2 Η ΥΠΟΧΡΕΩΤΙΚΗ ΕΡΓΑΣΙΑ ΣΤΟ ΜΑΘΗΜΑ «ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ – ΒΑΘΙΑ ΜΑΘΗΣΗ»

Ονοματεπώνυμο : Απόστολος Γιαννουλίδης

ΑΕΜ : 2906

Η υλοποίηση της εργασίας έγινε στην γλώσσα προγραμματισμού Python 3. Η εργασία αφορά την κατηγοροποίηση των δεδομένων από την Mnist βάση δεδομένων σε μονούς και ζυγούς αριθμούς .

Επιλογή kernel και Παραμέτρων :

Έγινε σύγκριση ανάμεσα σε τρεις διαφορετικούς πυρήνες (linear , polynomial ,rbf) σε δείγμα του αρχικού συνόλου .
Για να μπορέσουμε όμως να έχουμε σαφή εικόνα για την απόδοση του rbf προηγήθηκε αναζήτηση για τις τιμές των c και gamma .

Αναζήτηση των C , gamma :

Η αναζήτηση των τιμών για τις μεταβλητές αυτές έγινε σε ένα τυχαίο υποσύνολο από το σύνολο των δεδομένων που διαθέτουμε και εξετάστηκαν όλοι οι συνδιασμοί τιμών για τα C ίσο με (0.01,0.1,1,10) και gamma ισο με (0.001,0.01,0.1,1,10) . Ο κώδικας για την αναζήτηση βρίσκεται στο αρχείο **searchGandC.py**

Αποτελέσματα εκτέλεσης :

Ακρίβεια , C , gamma

0.4875 C = 0.010000 , G = 0.001000
0.495 C = 0.010000 , G = 0.010000
0.5125 C = 0.010000 , G = 0.100000
0.4975 C = 0.010000 , G = 1.000000
0.5225 C = 0.010000 , G = 10.000000
0.8625 C = 0.100000 , G = 0.001000
0.89 C = 0.100000 , G = 0.010000
0.6 C = 0.100000 , G = 0.100000
0.515 C = 0.100000 , G = 1.000000
0.49 C = 0.100000 , G = 10.000000
0.885 C = 1.000000 , G = 0.001000
0.9 C = 1.000000 , G = 0.010000
0.85 C = 1.000000 , G = 0.100000
0.51 C = 1.000000 , G = 1.000000
0.51 C = 1.000000 , G = 10.000000
0.9125 C = 10.000000 , G = 0.001000
0.935 C = 10.000000 , G = 0.010000
0.9 C = 10.000000 , G = 0.100000
0.585 C = 10.000000 , G = 1.000000
0.4975 C = 10.000000 , G = 10.000000

Παρατηρούμε πως για τις τιμές C ίσο με 10 και Gamma ίσο με 0.01 έχουμε την καλύτερη απόδωση .

Εφόσον επιλέξαμε τα c και gamma προχωράμε στο επόμενο στάδιο της επιλογής του καλύτερου πυρήνα .

Στο αρχείο **testDiffSvmKernels.py** βρίσκται ο κώδικας όπου συγκρίνονται οι τρεις διαφορετικοί πυρήνες .

Αποτελέσματα :

linear	Test acc: 0.881000	Train acc: 0.892200	Time : 12.828125
poly	Test acc: 0.978000	Train acc: 0.995200	Time : 2.015625
rbf	Test acc: 0.991000	Train acc: 0.998400	Time : 1.390625

Παρατηρούμε πως ο linear kernel είναι ταυτόχρονα και πιο αργός αλλα και έχει μικρότερη ακρίβεια . Παράλληλα ο πολυωνυμικός και ο rbf να φτάνουν στο 99% με μικρή διαφορά στον χρόνο .
Γίνεται η επιλογή του rbf .

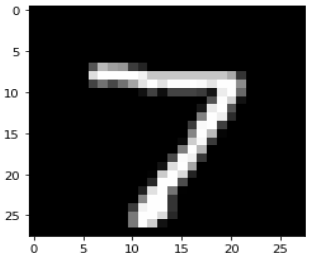
Εφαρμογή του πυρηνά σε όλο το σύνολο των δεδομένων και σύγκριση με άλλες μεθόδους :

Αρχικά μειώνουμε την διάσταση των δεδομένων *(λόγο των περιορισμένων διαθέσιμων πόρων)* με την εφαρμογή του PCA αλγορίθμου κρατώντας το 90% της πληροφορίας . Έπειτα προπονούμε το svm με πυρήνα rbf και παραμέτρους C και gamma στα δεδομένα μας .

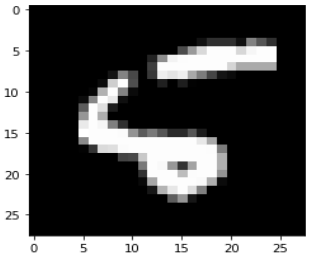
Ακρίβεια και χρόνος εκτέλεσης *(ο χρόνος συμπεριλαμβάνει και τον χρόνο του υπολογισμού της απόδοσης)* :
Test acc: 0.990600
Train acc: 0.996817
Time : 196.265625

Λανθασμένες και Σωστές προβλέψεις (αρχείο **predections.py**):

```
Prediction :  1
Real value :  1
```



```
Prediction :  0
Real value :  1
```



Το μοντέλο πρόβλεψε σωστά πως το 7 είναι μονός αριθμός όμως δεν μπόρεσε να προβλέψει πως το 5 είναι μονός αριθμός

Παρατήρηση : ενώ στην πραγματικότητα είναι 5 ο αριθμός στην δεύτερη φωτογραφία στην πραγματικότητα μοιάζει αρκετά με 6 το οποίο είναι ζυγός .

Σύγκριση με τους κατηγοριοποιητής πλησιέστερων γειτώνων και πλησιέστερου κέντρου κλάσης:

Αποτελέσματα από την εκτέλεση του αρχείου **svmTry.py :**
(Σημείωση ο αλγόριθμος KNeighborsClassifier όπως και ο NearestCentroid , ουσιαστικά δεν εκτελεί κάποιο training το οποίο σημαίνει πως ο χρόνος προπόνησης του μοντέλου είναι μηδενικός , όμως αντίθετα ο χρόνος για να μετρήσουμε την απόδοση του και γενικότερα να κάνουμε πρόβλεψη σε κάποιο μεγάλο σύνολο δεδομένων είναι αρκετά μεγάλός . Για την σύγκριση θα χρησιμοποιηθεί ο χρόνος περιλαμβανομένου και τον χρόνο που χρειαζόμαστε για την αξιολόγησή του .)

Svm:
Test acc: 0.990600
Train acc: 0.996817
Time : 193.140625

KNeighborsClassifier k=3:
Train acc k=3 = 0.99375
Test acc k=3 = 0.9867
Time : 470.515625

KNeighborsClassifier k=1:
Train acc k=1 = 1.0
Test acc k=1 = 0.9855
Time : 63.359375

NearestCentroid
Train acc = 0.8080166666666667
Test acc = 0.8018
Time : 0.09375

Παρατηρούμε πως οι τρεις πρώτη έχουν πολύ καλή ακρίβεια με καλύτερο μοντέλο το svm με μικρή διαφορά , όμως χρονικά ο *KNeighborsClassifier με k=1* πετυχένει 98% ακρίβεια σε πολύ ληγότερο χρόνο από το svm.