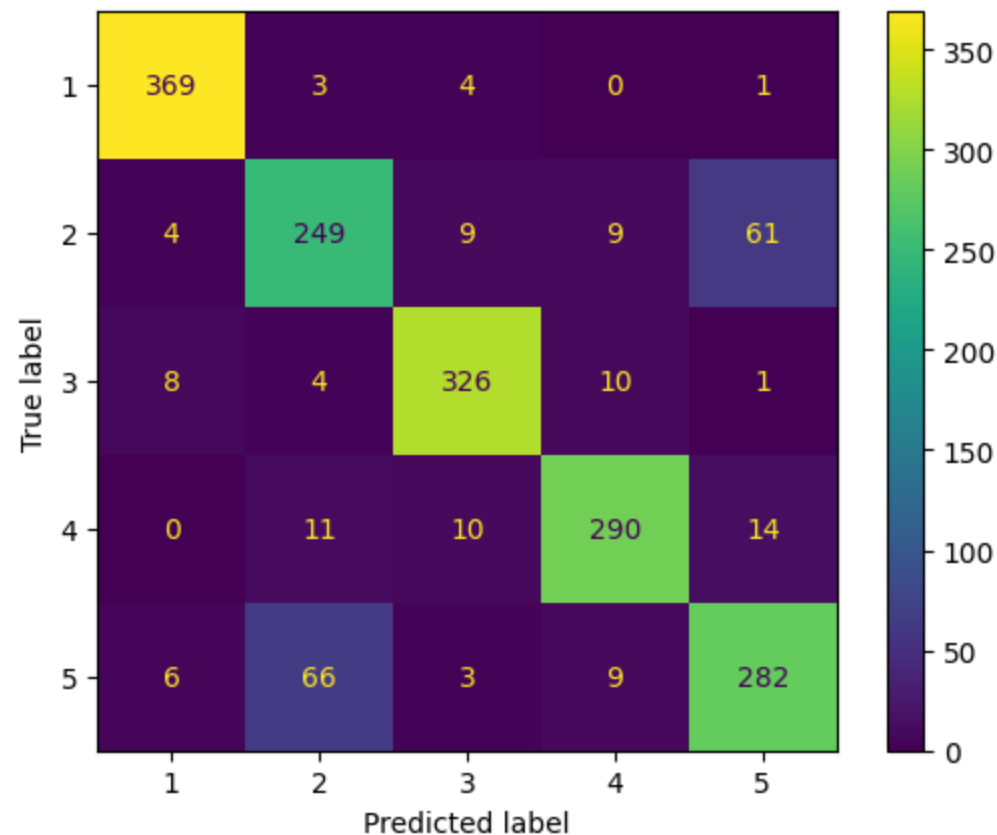
# Μέρος Δ

- Έπειτα δοκιμάζουμε έναν Random Forest ταξινομητή με τις καταλληλές παραμέτρους όπως προκύπτουν από την GridSearchCV().
- Επιτυγχάνουμε 79.817% ακρίβεια, συνεχίζοντας να έχουμε πρόβλημα στην ταξινόμηση της 2ης και 5ης κλάσης
- Για την εφαρμογή του παραπάνω πραγματοποιήσαμε πρώτα scaling των δεδομένων μας και στην συνέχεια εφαρμόσαμε PCA προκειμένου να μειώσουμε την διάσταση και να επιταχύνουμε την διαδικασία του Grid Search



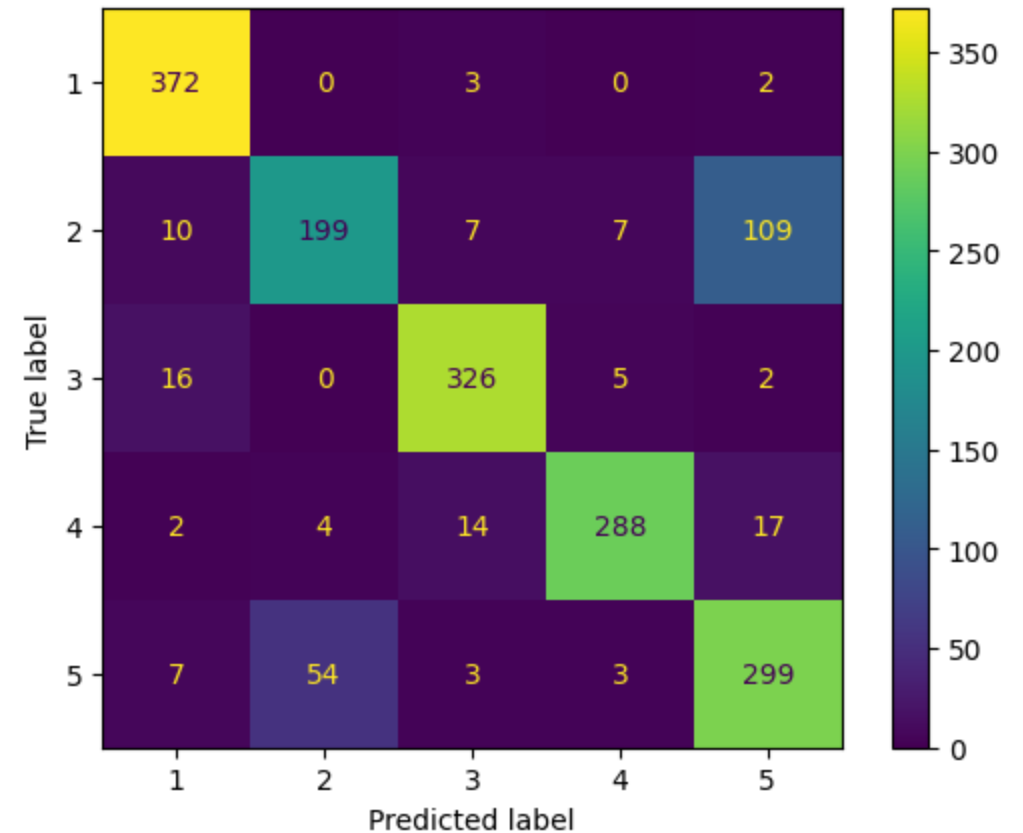
# Μέρος Δ

- Επίσης για την ταξινόμηση υλοποιούμε και ένα νευρωνικό δίκτυο χρησιμοποιώντας ωστόσο την PyTorch. Η δομή του νευρωνικού μας είναι αρκετά απλή καθώς έχει μόνο ένα hidden layer. Επιπλέον για να είναι συμβατό με την sklearn φτιάχνουμε μια wrapper κλάση προκειμένου να μπορούμε να χρησιμοποιήσουμε τις προηγούμενες μετρικές μας. Τα αποτελέσματα που λαμβάνουμε στο validation σετ είναι αρκετά ικανοποιητικά καθώς καταφέρνουμε να επιτύχουμε ένα αρκετά υψηλό ποσοστό ακρίβειας γύρω στο 86.68%



# Μέρος Δ

- Έπειτα υλοποιούμε ένα ταξινομητή K-Πλησιέστερου Γείτονα.
- Προκειμένου να βρούμε τις καλύτερες παραμέτρους για τον συγκεκριμένο ταξινομητή, χρησιμοποιούμε την `GridSearchCV()`.
- Παρατηρούμε ότι ο KNN ταξινομεί αρκετά καλά τα δεδομένα, καθώς επιτυγχάνει 84.8485% ακρίβεια.



# Μέρος Δ

- Ένα από τα τελευταία μοντέλα που δοκιμάσαμε ήταν ένα Stacking μοντελό.
- Αρχικά, έχοντας παρατηρήσει ότι το μεγαλύτερο πρόβλημα στην ταξινόμηση παρατηρείται στις κλάσεις 2 και 5, αποφασίζουμε να χρησιμοποιήσουμε δύο δυαδικούς SVM ταξινομητές , οι οποίοο δίνουν έμφαση στις κλάσεις 2 και 5 αντίστοιχα, χρησιμοποιώντας την προσέγγιση OneVsRest

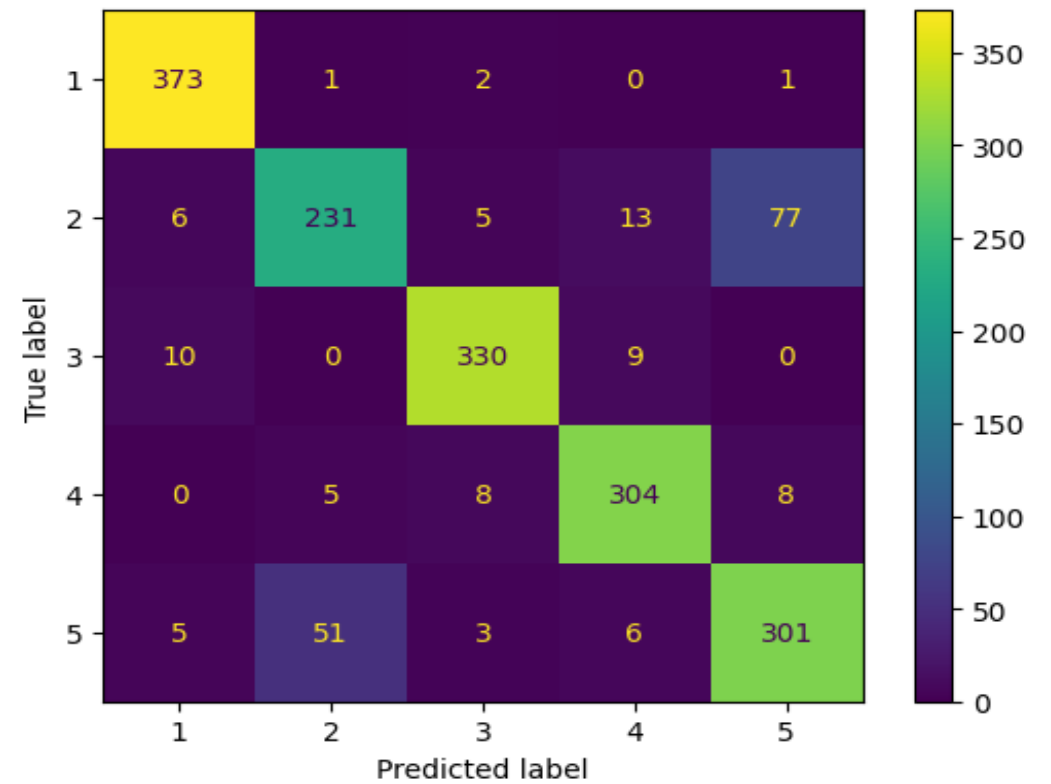
# Μέρος Δ

- Έπισης έχοντας παρατηρήσει ότι το MLP μοντέλο, ταξινομεί καλύτερα τις κλάσεις 2 και 5, εφαρμόζουμε ένα RandomizedSearchCV() στην προσπάθεια μας να προσδιορίσουμε τις κατάλληλες παραμέτρους.
- Το συγκεκριμένο μοντέλο το χρησιμοποιούμε σαν τελικό εκτιμητή στο stacking μοντέλο μας.
- Οι εκτιμητές που χρησιμοποιήθηκαν, η πλειονότητα των οποίων ήταν από προηγούμενα μοντέλα, είναι οι εξής:

```
estimators = [  
    ('svm', best_svm), # add the best SVM model  
    ('svm_bag', best_svm_bag), # add the best bagging SVM model  
    ('log_reg', best_log_reg), # add the best logistic regression model  
    ('knn', best_knn), # add the best kNN model  
    ("xgb", xgb.XGBClassifier(n_estimators=1000, Learning_rate=0.05, n_jobs=-1, random_state=42)), # add the XGBoost model  
    ("mlp", mlp_model2), # add the best MLP model  
    ('binary_svm_2', binary_svm_2), # add the binary SVM model for class 2  
    ('binary_svm_5', binary_svm_5), # add the binary SVM model for class 5  
]
```

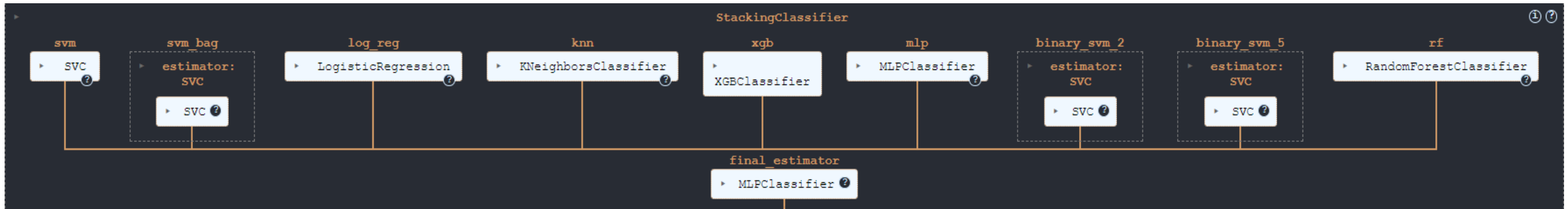
# Μέρος Δ

- Το confusion matrix που επιτύχαμε για το παραπάνω μοντέλο φαίνεται παρακάτω ενώ επίσης η ακρίβεια που επιτυγχάνουμε στο validation set είναι περίπου 87.99%



# Μέρος Δ

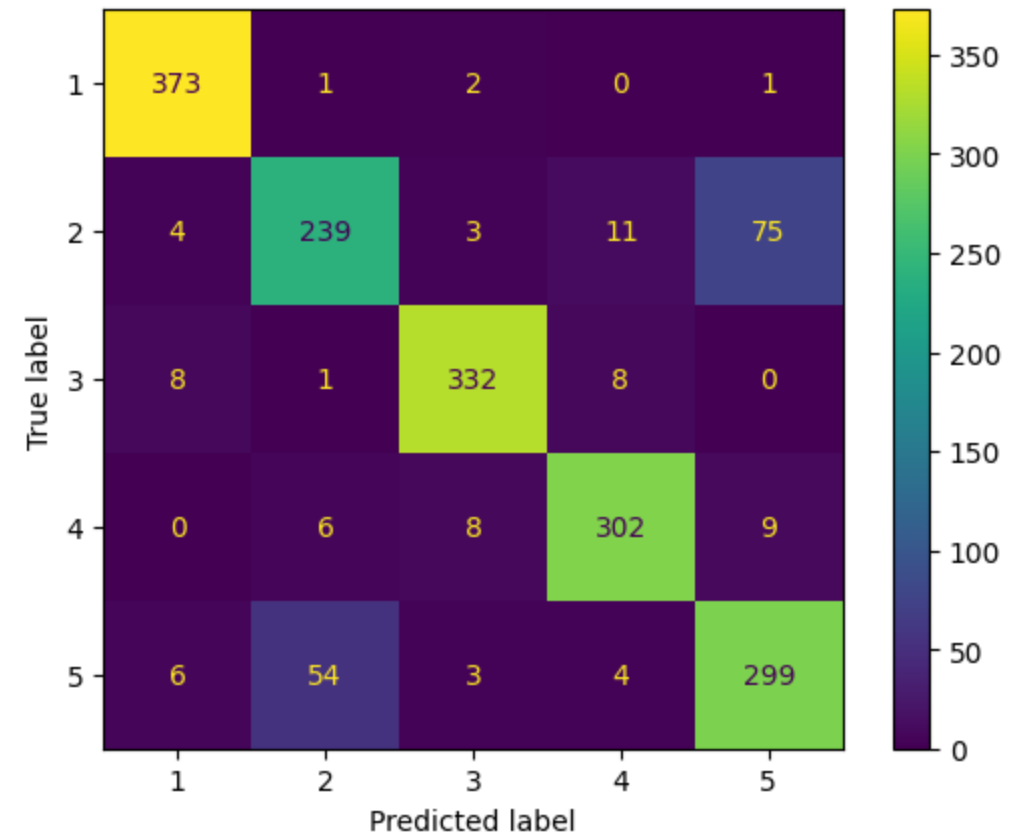
- Το τελευταίο και καλύτερο μοντέλο που υλοποιήσαμε, ήταν πάλι ένα Stacking μοντέλο, με την διαφορά ότι προσθέσαμε ένα random Forest ταξινομητή και επίσης βάλαμε στα μοντέλα βάρη για την κάθε κλάση.
- Αυτό το μοντέλο απέδωσε καλύτερα από όλα τα προηγούμενα. δίνοντάς μας ακρίβεια 88.33% στο validation set και είναι αυτό που χρησιμοποιήσαμε για την εξαγωγή των ζητούμενων labels για το test set που μας δώθηκε.





# Μέρος Δ

- Το confusion matrix που επιτύχαμε για το καλύτερο μας μοντέλο είναι το παραπάνω στο οποίο βλέπουμε πως έχουμε επιτύχει πολύ καλύτερα αποτελέσματα



# Μέρος Δ

- Παρακάτω φαίνονται οι ακρίβειες όλων των μοντέλων που δοκιμάστηκαν στο τελευταίο ζητούμενο της εργασίας.

