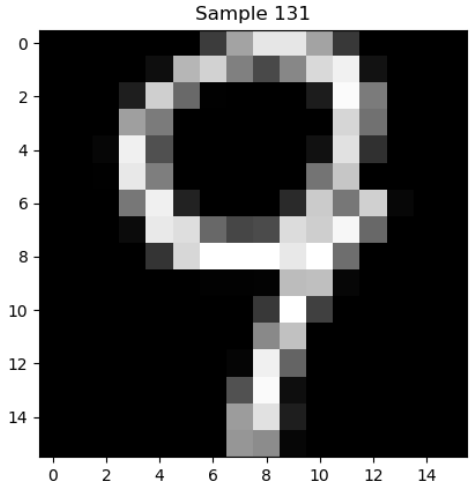
Βήμα 1

Αρχικά, φορτώθηκαν τα δεδομένα εκπαίδευσης (training set) και τα δεδομένα αξιολόγησης (test set), μαζί με τις ετικέτες τους. Συνολικά διαθέτουμε 7291 εικόνες για εκπαίδευση και 2007 για αξιολόγηση, κάθε μια εκ των οποίων είναι διάνυσμα 256 χαρακτηριστικών.

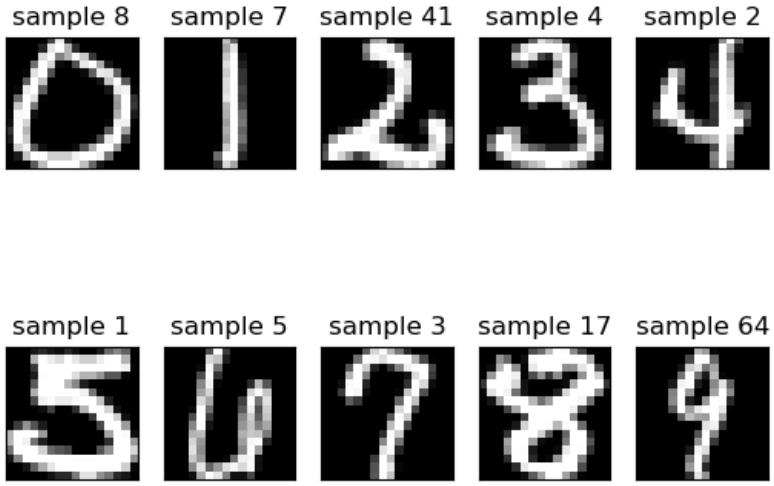
Βήμα 2

Παρακάτω απεικονίζεται το δείγμα 131 από τα δεδομένα εκπαίδευσης:



Βήμα 3

Παρακάτω φαίνονται 10 δείγματα ψηφίων, το πρώτο που συναντάμε από κάθε κατηγορία:



Βήμα 4

Η μέση τιμή του pixel (10,10) για το ψηφίο “0” προέκυψε -0.504, αθροίζοντας την αντίστοιχη τιμή του κάθε δείγματος από το σύνολο εκπαίδευσης και διαιρώντας με το πλήθος των δειγμάτων.

Βήμα 5

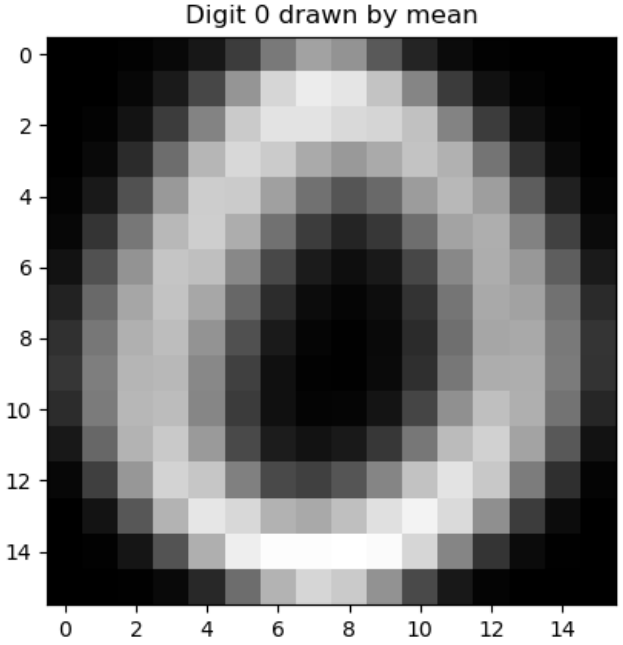
Η διασπορά του pixel (10,10) για το ψηφίο “0” προέκυψε 0.525, αθροίζοντας τη διαφορά της αντίστοιχης τιμής του κάθε δείγματος από το σύνολο εκπαίδευσης από τη μέση τιμή, στο τετράγωνο, διαιρώντας με το πλήθος των δειγμάτων.

Βήμα 6

Με τον ίδιο τρόπο, υπολογίσαμε για κάθε pixel τη μέση τιμή και τη διασπορά για το ψηφίο “0”.

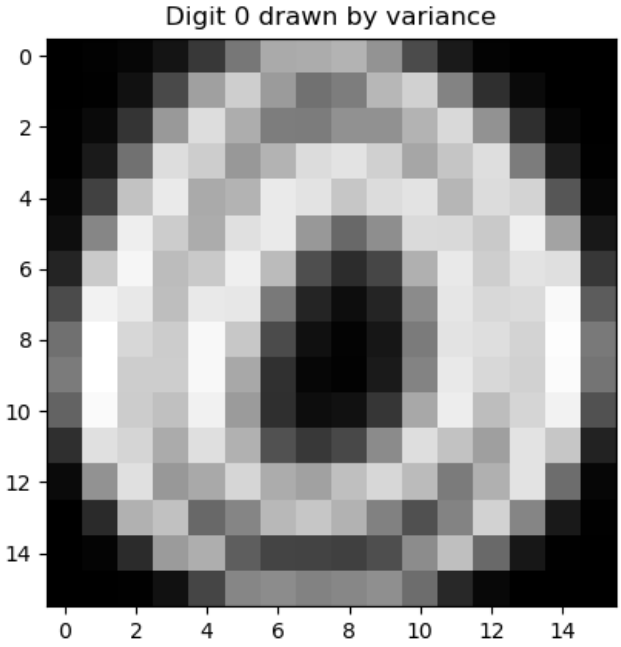
Βήμα 7

Παρακάτω φαίνεται το ψηφίο “0”, σχεδιασμένο με βάση τις μέσες τιμές, που υπολογίσαμε στο βήμα 6.



Βήμα 8

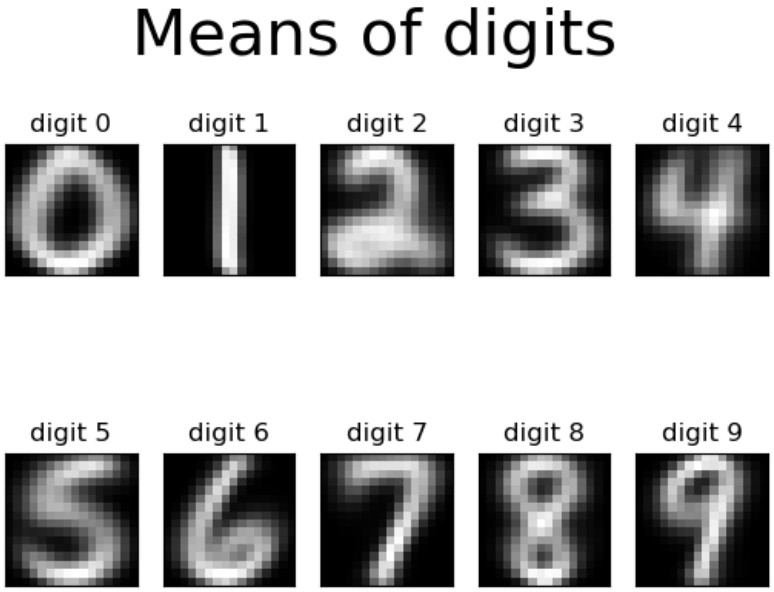
Παρακάτω φαίνεται το ψηφίο “0”, σχεδιασμένο με βάση τις διασπορές, που υπολογίσαμε στο βήμα 6.

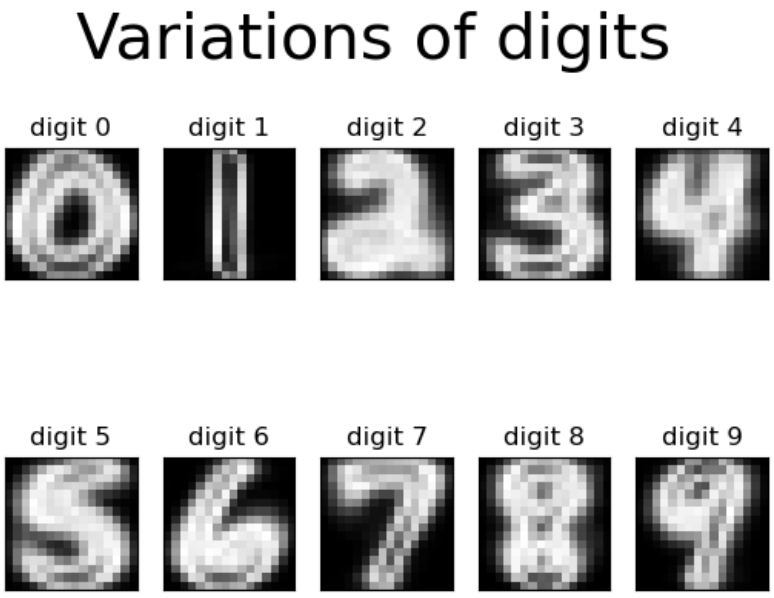


Παρατηρούμε ότι σχεδιάζοντας το ψηφίο με οποιονδήποτε από τους δύο τρόπους, μπορούμε να διακρίνουμε για ποιο ψηφίο πρόκειται. Η εικόνα της μέσης τιμής φαίνεται γενικά να έχει λιγότερο θόρυβο.

Βήμα 9

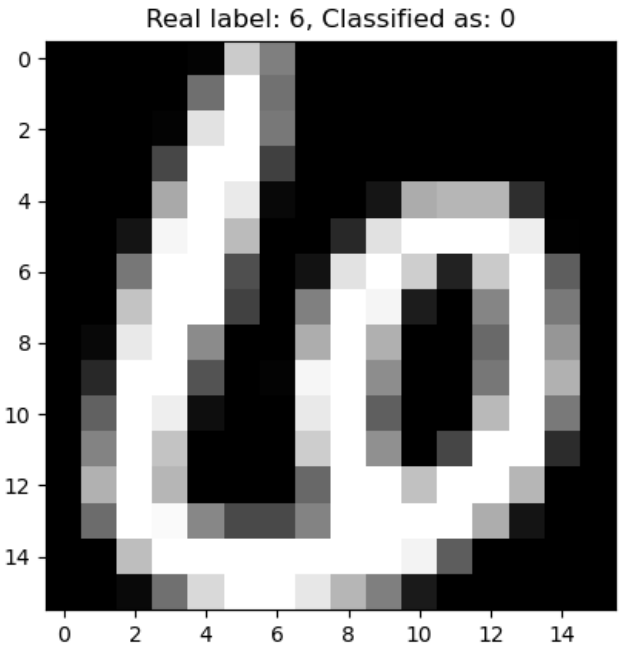
Ακολουθήσαμε την ίδια διαδικασία με το βήμα 6 για όλα τα ψηφία και παρακάτω τα παραθέτουμε σχεδιασμένα με βάση τη μέση τιμή και τη διασπορά.





Βήμα 10

Για το βήμα αυτό, δημιουργήσαμε συνάρτηση που υπολογίζει την ευκλείδια απόσταση μεταξύ δύο εικόνων και διαλέξαμε την κλάση, από την οποία το δείγμα 101 (του συνόλου αξιολόγησης) απέχει τη μικρότερη ευκλείδια απόσταση από τη μέση τιμή της. Παρακάτω φαίνεται το δείγμα 101, το οποίο ανήκει στην κλάση 6, ενώ με βάση την ευκλείδια απόσταση κατηγοριοποιήθηκε στην κλάση 0.



Βήμα 11

Με τον ίδιο τρόπο, κατηγοριοποιήθηκαν όλα τα δείγματα του test set και το ποσοστό επιτυχίας που προέκυψε ηταν 0.814 ή 81.4%.

Βήμα 12

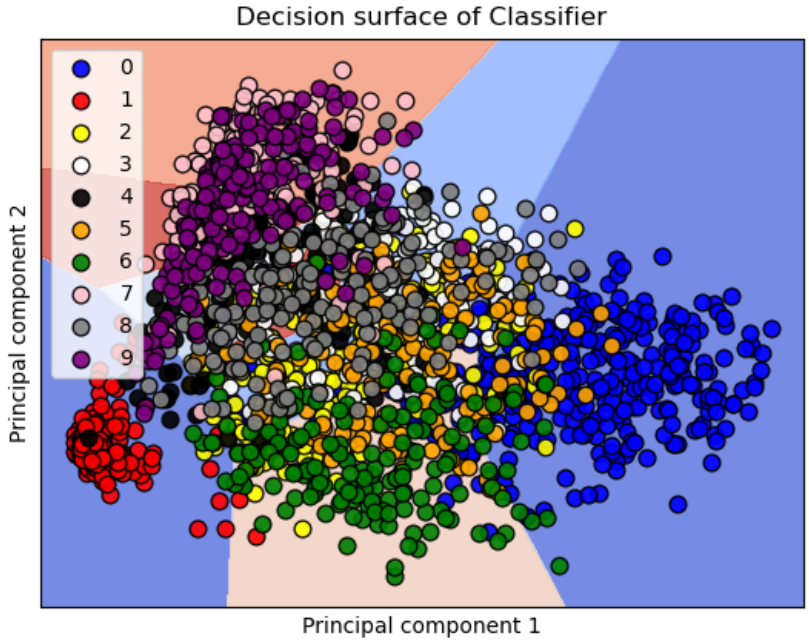
Στη συνέχεια, υλοποιήσαμε τον ευκλείδιο ταξινομητή σε μορφή συμβατή με το πακέτο scikit-learn, δηλαδή ένα class με μεθόδους fit, predict και score. Η σωστή λειτουργία του ταξινομητή επαληθεύτηκε συγκρίνοντας την ακρίβειά του με το προηγούμενο ερώτημα (81.4%).

Η μέθοδος fit παίρνει το training set, δηλαδή τα δείγματα εκπαίδευσης με τα labels τους και υπολογίζει για κάθε label, τη μέση τιμή του.
Η μέθοδος predict παίρνει το test set, δηλαδή τα δείγματα αξιολόγησης με τα labels τους και υπολογίζει για κάθε δείγμα, την κλάση από την οποία απέχει την ελάχιστη απόσταση.
Η μέθοδος score παίρνει το test set και μέσω της predict, υπολογίζει την ακρίβεια κατηγοριοποίησης του ταξινομητή με βάση τις προβλέψεις των labels του test set.

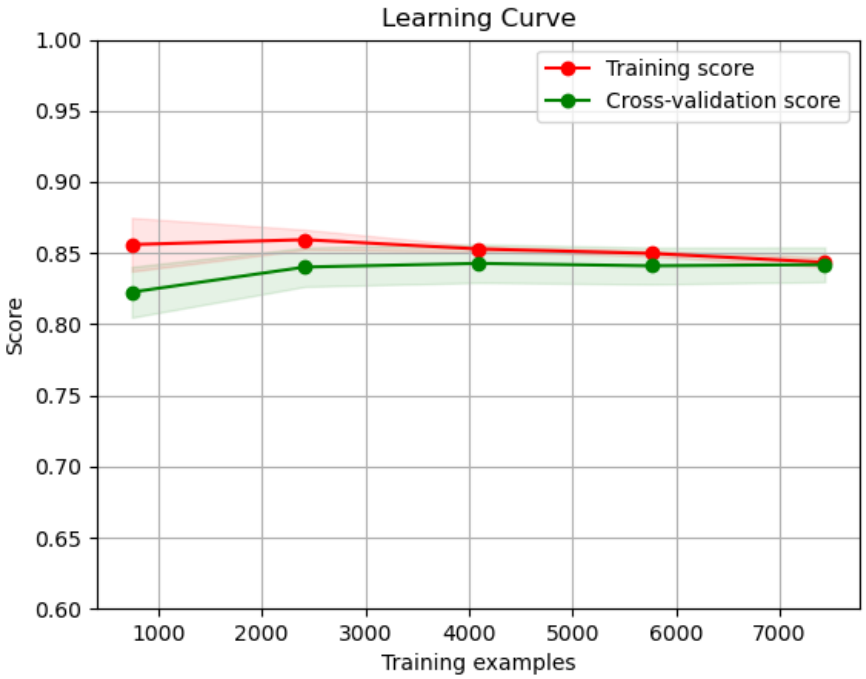
Βήμα 13

α) Για την αξιολόγηση του παραπάνω ταξινομητή, χρησιμοποιήσαμε την τεχνική του cross validation. Η τεχνική αυτή παίρνει ολόκληρο το σύνολο των δεδομένων (training και test set) και το χωρίζει σε ένα συγκεκριμένο αριθμό παρτίδων. Ένα από αυτά τα μέρη επιλέγεται σαν test set και τα υπόλοιπα σαν training set και αξιολογείται η ακρίβεια του ταξινομητή με βάση αυτό το split. Έπειτα, επιλέγεται σαν test set μια διαφορετική παρτίδα και σαν training set οι υπόλοιπες, έως ότου να δοκιμαστούν όλοι οι δυνατοί συνδυασμοί. Τελικά, σαν αποτέλεσμα της αξιολόγησης δίνεται ένας συνδυασμός (συνήθως ο μέσος όρος) της ακρίβειας κατηγοριοποίησης του κάθε συνδυασμού training και test set. Σκοπός αυτής της τεχνικής, είναι να καταδείξει προβλήματα όπως overfitting και selection bias και να αποτιμήσει την ανεξαρτησία του μοντέλου από το dataset. Στο συγκεκριμένο πείραμα, επιλέξαμε τον αριθμό των παρτίδων να είναι 5 και το score του ευκλείδιου ταξινομητή από την τεχνική cross validation προέκυψε:
84.1% ±1.4%.

β) Για το σχεδιασμό της περιοχής απόφασης του ευκλείδιου ταξινομητή στο επίπεδο, χρειάστηκε η μείωση της διαστατικότητας των δεδομένων από 256 σε 2. Για το σκοπό αυτό, χρησιμοποιήσαμε την τεχνική PCA από το πακέτο scikit-learn και ο ευκλείδιος ταξινομητής κατηγοριοποίησε τα 2Δ πλέον δεδομένα, όπως φαίνεται στο παρακάτω γράφημα:



γ) Τέλος, σχεδιάσαμε την καμπύλη εκμάθησης του μοντέλου μας, όπως φαίνεται παρακάτω, όπου παρατηρούμε οτι το score του training set και του cross validation σταθεροποιείται περίπου στα 5000 δείγματα, ενώ μετά τα 7000 το training score πέφτει, γεγονός που σηματοδοτεί πιθανό overfitting.



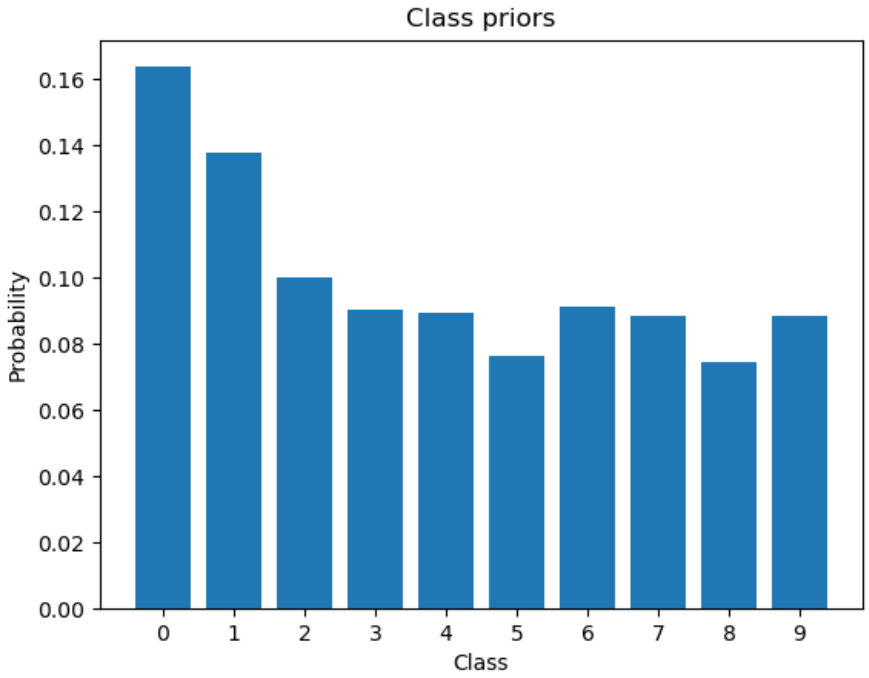
Τα παρακάτω βήματα αποτελούν το κύριο μέρος του εργαστηρίου.

Βήμα 14

Οι a-priori πιθανότητες για κάθε κλάση (class priors) υπολογίζονται από το λόγο του πλήθους δειγμάτων εμφάνισης της κάθε κλάσης προς το συνολικό αριθμό δειγμάτων:

$$p(c_i) = \frac{N_i}{N}$$

Παρακάτω φαίνεται η κατανομή των δειγμάτων του training set μας:



Βήμα 15

α) Στο συγκεκριμένο βήμα το ζητούμενο ήταν η υλοποίηση ενός Naive Bayes ταξινομητή, ο οποίος θα ταξινομεί κάθε στοιχείο του συνόλου αξιολόγησης σε κάποια από τις δέκα κλάσεις.
Ο ταξινομητής το επιτυγχάνει αυτό επιλέγοντας την κλάση στην οποία έχει την μεγαλύτερη πιθανότητα να ανήκει το εκάστοτε στοιχείο του test set. Το κάθε στοιχείο του test set αποτελείται από 256 χαρακτηριστικά και ο κανόνας του Bayes, εφαρμόζεται ως εξής για κάθε στοιχείο:

$$\text{prediction} = argmax(P(y_i|x_1, x_2, ..., x_{256})), \quad i = 1, 2, ..., 10$$

όπου:

$$P(y_i|x_1, x_2, ..., x_{256}) = P(x_1|y_i) \cdot P(x_2|y_i) \cdots P(x_{256}|y_i) \cdot P(y_i)$$

Στον παραπάνω τύπο οι a-priori πιθανότητες των y_i υπολογίζονται στο βήμα 14, ενώ για τις εξαρτημένες πιθανότητες της μορφής $P(x_i|y_i)$ θεωρούμε ότι κάθε κλάση ακολουθεί μία κανονική γκαουσιανή κατανομή με συγκεκριμένη μέση τιμή και διασπορά (την υπολογίσαμε σε προηγούμενο ερώτημα) και έτσι μπορούμε να την υπολογίσουμε από τον τύπο:

$$P(x_i|y_i) = P(x_i) \sim N(\mu_i, \sigma_i))$$

όπου τα μ_i, σ_i είναι η αντίστοιχη μέση τιμή και διασπορά της κλάσης y_i.

Β)

Το score για το βήμα 15 α) το υπολογίζουμε ως τον απλό μέσο των σωστών προβλέψεων που έκανε ο classifier επάνω στο test set, δηλαδη:

$$score = \frac{N_{\text{correct}}}{N_{\text{total}}}$$

Για το δοσμένο test set, το score ήταν 62.4%.

Γ)

Χρησιμοποιώντας την υλοποιημένη συνάρτηση για τον naive bayes από το scikit learn εκπαιδεύσαμε έναν naive bayes classifier με το ίδιο train set και επάνω στο ίδιο test set επιτύγχανε σκορ 71.9%.
Επίσης οι δύο ταξινομητές για 5 fold cross validation επάνω στο ίδιο train set επιτυγχάνουν σκορ 64.8% ± 1.8% και 74.8% ± 2.3% αντίστοιχα. Επομένως και χωρίς να είχαμε στην κατοχή μας το test set θα επιλέγαμε τον ταξινομητή της scikit learn ως αυτόν με το καλύτερο σκορ.

Βήμα 16

Στο συγκεκριμένο βήμα, και για να θέσουμε την διασπορά ίση με 1 για όλα τα pixel θέσαμε μια boolean παράμετρο στην συνάρτηση fit της κλάσης CustomNBClassifier, όπου ανάλογα την τιμή της μας λέει αν η διασπορά είναι πάντα ίση με 1 ή όχι. Έπειτα εκπαιδεύοντας ξανά τον ταξινομητή με το ίδιο σετ και μετρώντας το σκορ παρατηρούμε μεγαλύτερα αποτελέσματα σε σύγκριση με την προηγούμενη υλοποίηση του ταξινομητή. Συγκεκριμένα πετυχαίνουμε σκορ 81.3%. Αυτό πιθανόν να οφείλεται στο γεγονός ότι οι μέσες τιμές της κάθε κλάσης βρίσκονται κάθε φορά πολύ κοντά στη σωστή πρόβλεψη και η εκάστοτε διασπορά δημιουργεί μεγαλύτερες αποκλίσεις που δεν οφελούν στην σωστή πρόβλεψη.

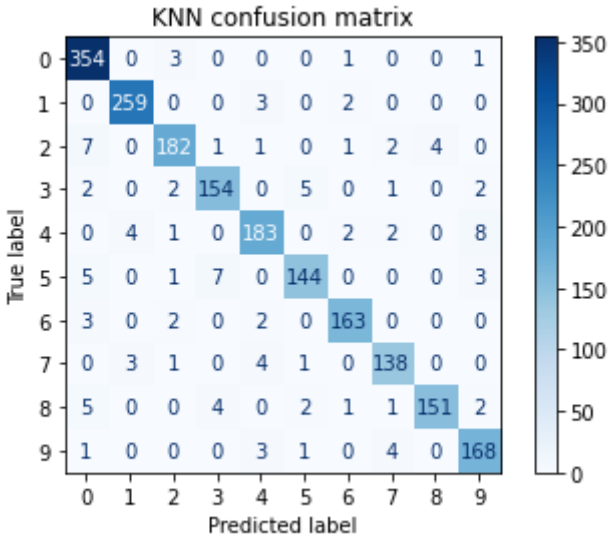
Βήμα 17

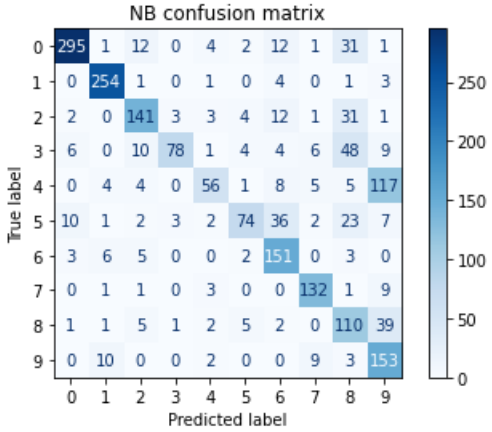
Στο συγκεκριμένο βήμα χρησιμοποιήσαμε τις έτοιμες υλοποιήσεις της βιβλιοθήκης scikit learn για τους ταξινομητές SVM και k nearest neighbours. Και οι δύο ταξινομητές πέτυχαν υψηλότερα σκορ από τον naive bayes.
Πιο συγκεκριμένα, ο ταξινομητής k nearest neighbours για $k = 5$ είχε σκορ 94.5%.

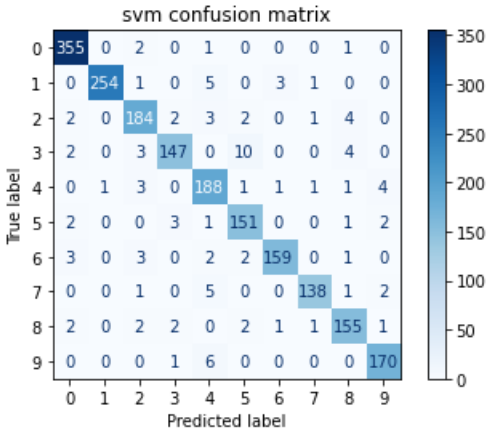
Παρόμοια αποτελέσματα είχε και ο ταξινομητής SVM, ο οποίος υλοποιήθηκε με δύο διαφορετικούς πυρήνες, με το σκορ να φτάνει κοντά στο 94.7% . Για τον συγκεκριμένο ταξινομητή χρησιμοποιήθηκε η μέθοδος σύγκρισης ένας εναντίον ενός, σε ότι αφορά την σύγκριση των κλάσεων μεταξύ τους, αντί της μεθόδου ένας εναντίον πολλών, καθώς αυτή η μέθοδος ενδείκνειται για classification μεταξύ πολλών κλάσεων.
Και οι δύο ταξινομητές εκπαιδεύτηκαν και δοκιμάστηκαν στα ίδια δεδομένα, όπως και ο naive bayes. Όπως φαίνεται, αυτοί οι δύο ταξινομητές πετυχαίνουν μεγαλύτερα score από τον naive bayes και αυτό οφείλεται στο ότι η θεώρηση ότι η πιθανότητα του κάθε pixel της εικόνας να ανήκει σε κάποια κλάση ακολουθεί κανονική κατανομή, είναι λανθασμένη.

Βήμα 18

α) Αρχικά, σε αυτό το βήμα για να επιλέξουμε ποιους ταξινομητές θα χρησιμοποιήσουμε για να τους συνδυάσουμε, χρειάστηκε να σχεδιάσουμε τον confusion matrix του κάθε ταξινομητή. Για να συνδιαστούν βέλτιστα οι ταξινομητές θέλω να τους συνδυάσω με τρόπο τέτοιο ώστε ο κάθε ταξινομητής να τείνει να σφάλει σε διαφορετικές κλάσεις της ταξινόμησης και έτσι συνδυαστικά ο ένας να διορθώνει τα σφάλματα του άλλου. Παρακάτω φαίνονται αυτοί οι πίνακες:







Έτσι μετά από αυτό το αρχικό βήμα οι ταξινομητές που συνδυάσαμε ήταν ο KNN μαζί με τον SVM, αλλά και τον naive bayes, καθώς ο καθένας έσφαλε για διαφορετικά ψηφία της ταξινόμησης. Ο τελικός ταξινομητής συνδύασε τους επιμέρους ταξινομητές με την τεχνική του hard voting και για τον λόγο αυτό απαιτείται μονός αριθμός ταξινομητών για να μην προκύπτουν ισοπαλίες.

β) Για την δημιουργία του bagging ταξινομητή χρησιμοποιήσαμε από τους αρχικούς ταξινομητές τον SVM.
Αυτός εφαρμόστηκε σε 10, όχι απαραίτητα ξένα μεταξύ τους, υποσύνολα του train set και δημιούργησε το τελικό μοντέλο.

γ) Και για τους δύο παραπάνω μέτα-ταξινομητές παρατηρούμε ότι το σκορ δεν αυξήθηκε αισθητά σε σχέση με τους υλοποιημένους ταξινομητές KNN και SVM, κάτι όμως που θεωρείται φυσιολογικό καθώς το σκορ ήταν ήδη πολύ υψηλό (κοντά στο 95%).
Αυτό που αναμένεται από τους μέτα-ταξινομητές είναι σε περιπτώσεις πιο ασθενών εκτιμητών, να καταφέρνουν να τους συνδυάζουν έτσι ώστε να αξιοποιούνται τα θετικά χαρακτηριστικά του κάθε ένα και να ελαχιστοποιούνται τα σφάλματα.

Βήμα 19

Στο τελευταίο βήμα, ασχοληθήκαμε με χρήση νευρωνικών δικτύων μέσω PyTorch για το πρόβλημα κατηγοριοποίησής μας.

α) Αρχικά, υλοποιήσαμε μια κλάση που κληρονομεί από την Dataset της PyTorch και φέρνει τα δεδομένα μας σε μορφή συμβατή με έναν DataLoader. Ο DataLoader κλήθηκε με ορίσματα για μέγεθος batch δειγμάτων και ένα flag για shuffle των δεδομένων.

β) Στη συνέχεια, υλοποιήσαμε μια κλάση LinearWActivation, η οποία κληρονομεί από την nn.Module και είναι ουσιαστικά η υλοποίηση ενός fully connected layer του νευρωνικού. Παίρνει την είσοδο του layer, εφαρμόζει τη γραμμική πράξη με τα weights και περνάει το αποτέλεσμα από τη συνάρτηση ενεργοποίησης.

Επίσης, υλοποιήσαμε μια κλάση για τη δημιουργία του ολοκληρωμένου μοντέλου του νευρωνικού δικτύου, η οποία παίρνει ως είσοδο όλες τις υπερπαραμέτρους, όπως το πλήθος των layers, το πλήθος των neurons και τη συνάρτηση ενεργοποίησης.
Για να γίνει η κατηγοριοποίηση θέτουμε ένα αριθμό epochs, κάνουμε forward pass τα batches ένα-ένα, υπολογίζουμε το σφάλμα και εφαρμόζουμε το back propagation με τον optimizer. Στην περίπτωσή μας το σφάλμα είναι το cross-entropy, ενώ ο optimizer το Stochastic Gradient Descent. Έτσι, κάνουμε update τις παραμέτρους (weights, biases) του νευρωνικού σε κάθε βήμα και υλοποιούμε ουσιαστικά το training.

Για το training επιλέξαμε 30 epochs, batch size ίσο με 32 και πειραματιστήκαμε με διαφορετικούς συνδυασμούς των υπερπαραμέτρων. Στον παρακάτω πίνακα, συνοψίζουμε τα αποτελέσματα πάνω στο test set, για όλους τους συνδυασμούς:

Hidden layers	Neurons	Activation function	Accuracy
1	50	ReLU	91.2%
1	50	tanh	91.0%
1	100	ReLU	91.2%
1	100	tanh	90.5%
2	100, 50	ReLU	90.9%
2	100, 50	tanh	89.2%
2	128, 64	ReLU	91.3%
2	128, 64	tanh	90.8%

Παρατηρούμε ότι με αύξηση του πλήθους των layers και των neurons, η ακρίβεια μένει ίδια ή και μειώνεται σε κάποιο μικρό βαθμό, το οποίο μπορεί να οφείλεται σε overfitting πάνω στο training set.
Σε ό,τι αφορά τις συναρτήσεις ενεργοποίησης, η ReLU φαίνεται να δίνει γενικά λίγο καλύτερα αποτελέσματα, γεγονός που πιθανό να σημαίνει καλύτερο match με το συγκεκριμένο dataset και πρόβλημα.

γ) Αρχικά, το training set χωρίστηκε σε training και validation, ώστε να μπορεί να γίνει αξιολόγηση του μοντέλου χωρίς να επεμβαίνουμε στο test set. Στη συνέχεια,

υλοποιήσαμε την κλάση του νευρωνικού δικτύου σε μορφή συμβατή με το scikit-learn, δηλαδή με μεθόδους constructor (init), fit και score. Στην init δίνουμε όλες τις υπερπαραμέτρους του μοντέλου καθώς και δημιουργούμε το ίδιο το μοντέλο που θα χρησιμοποιούν οι επόμενες μέθοδοι. Στην fit γίνεται το training, καθώς και μια αξιολόγηση πάνω στο validation set, ενώ στη score γίνεται η αξιολόγηση του μοντέλου πάνω σε test δεδομένα, με τρόπο παρόμοιο με το προηγούμενο ερώτημα.
Η ακρίβεια στο validation set είναι 92.7% .

δ) Για την αξιολόγηση πάνω στο test set, κλήθηκε η score και έδωσε ακρίβεια 88.7% .