ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΝΑΦΟΡΑ ΑΝΑΓΝΩΡΙΣΗΣ ΨΗΦΙΩΝ

Τρυφωνόπουλος Δημήτριος edem0037

Βήμα 1

Διαβάστε τα δεδομένα από το αρχείο. Τα δεδομένα πρέπει να διαβαστούν σε μορφή συμβατή με το scikit-learn σε 4 πίνακες X_train, X_test, y_train και y_test. Ο πίνακας X_train περιέχει τα δείγματα εκπαίδευσης ,χωρίς τα labels) και είναι διάστασης (n_samples_train x n_features). Ο y_train είναι ένας μονοδιάστατος πίνακας μήκους n_samples και περιέχει τα αντίστοιχα labels για τον X_train. Αντίστοιχα για τα test δεδομένα.

Διαβάζουμε τα δεδομένα χρησιμοποιώντας την **np.loadtxt** μιας και το αρχείο μας είναι σε μορφή **.txt**. Για το **X_train & X_test** κρατάμε όλα τα δεδομένα εκτός από αυτά της πρώτης στήλης μιας και περιέχουν το label του κάθε ψηφίου. Τα στοιχεία (labels) της πρώτης στήλης διαβάζονται ως **y_train & y_test** μιας και είναι η πραγματικές προβλέψεις για κάθε στοιχείο.

Βήμα 2

Σχεδιάστε το υπ' αριθμόν 131 ψηφίο , (βρίσκεται στη θέση 131) των train δεδομένων. Υπόδειξη: χρησιμοποιήστε τη συνάρτηση numpy.reshape για να οργανώσετε τα 256 χαρακτηριστικά σε ένα πίνακα 16x16, και τη συνάρτηση matplotlib.pyplot.imshow για την απεικόνιση του ψηφίου.

Για να απεικονίσουμε το στοιχείο στην θέση 131 επιλέγουμε τα χαρακτηριστικά του ψηφίου αυτού από το X_train[131,:], αλλάζουμε το σχήμα του vector σε 16x16 για να καταφέρουμε να τα απεικονίσουμε.

Βήμα 3

Διαλέξτε 1 τυχαίο δείγμα από κάθε label ,συνολικά 10 δείγματα). Σχεδιάστε τα σε ένα figure με subplots. (`Hint: fig = plt.figure(); fig.add_subplot(,,,))`

Χρησιμοποιώντας την συνάρτηση **random** επιλέγουμε τυχαία για κάθε νούμερο από το 0-9, κάποιο από όλα τα δείγματα, αφού πρώτα έχουμε κρατήσει σε ένα πίνακα το κάθε νούμερο επαναληπτικά. Αφού αλλάξουμε και πάλι το σχήμα από line vector σε 16x16 είμαστε σε θέση να απεικονίσουμε το κάθε τυχαία επιλεγμένο δείγμα για κάθε έναν από τους αριθμούς.

Βήμα 4

Υπολογίστε τη μέση τιμή των χαρακτηριστικών του pixel (10, 10) για το ψηφίο «0» με βάση τα train δεδομένα.

Αφού κρατήσουμε μόνο τα δείγματα που περιέχουν τον αριθμό '0' βρίσκουμε το index για τον αριθμό στο pixel (10,10) και υπολογίζουμε την μέση τιμή του (np.mean) 10*16 + 11 = 171.
Μια άλλη επίλυση του παραπάνω προβλήματος μέσω της συνάρτησης np.reshape.

Βήμα 5

Υπολογίστε τη διασπορά των χαρακτηριστικών του pixel (10, 10) για το ψηφίο «0» με βάση τα train δεδομένα.

Αντίστοιχη υλοποίηση με το προηγούμενο βήμα αλλά αντί για το np.mean θα χρησιμοποιήσουμε το **np.std.**

Βήμα 6
Υπολογίστε τη μέση τιμή και διασπορά των χαρακτηριστικών κάθε pixel για το ψηφίο «0» με βάση τα train δεδομένα.

Αφού κρατήσουμε σε έναν πίνακα (A) όλα τα samples που αντιπροσωπεύουν το ψηφίο '0' βάση των labels από τα y_train δεδομένα. Παίρνουμε την μέση τιμή ανά στήλη (axis=0) και έτσι βρίσκουμε την μέση τιμή για κάθε ένα pixel του αριθμού μηδέν για όλα τα δείγματα.

Βήμα 7
Σχεδιάστε το ψηφίο «0» χρησιμοποιώντας τις τιμές της μέσης τιμής που υπολογίσατε στο Βήμα 6.

Χρησιμοποιώντας το mat_mean πίνακα που βρήκαμε παραπάνω και αφού το αναδιατάξουμε σχεδιάζουμε το ψηφίο '0' βασισμένοι στην μέση τιμή του κάθε pixel (feature) του.

Βήμα 8
Σχεδιάστε το ψηφίο «0» χρησιμοποιώντας τις τιμές της διασποράς που υπολογίσατε στο Βήμα 6. Συγκρίνετε το αποτέλεσμα με το αποτέλεσμα του Βήματος 7 και εξηγείστε τυχόν διαφορές.

Σχεδιάζοντας το ψηφίο '0' χρησιμοποιώντας τις τιμές της διασποράς για όλα τα pixel του παρατηρούμε ότι από το ψηφίο 'λείπει' το μέρος στην μέση, αναμενόμενο μιας και η διασπορά συμβολίζει τις +- αποκλίσεις από την μέση τιμή. Για την καλύτερη απεικόνιση παρατίθεται στην αναφορά το άθροισμα αλλά και η διαφορά της μέσης τιμής και της απόκλισης με την χρήση της εντολής subplot.

Βήμα 9
(α) Υπολογίστε τη μέση τιμή και διασπορά των χαρακτηριστικών για όλα τα ψηφία (0-9) με βάση τα train δεδομένα.
(β) Σχεδιάστε όλα τα ψηφία χρησιμοποιώντας τις τιμές της μέσης τιμής που υπολογίσατε στο Βήμα 9(α).

Αντίστοιχα με τα προηγούμενα ερωτήματα δημιουργούμε έναν πίνακα διαστάσεων (10,256) όπου 10- διαφορετικά νούμερα, 256 -τιμές όπου είναι οι μέση τιμή για κάθε pixel για κάθε νούμερο.
Εν συνεχεία αφού τα αναδιατάξουμε σε (16x16) τα σχεδιάζουμε.

Βήμα 10
Ταξινομήστε το υπ΄αριθμόν 101 ψηφίο των test δεδομένων (βρίσκεται στη θέση 101) σε μία από τις 10 κατηγορίες (κάθε ένα από τα 10 ψηφία, 0-9, αντιπροσωπεύει μία κατηγορία) βάσει της της Ευκλείδειας απόστασης[1] (υπόδειξη: χρησιμοποιείστε τις τιμές που υπολογίσατε στο Βήμα 9(α)). Ήταν επιτυχής η ταξινόμηση;

Αφού υπολογίσουμε την ευκλείδεια απόσταση μεταξύ κάθε δείγματος από την μέση τιμή κάθε χαρακτηριστικού που υπολογίσαμε στα προηγούμενα παραδείγματα, κρατάμε την ελάχιστη, μαζί με το αντίστοιχο index (np.argmin) το οποίο και αντιπροσωπεύει τον αριθμό με τον οποίο το εν λόγο δείγμα βρέθηκε να έχει την μικρότερη ευκλείδεια απόσταση, την αντίστοιχη πρόβλεψη δηλαδή. Παρατηρούμε όμως τελικά ότι το classification δεν ήταν σωστό γεγονός που εξηγείται αν αναπαραστήσουμε γραφικά το συγκεκριμένο δείγμα. Θα διαπιστώσουμε ότι είναι ο αριθμός 6 αντί του 0 αλλά λόγω του τρόπου γραφής του μοιάζει με το 0.

Βήμα 11
(α) Ταξινομήστε όλα τα ψηφία των test δεδομένων σε μία από τις 10 κατηγορίες με βάση την Ευκλείδεια απόσταση.
(β) Υπολογίστε το ποσοστό επιτυχίας για το Βήμα 11(α).

Επαναλαμβάνουμε την παραπάνω διαδικασία ταξινόμησης βασιζόμενοι στην ευκλείδεια απόσταση. Το ποσοστό επιτυχίας ανέρχεται σε ~ 81.42%

Βήμα 12
Υλοποιήστε τον ταξινομητή ευκλείδειας απόστασης σαν ένα scikit-learn estimator.
(Hint: το αρχείο lib.py που σας δίνεται)

Ακολουθώντας την φόρμα των κλάσεων των sklearn ταξινομητών (_init_, fit, predict, score) δημιουργούμε την συνάρτηση του ευκλείδειου ταξινομητή.

Βήμα 13
α) Υπολογίστε το score του ευκλείδειου ταξινομητή με χρήση 5-fold cross-validation
β) Σχεδιάστε την περιοχή απόφασης του ευκλείδειου ταξινομητή.
γ) Σχεδιάστε την καμπύλη εκμάθησης του ευκλείδειου ταξινομητή (learning curve)

α) Καλώντας την cross_val_score function από το πακέτο sklearn.model_selection, και την K-fold με αριθμό χωρισμάτων n_splits=5 βρίσκουμε το score του ταξινομητή αυτου.

β) Από το tutorial_Skikit-learn του μαθήματος αντιγράφουμε την plot_clf συνάρτηση η οποία και χρησιμοποιείται για να δημιουργήσει το Decision Surface για τον ευκλείδειο ταξινομητή.

γ) Σχεδιάζουμε την καμπύλη εκμάθησης χρησιμοποιώντας την συνάρτηση learning_curve από το sklearn.model_selection.

Βήμα 14
Υπολογίστε τις a-priori πιθανότητες για κάθε κατηγορία (class priors).

Υπολογίζω τις apriori ως την τιμή τον αριθμό των δειγμάτων για κάθε ψηφίο ως προς τον αριθμό των ψηφίων.

Βήμα 15
α) Ταξινομήστε όλα τα ψηφία των test δεδομένων ως προς τις 10 κατηγορίες χρησιμοποιώντας τις τιμές της μέσης τιμής και διασποράς που υπολογίσατε στο Βήμα 9(α), υλοποιώντας έναν Naive Bayesian ταξινομητή[2]. Μην χρησιμοποιήσετε έτοιμες υλοποιήσεις. ***Η υλοποίηση σας πρέπει να είναι συμβατή με το scikit-learn όπως δείξαμε στο Βήμα 12.***
β) Υπολογίστε το σκορ για το Βήμα 15(α).
γ) Συγκρίνετε την υλοποίηση σας του Naive Bayes με την υλοποίηση του scikit-learn (GaussianNB).

α) Διαφορετικές υλοποιήσεις για το .pynb και το lib.py

β) Χρησιμοποιώντας το accuracy score υπολογίζω το σκορ του προηγούμενο βήματος 0.82

γ) Συγκρίνοντας τις αποδόσεις ο custom_NaiveBayes ταξινομητής συμπεριφέρεται καλύτερα κατά 14%

Βήμα 16

Επαναλάβατε το Βήμα 15(α), (β) υποθέτοντας ότι η διασπορά για όλα τα χαρακτηριστικά, για όλες τις κατηγορίες ισούται με 1.

Για τον ίδιο αριθμό var για όλα τα samples η απόδοση του ταξινομητή ήταν καλύτερη γεγονός αναμενόμενο. Με διαφορετικό var έχω διαφορετικές προβλέψεις κάθε φορά. Όσο μειώνεται το var τόσο καλύτερες οι προβλεψεις.

Βήμα 17

Συγκρίνετε την επίδοση των ταξινομητών Naive Bayes, Nearest Neighbors, SVM (με διαφορετικούς kernels). Μπορείτε να χρησιμοποιήσετε τις υλοποιήσεις του scikit-learn.

Στην υλοποίηση των αλγορίθμων μπορούμε να παρατηρήσουμε τις διαφορετικές επιδόσεις των ταξινομητών.
Για το Nearest Neighbors παρατηρούμε ότι όσο αυξάνεται ο αριθμός των γειτόνων τόσο καλύτερη η απόδοση του ταξινομητή.

Βήμα 18

Η βασική ιδέα του βήματος αυτού είναι ο συνδυασμός κάποιων ταξινομητών με αρκετά καλή επίδοση με στόχο να επιτευχθεί επίδοση υψηλότερη των επιμέρους επιδόσεων. Αυτή η τεχνική είναι γνωστή ως ensembling. Είναι σημαντικό οι ταξινομητές που θα συνδυαστούν να χαρακτηρίζονται από διαφορετικό τύπο λαθών, π.χ., ο ένας ταξινομητής να τείνει να ταξινομεί λάθος το ψηφίο 3, ενώ ένας άλλος να τείνει να ταξινομεί λάθος το ψηφίο 7.
α) Επιλέξτε κάποιους από τους ταξινομητές που χρησιμοποιήσατε στα προηγούμενα βήματα. Χρησιμοποιήστε το VotingClassifier του scikit-learn για να τους συνδυάσετε σε hard ή soft voting. Αυτός ο μετα-ταξινομητής συνδυάζει τους επιμέρους ταξινομητές βάζοντάς τους να ψηφίσουν για το αποτέλεσμα. Πρέπει να επιλέξετε μονό αριθμό ταξινομητών. Γιατί;
β) Επιλέξτε έναν ταξινομητή από τα προηγούμενα βήματα και χρησιμοποιήστε τον BaggingClassifier για να δημιουργήσετε ένα ensemble. Η τεχνική bagging, αφορά στο χωρισμό του ,training) dataset σε τυχαία υποσύνολα (με πιθανές επικαλύψεις) και την εφαρμογή ενός ταξινομητή σε κάθε ένα από αυτά. Η τελική απόφαση βγαίνει μέσω ψηφοφορίας ή μέσου όρου των προβλέψεων των επιμέρους ταξινομητών. Ο συνδυασμός αυτής της τεχνικής με Decision Trees μας δίνει τον ταξινομητή Random Forest.
γ) Σχολιάστε τα αποτελέσματα.

α) Χρησιμοποιήσαμε την επιλογή των παραπάνω ταξινομητών καθώς είχαν την βέλτιστη απόδοση στα προηγούμενα ερωτήματα. Ο μονός αριθμός ταξινομητών επιτυγχάνει την αποφυγή του να έχω ισοπαλίες στα score τους.