



Αναγνώριση Προτύπων & Μηχανική Μάθηση

2024 - 2025

Βαπόρης Δημήτριος ΑΕΜ: 10625
Μπαρμπαγιάννος Βασίλειος ΑΕΜ: 10685

Μάθημα: Αναγνώριση Προτύπων και Μηχανική Μάθηση

Εξάμηνο: Χειμερινό 2024 - 2025

Υπεύθυνος Καθηγητής: Παναγιώτης Πετραντωνάκης

Βοηθός διδασκαλίας: Υπ. Διδ. Στέφανος Παπαδόπουλος



Μέρος Α

Ασχολούμαστε με την αναγνώριση του επιπέδου του στρες στους χρήστες βιντεοπαιχνιδιών.

Έχουμε έναν δείκτη-αριθμό x τον οποίο θα χρησιμοποιήσουμε σε ένα σύστημα ταξινόμησης για να διαπιστωθεί κάθε φορά αν ο χρήστης αισθάνεται στρες ή όχι.

Έχουμε δύο κλάσεις:

- κλάση ω_1 : χωρίς στρες
- κλάση ω_2 : με στρες



Η κατανομή πυκνότητας πιθανότητας και για τις δύο κλάσεις είναι:

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

Η παράμετρος θ είναι άγνωστη. Ζητήσαμε από 12 συναδέλφους να παίξουν το παιχνίδι και μετρήσαμε τον δείκτη x για καθέναν από αυτούς. Μετά τους ρωτήσαμε αν ένιωσαν ή όχι στρες. Οι 7 δεν ένιωσαν στρες ενώ οι 5 ένιωσαν έντονο στρες.

Θα υλοποιήσουμε έναν ταξινομητή μέγιστης πιθανοφάνειας.




1ο ερώτημα:

Θα εκτιμήσουμε τις παραμέτρους θ_1 και θ_2 και για τις δύο κλάσεις. Η μέθοδος μέγιστης πιθανοφάνειας δίνει την καλύτερη εκτίμηση για την παράμετρο με βάση το δείγμα.

Για ευκολία στους υπολογισμούς, θα δουλέψουμε με τη συνάρτηση log-likelihood, η οποία για την κατανομή μας είναι:


$$\log L(\theta) = -N \log(\pi) - \sum_{i=1}^N \log(1 + (x_i - \theta)^2)$$



Για κάθε κλάση, θα βρούμε την τιμή της παραμέτρου που μεγιστοποιεί το log-likelihood.

Στην `pytho` εφαρμόζουμε αριθμητική βελτιστοποίηση για να βρούμε τις τιμές εκτιμήσεις των παραμέτρων θ_1 και θ_2 .


Επίσης θα απεικονίσουμε τις $\log p(D | \theta)$ σε συνάρτηση με τη θ για κάθε κλάση.



Στον κώδικά μας, αρχικά ορίζουμε την κλάση Classifier. Περιέχει τις μεταβλητές `theta1`, `theta2` που θα αποθηκεύσουν τις εκτιμήσεις των παραμέτρων θ_1 και θ_2 .

Η συνάρτηση `log_likelihood()` υπολογίζει την log-πιθανοφάνεια και είναι αρνητική, διότι θα την ελαχιστοποιήσουμε για να μεγιστοποιήσουμε την πιθανότητα.

Η συνάρτηση `fit()` δέχεται τα δεδομένα D1 και D2 για τις δύο κλάσεις. Κάνουμε `import` την `minimize` από την `scipy.optimize` για να ελαχιστοποιήσουμε την αρνητική log-πιθανοφάνεια για κάθε κλάση και να βρούμε τις εκτιμήσεις των παραμέτρων θ_1 , θ_2 .



Η συνάρτηση `compute_log_probability()` υπολογίζει την log-πιθανότητα για κάθε κλάση, για τα δεδομένα D1 και D2. Δηλαδή υπολογίζει τις $\log p(D1 | \theta)$ και $\log p(D2 | \theta)$.

Στη συνέχεια οπτικοποιούμε τα αποτελέσματα.



2ο ερώτημα:

Σε αυτό το ερώτημα θα ταξινομήσουμε τα δύο σύνολα τιμών χρησιμοποιώντας τη συνάρτηση διάκρισης:

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

Ο κανόνας απόφασης είναι απλός:

Αν $g(x) > 0$, το x ταξινομείται στην κλάση ω_1 , ενώ αν $g(x) < 0$ το x ταξινομείται στην κλάση ω_2 .

Ορίζουμε τις α-priori πιθανότητες των δύο κλάσεων ως $P(\omega_1) = 7/12$ και $P(\omega_2) = 5/12$.



Η κλάση Classifier περιέχει τις εξής συναρτήσεις:

1. *fit()* : υπολογίζει τις εκτιμήσεις των παραμέτρων θ_1 και θ_2 με την μέθοδο της μέγιστης πιθανοφάνειας.
2. *g_function()* : υπολογίζει τη συνάρτηση διάκρισης για ένα σημείο x .
3. *predict()* : υπολογίζει τις τιμές της συνάρτησης διάκρισης για ένα σύνολο δεδομένων.



Μέρος Β

Στο μέρος Β της εργασίας θα υλοποιήσουμε έναν νέο ταξινομητή, εκτιμώντας την άγνωστη παράμετρο θ με τη μέθοδο εκτίμησης κατά Bayes.

Η συνάρτηση πυκνότητας πιθανότητας για την παράμετρο θ είναι:

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

Θα υπολογίσουμε την α-posteriori πιθανότητα $p(\theta | D)$ και την πυκνότητα πιθανότητας $p(x | D_j)$, $j=1,2$.



1ο ερώτημα:

Ζητείται να απεικονίσουμε τις εκ των υστέρων συναρτήσεις πυκνότητας πιθανότητας $p(\theta | D1)$ και $p(\theta | D2)$.

Για να το κάνουμε αυτό θα χρησιμοποιήσουμε τον κανόνα του Bayes, σύμφωνα με τον τύπο:

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

για καθένα από τα σύνολα $D1, D2$.




Το ολοκλήρωμα θα το υπολογίσουμε με αριθμητική ολοκλήρωση με τον κανόνα του τραπεζίου.

Γι' αυτό το σκοπό εισάγουμε τη συνάρτηση *quad* της βιβλιοθήκης `scipy.integrate`.

Χρειάζεται να υπολογίσουμε την α-priori πιθανότητα $p(\theta)$, το οποίο γίνεται με τη συνάρτηση *priori()*.

Την πιθανοφάνεια $p(D | \theta)$ την υπολογίζουμε με την *likelihood()*.

Τέλος υπολογίζει τις α-posteriori πιθανότητες $p(\theta | D)$ με εφαρμογή του κανόνα του Bayes.



Η μπλε καμπύλη δείχνει πιο διασπαρμένη κατανομή με υψηλή πιθανότητα γύρω από θετικές τιμές θ .

Η κόκκινη καμπύλη είναι πιο συγκεντρωμένη και έχει υψηλή πιθανότητα για αρνητικές τιμές του θ .

Και οι δύο *a-posteriori* κατανομές συγκεντρώνονται γύρω από τις πιο πιθανές τιμές για το θ , όπως υποδηλώνουν και τα δεδομένα $D1$ και $D2$.

Σε σχέση με την *priori* πιθανότητα παρατηρούμε ότι:

Η $p(\theta)$ είναι πιο πλατιά δείχνοντας έτσι την αρχική αβεβαιότητα για το θ και οι εκ των υστέρων πιθανότητες είναι πιο στενές αφού τα δεδομένα αυξάνουν τη βεβαιότητά μας για την εκτίμηση της παραμέτρου.




2ο ερώτημα:

Θα υλοποιήσουμε μια συνάρτηση `predict` που θα υπολογίζει τις τιμές μιας συνάρτησης διάκρισης:

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

Αρχικά κατασκευάζουμε τη συνάρτηση $h(x)$ και έπειτα καλούμε τη συνάρτηση `predict()` για να μας υπολογίσει την $h(x)$ για μερικές τιμές x . Έπειτα μας λέει σε ποια κλάση ανήκει κάθε x .



Η μέθοδος εκτίμησης παραμέτρων κατά Bayes λαμβάνει υπόψη της την prior πιθανότητα για τα δεδομένα, κάτι που δεν κάνει η μέθοδος μέγιστης πιθανοφάνειας.

Η $h(x)$ καταφέρει να αποφασίσει με 100% ακρίβεια για όλα τα δεδομένα των συνόλων $D1$ και $D2$.

Η διαφορά των δύο προσεγγίσεων στο συγκεκριμένο παράδειγμα οφείλεται στο γεγονός ότι η $D2$ έχει λιγότερες παρατηρήσεις οπότε στη μέθοδο μέγιστης πιθανοφάνειας η πιθανότητα $p(x | D2)$ θα επηρεαστεί δυσανάλογα, ενώ η μέθοδος Bayes η prior πιθανότητα εξισορροπεί αυτή τη διαφορά.



Μέρος Γ | 1η ενότητα

Εργαζόμαστε πάνω σε μία έρευνα που αφορά την αυτοματοποιημένη αναγνώριση διαφορετικών ειδών του φυτού της Ίριδας.

Τρία συγκεκριμένα είδη: η *Iris setosa*, η *Iris versicolor*, και η *Iris virginica* παρουσιάζουν διαφορές στο μήκος και πλάτος των σεπάλων και των πετάλων του άνθους τους.

Διαθέτουμε μια βάση από 150 (50 για κάθε είδος) μετρήσεις του μήκους και του πλάτους των σεπάλων και των πετάλων του άνθους κάθε είδους.

Απομονώνοντας μόνο τα δύο πρώτα χαρακτηριστικά της βάσης και χρησιμοποιώντας τον έτοιμο αλγόριθμο `DecisionTreeClassifier` από τη βιβλιοθήκη `sklearn` θα ταξινομήσουμε το 50% των τυχαίων δειγμάτων του συνόλου αφού πρώτα εκπαιδεύσουμε τον αλγόριθμο με το υπόλοιπο 50%.



1ο ερώτημα:

Σε αυτό το ερώτημα εκπαιδεύουμε το `DecisionTreeClassifier` με το 50% των δεδομένων και αξιολογούμε το ποσοστό ακρίβειας με το υπόλοιπο 50%.

Οπότε φορτώνουμε τα δεδομένα, τα διαχωρίζουμε και εκπαιδεύουμε το `DecisionTreeClassifier` για διάφορες τιμές βάθους του δέντρου. Υπολογίζουμε την ακρίβεια για κάθε βάθος.

Η μέγιστη ακρίβεια που λαμβάνουμε είναι 0.79 για βάθος ίσο με 3. Το βάθος είναι σχετικά μικρό, άρα το μοντέλο είναι απλό.



2ο ερώτημα:

Τώρα θα απεικονίσουμε τα όρια απόφασης του ταξινομητή για το καλύτερο αποτέλεσμα.

Ο κώδικας δημιουργεί ένα πλέγμα σημείων για τα δύο χαρακτηριστικά του σεπάλου (μήκος και πλάτος).

Έπειτα κάνει πρόβλεψη για κάθε σημείο του πλέγματος με το εκπαιδευμένο μοντέλο.

Τέλος με τη συνάρτηση `contourf()` οπτικοποιεί τα όρια απόφασης.



Το γράφημα που παράγει ο κώδικας:

- ❖ Απεικονίζει τα όρια απόφασης για τον ταξινομητή.
- ❖ Τα σημεία εκπαίδευσης (train) και δοκιμής (test) φαίνονται με διαφορετικό σύμβολο και χρώμα (βλέπε legend).
- ❖ Οι διαφορετικές περιοχές χρώματος αναπαριστούν τις κλάσεις που προβλέπει το μοντέλο.



Μέρος Γ | 2η ενότητα

Τώρα θα δημιουργήσουμε έναν Random Forest ταξινομητή 100 δέντρων με την τεχνική Bootstrap.

Το 50% των δειγμάτων που χρησιμοποιήσαμε για εκπαίδευση στην προηγούμενη ενότητα (σύνολο A) το χρησιμοποιούμε τώρα για την δημιουργία 100 νέων συνόλων εκπαίδευσης, ένα για κάθε δέντρο, όπου κάθε φορά θα χρησιμοποιείται το $\gamma = 50\%$ του συνόλου A . Το σύνολο που ταξινομήσαμε στο προηγούμενο μέρος θα το χρησιμοποιήσουμε και εδώ για αξιολόγηση του αλγορίθμου. Όλα τα δέντρα θα έχουν το ίδιο μέγιστο βάθος.



1ο ερώτημα:

Φορτώνουμε το σύνολο δεδομένων Iris και διαχωρίζουμε το σύνολο A.

Δημιουργούμε 100 δείγματα bootstrap.

Εκπαιδεύουμε έναν Random Forest ταξινομητή με διαφορετικά βάθη (1 έως 10) και υπολογίζουμε την ακρίβεια σε κάθε βάθος.

Επίσης δημιουργούμε γράφημα με την ακρίβεια του Random Forest σε κάθε βάθος. Παρατηρούμε ότι το βέλτιστο βάθος είναι 2.



2ο ερώτημα:

Σε αυτό το ερώτημα απεικονίζουμε τα όρια απόφασης για τους δύο ταξινομητές: Decision Tree και Random Forest.

Τα όρια απόφασης του Decision Tree είναι απλούστερα και ευθύγραμμα.

Για το Random Forest, τα όρια απόφασης είναι πιο ομαλά, πιο λεπτομερή και πιο προσαρμοσμένα στα δεδομένα, παρέχοντας καλύτερη ταξινόμηση έναντι του Decision Tree.



3ο ερώτημα:

Πώς πιστεύετε ότι επηρεάζει το ποσοστό γ την απόδοση του αλγορίθμου; Δώστε παραδείγματα.

Το γ καθορίζει το ποσοστό των δειγμάτων που επιλέγονται τυχαία (με επανατοποθέτηση) από το σύνολο A για την εκπαίδευση κάθε δέντρου.



Μεγάλο $\gamma \Rightarrow$ χρήση περισσότερων δειγμάτων.

Τα δέντρα εκπαιδεύονται σε πιο ολοκληρωμένα υποσύνολα, άρα πιο επαρκής εκπαίδευση και μείωση υπερπροσαρμογής.

Μικρό $\gamma \Rightarrow$ χρήση λιγότερων δειγμάτων.

Τα δέντρα είναι λιγότερο ακριβή μεμονωμένα, αλλά η μεγαλύτερη ποικιλία οδηγεί σε καλύτερη γενίκευση.

Αν όμως το γ είναι υπερβολικά μικρό τα δέντρα θα είναι υπερβολικά απλά και η απόδοση μπορεί να μειωθεί.

Στον κώδικά μας έχουμε τρία παραδείγματα, για $\gamma=0.1$, $\gamma=0.5$ και $\gamma=0.9$.



Μέρος Δ

Ζητούμενο του μέρους Δ είναι η ανάπτυξη ενός αλγορίθμου ταξινόμησης μεταξύ 5 κλάσεων.

Το αρχείο `datasetTV.csv` περιέχει το `training set`, το οποίο αποτελείται από 8743 δείγματα με 224 χαρακτηριστικά το καθένα, ενώ περιέχει και το σωστό `label` για κάθε δειγμα.


Το αρχείο `datasetTest.csv` περιέχει το `testing set` του αλγορίθμου μας, αποτελείται από 6955 δείγματα με 224 χαρακτηριστικά και δεν περιέχει το `ορθό label`.



Πρώτες προσπάθειες - Κατανόηση του dataset

Αρχικά, δοκιμάστηκαν απλοί αλγόριθμοι GridSearch για τους ταξινομητές Random-Forest-Classififer και K-Nearest-Neighbors-Classififer, όπως έχει παρουσιαστεί στις διαλέξεις ασκήσεων.

Επιτεύχθηκε ακρίβεια γύρω στο 81% και για τους δύο αλγορίθμους. Πρόκειται για καλή αρχική τιμή, αλλά είναι επιθυμητή η βελτίωση.



Παρατηρούμε ότι το δεδομένο dataset έχει ιδιαίτερα υψηλό αριθμό δειγμάτων (224 για την ακρίβεια). Το πρόβλημα, δηλαδή, είναι 224-διάστατο, γεγονός που καθιστά ιδιαίτερα δύσκολη την επίλυσή του.

Δοκιμάζονται, επομένως, διάφορες μέθοδοι dimensionality reduction, ώστε να μειωθεί η διάσταση του προβλήματος και λοιπόν να γίνει πιο διαχειρίσιμη η επίλυσή του.


Στο μάθημα παρουσιάστηκε η μέθοδος PCA, οπότε ξεκινάμε την ανάλυση από αυτήν.



Μέθοδος PCA

Η μέθοδος PCA (Principal Component Analysis) προσπαθεί να προβάλει τα δεδομένα ενός dataset γραμμικά σε έναν χώρο χαμηλότερης διάστασης.

Στόχος είναι η μεγιστοποίηση της διακύμανσης, δηλαδή ουσιαστικά να παραμείνουν τα χαρακτηριστικά που προκαλούν τη μεγαλύτερη διαφοροποίηση και να αφαιρεθούν αυτά που δε μεταφέρουν ιδιαίτερη πληροφορία για την ταξινόμηση.



Εφαρμόζοντας τη μέθοδο PCA στο δεδομένο dataset, παρατηρούμε ότι πολλά χαρακτηριστικά συμμετέχουν πράγματι στη διακύμανση του dataset, οπότε δεν είναι ιδιαίτερα επιτυχής.

Για παράδειγμα, εκτελώντας την εντολή `PCA(0.9)`, που διατηρεί χαρακτηριστικά μέχρι να μεταφραστεί τουλάχιστον το 90% του συνολικού variance, διατηρούνται περίπου 100 χαρακτηριστικά.

Υπάρχει, δηλαδή, dimensionality reduction, αλλά όχι σε επίπεδο που μπορούν τα δεδομένα να οπτικοποιηθούν και να είναι δυνατόν να αντληθούν συμπεράσματα από αυτά.



Άλλες μέθοδοι Dimensionality Reduction Manifold Learning

Λόγω της γραμμικής του φύσης, ο αλγόριθμος PCA δεν ήταν ικανός να ελαχιστοποιήσει σημαντικά τη διάσταση του προβλήματος.

Δοκιμάζονται μερικές μέθοδοι μη γραμμικού dimensionality reduction, οι οποίες δεν παρουσιάστηκαν στο μάθημα.

Συγκεκριμένα, δοκιμάζονται οι αλγόριθμοι TSNE (T-distributed Stochastic Neighbor Embedding) και UMAP (Uniform Manifold Approximation and Mapping).

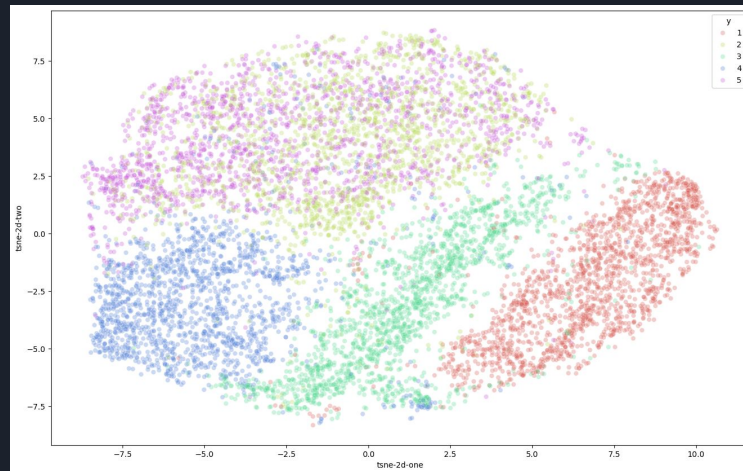


Μέθοδος TSNE

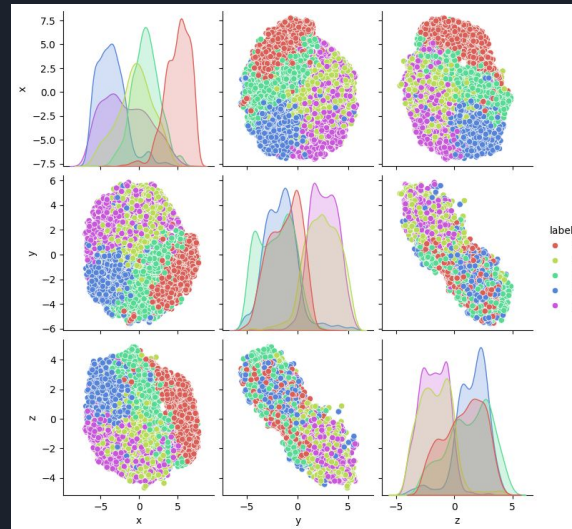
Η μέθοδος TSNE είναι ικανή να οπτικοποιήσει δεδομένα υψηλότερων διαστάσεων, διατηρώντας τη σχετική απόσταση μεταξύ σημείων σταθερή. Δηλαδή, δεδομένα που βρίσκονται μακριά στις 224 διαστάσεις θα παραμείνουν μακριά και στις 2 διαστάσεις και το αντίθετο.

Προτού εφαρμοστεί η μέθοδος, χρησιμοποιείται η μέθοδος PCA για να μειωθεί η διάσταση του προβλήματος ικανοποιητικά (το documentation του TSNE στο scikit learn αναφέρει ως παράδειγμα τις 50 διαστάσεις).

Παρακάτω φαίνονται τα αποτελέσματα του αλγορίθμου TSNE στο dataset μας. Παρατηρούμε ότι τα δεδομένα διαχωρίζονται ικανοποιητικά για τις κλάσεις 1, 3 και 4, ενώ για τις κλάσεις 2 και 5 υπάρχει μεγάλο overlap, τουλάχιστον στις 2 διαστάσεις. Δοκιμάστηκαν αλγόριθμοι ταξινόμησης στο dataset μετά τη μετατροπή, χωρίς ιδιαίτερη βελτίωση (περίπου 82% ακρίβεια).



Δοκιμάζεται ο αλγόριθμος TSNE και για τις 3 διαστάσεις. Όπως φαίνεται από τα παρακάτω διαγράμματα, η επικάλυψη των κλάσεων 2 και 5 παραμένει υψηλή, ακόμα και σε μία διάσταση πάνω. Δοκιμάστηκαν και πάλι ταξινομητές, χωρίς αξιοσημείωτη βελτίωση.





Μέθοδος UMAP

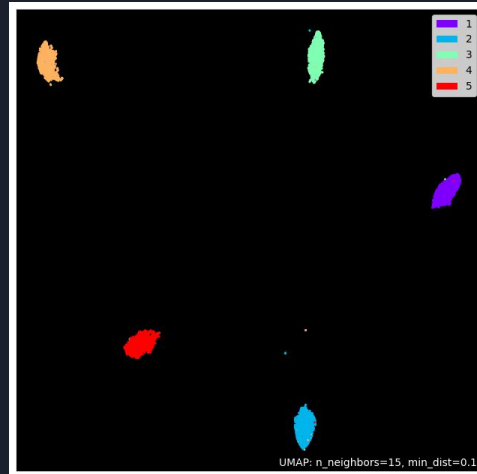
Η μέθοδος UMAP δουλεύει παρόμοια με τη μέθοδο TSNE όσον αφορά τη δυνατότητα οπτικοποίησης των δεδομένων.


Είναι ιδιαίτερα ικανή, επίσης, στο γενικό dimensionality reduction.

Στόχος της μεθόδου είναι η μετατροπή των δεδομένων σε κάποιον χώρο χαμηλότερης διάστασης με τη μεγαλύτερη τοπολογική συγγένεια με τον αρχικό χώρο.

Πρόκειται, δηλαδή, για μη γραμμικό αλγόριθμο Manifold Learning, όπως ο TSNE.

Παρακάτω φαίνονται τα αποτελέσματα της εφαρμογής της μεθόδου UMAP σε ένα υποσύνολο του dataset. Φαίνεται ότι ο αλγόριθμος είναι ικανός να διαχωρίσει τις κλάσεις. Ωστόσο, δοκιμάζοντας ταξινομητές μετά τη μετατροπή των δεδομένων για το validation set, παρατηρούμε χαμηλή ακρίβεια (γύρω στο 82%). Αυτό οφείλεται στο overfitting που συμβαίνει, καθώς ο αλγόριθμος δεν ανταπεξέρχεται στα νέα δεδομένα του validation set.





Το παραπάνω αποτέλεσμα φαίνεται αρκετά ικανό να προσφέρει βελτίωση. Δοκιμάζεται, επομένως, η δημιουργία ενός Pipeline με κατάλληλο data scaling, data transformation με UMAP και ταξινόμηση με Random-Forest-Classifer.

Εγκαθιστούμε και χρησιμοποιούμε τη βιβλιοθήκη optuna, ειδική στο να λύνει προβλήματα βελτιστοποίησης υπερπαραμέτρων, ώστε να βρεθούν οι βέλτιστες υπερπαραμέτροι του Pipeline και να δημιουργηθεί ο καλύτερος δυνατός αλγόριθμος ταξινόμησης.

Προκύπτει βέλτιστος αλγόριθμος ταξινόμησης με ακρίβεια 84%. Ιδιαίτερο ενδιαφέρον προκαλεί το γεγονός ότι η μέθοδος UMAP γίνεται βέλτιστη για μείωση σε 223 (από τις 224) διαστάσεις.




Αλλαγή Προσέγγισης - Προχωρημένοι Ταξινομητές

Η τακτική dimensionality reduction δεν επέφερε σημαντικές βελτιώσεις.

Αντ' αυτής, δοκιμάζονται προχωρημένοι ταξινομητές με βάση τον Random-Forest-Classifer.


Συγκεκριμένα, θα χρησιμοποιηθούν οι αλγόριθμοι XGBoost (παρέχει parallel tree boosting, που προσδίδει ταχύτητα και ακρίβεια στον τυπικό RFC) και LightGBM (boosting framework με λιγότερες απαιτήσεις σε μνήμη και υψηλότερη αποδοτικότητα).



Δοκιμάζονται οι δύο νέοι ταξινομητές και επιστρέφουν αποτελέσματα με ακρίβεια από 84% έως 85%.

Χρησιμοποιώντας μετρικές όπως το classification report, παρατηρούμε ότι το ιδιαίτερο πρόβλημα στη διάκριση βρίσκεται μεταξύ των κλάσεων 2 και 5, στις οποίες υπάρχει ακρίβεια γύρω στο 70%. Για τις υπόλοιπες κλάσεις η ακρίβεια είναι ικανοποιητικά υψηλή.


Έχοντας λάβει ικανοποιητικά αποτελέσματα ήδη χωρίς hyperparameter tuning, η ανάλυση προχωρά με τη διαδικασία αυτή.



Λόγω της ταχύτητάς του, η οποία επιτρέπει να πραγματοποιηθούν πολλές δοκιμές, επιλέγεται η επιλογή υπερπαραμέτρων για τη μέθοδο LightGBM.

Ακολουθώντας τον κλασικό τρόπο GridSearch, όπως παρουσιάστηκε στο μάθημα, για τη μέθοδο LightGBM, βρίσκονται οι βέλτιστες υπερπαραμέτροι οι οποίες οδηγούν σε ακρίβεια περίπου 85%.

Εφόσον οι προβληματικές κλάσεις είναι 2, δοκιμάζεται μία εναλλακτική μέθοδος: να δημιουργηθεί ειδικός δυαδικός ταξινομητής που να ταξινομεί κατάλληλα τα δεδομένα των 2 επικαλυπτόμενων κλάσεων.




Δοκιμάζουμε την παραπάνω τακτική για τους δύο νέους ταξινομητές. Προκύπτει χωρίς hyperparameter tuning ακρίβεια γύρω στο 85.5%.

Για τον ίδιο λόγο με πριν, για το hyperparameter tuning επιλέγεται ο αλγόριθμος με τους δύο ταξινομητές LightGBM, τον κλασικό multi-class και τον δυαδικό.

Μετά το hyperparameter tuning, επιτυγχάνεται ακρίβεια 86.3%, που είναι και η καλύτερη που έχει βρεθεί ως τώρα.

Επιλέγουμε, λοιπόν, τον αλγόριθμο αυτόν: πρώτα έναν multi-class LGBM ταξινομητή και μετά έναν δυαδικό ταξινομητή LGBM για τις κλάσεις 2 και 5 (με τη μεγάλη επικάλυψη).



Επιβεβαιώνονται τα αποτελέσματα για την ακρίβεια που βρέθηκε κατά τη διάρκεια της βελτιστοποίησης μέσω `optuna`, εισάγοντας στους ταξινομητές ακριβώς τις βέλτιστες παραμέτρους που υπολογίστηκαν.

Με τον αλγόριθμο αυτόν, προπονούνται τώρα οι ταξινομητές με το πλήρες `training set` του αρχείου `datasetTV.csv`. Οι προπονημένοι ταξινομητές προβλέπουν τα `labels` για το `testing set` του αρχείου `datasetTest.csv` και οι προβλέψεις αποθηκεύονται στο ζητούμενο αρχείο `labels59.npy`.

Δοκιμάζεται αν το μέγεθος του αρχείου `labels59` είναι σωστό, όπως και το αν διαβάζεται με την εντολή `numpy.load()`.



Συμπεράσματα - Σχολιασμός Αποτελεσμάτων

Η υψηλότερη ακρίβεια παρατηρείται για τον αλγόριθμο με τον ειδικό δυαδικό ταξινομητή που προπονείται για να διακρίνει μεταξύ των 2 επικαλυπτόμενων κλάσεων, πέραν του κλασικού multi-class classifier.

Η ιδέα του dimensionality reduction δεν επέφερε σημαντικές βελτιώσεις, ωστόσο θα μπορούσε πιθανώς να οδηγήσει και σε βελτίωση του αλγορίθμου που προτάθηκε.

Δεν εξερευνήθηκε η ιδέα της αύξησης των χαρακτηριστικών με μη γραμμικές μεθόδους με στόχο τη διάκριση μεταξύ των κλάσεων 2 και 5. Πιθανόν η μέθοδος αυτή να οδηγούσε σε βελτίωση της ακρίβειας.