Γιώργος Μπισιρόπουλος ΑΜ:2771

Αναγνώριση προτύπων - 1η Σειρά ασκήσεων

Files:

- kNearNeighbor.py
- naiveBayes.py
- neuralNet.py
- svm.py
- m_loader.py :
  - περιέχει βοηθητικές συναρτήσεις που υλοποίησα για το διάβασμα των δεδομένων από τα csv αρχεία και τον χωρισμό σε folds για το cross validation
- m_metrics.py :
  - περιέχει βοηθητικές συναρτήσεις που υλοποίησα για τον υπολογισμό των accuracy, f1 score. Τελικά όμως οι αντίστοιχες μέθοδοι sklearn.metrics επέστρεφαν με λίγο μεγαλύτερη ακρίβεια τα αποτελέσματα των διαιρέσεων με πολλά δεκαδικά ψηφία.

Για τα κυρίως files:

Αρχικά προσπάθησα να υλοποιήσω τις μεθόδους kNN & naiveBayes κλπ από το μηδέν, αλλά κυρίως λόγω τον πολλών features του spam dataset χρειάστηκε να χρησιμοποιήσω μερικές συναρτήσεις από την βιβλιοθήκη sklearn και να επικεντρωθώ στην δοκιμή διάφορους συνδυασμών παραμέτρων.

Προεπεξεργσία:

Επιπλέον για τα Instances του Spambase Dataset, λόγω του εύρους τιμών χρειάστηκε να κάνω pre-processing κάνοντας χρήση της μεθόδου StandardScaler για κάθε feature προκειμένου να βελτιώσω την απόδοση (Neural Networks ,SVM) και παράλληλα μειώθηκε και ο χρόνος εκπαίδευσης.

Comparison of pre-processing methods for SVM

	avg_accuracy	avg_f1_score
StandardScaler	0.9282758	0.9076301
MinMaxScaler	0.9063235	0.8751266

Όταν προσπάθησα να κάνω την ίδια σύγκριση και στα Neural Networks όταν χρησιμοποιούσα την MinMaxScaler κατέληγα σε no true samples οπότε και σε συνδυασμό με την προηγούμενη παρατήρηση επέλεξα να χρησιμοποιήσω και στις 2 περιπτώσεις την StandardScaler η οποία εν συντομία κάνει scaling με βάση την τυπική απόκλιση και τη μέση τιμή.

Spambase (4601 Instances)

Spambase Dataset			
Method	Accuracy	F1 score	Notes
kNN (k=1)	0.823955151	0.775932496	
kNN (k=4)	0.833093627	0.7870823192	
kNN (k=13)	0.826793064	0.7774852699	
kNN (k=31)	0.815056269	0.7630664477	
Naive Bayes	0.822215073	0.8090704735	
Neural Networks (1HL: K1 = 150)	0.902848547	0.8723060307	invscaling with INIT 0.1 & MAX_ITER 600
Neural Networks (2HL)	0.607259939	0.006551567	Even with K = (150,200) no true samples in 60% of tests
SVM (linear kernel)	0.928716734	0.9082235648	
SVM (rbf kernel)	0.934148712	0.9146153797	

(Τιμές όχι ποσοστά % αλλά μεταξύ 0.0 – 1.0)

Σχόλια:

- Καθώς αυξάνω το k στον αλγόριθμο kNN παρατήρησα μείωση του accuracy, (και F1 score) αλλά για k=4 είχα ικανοποιητικά αποτελέσματα. Επίσης στο occupancy dataset με λιγότερα feautures είχε μεγαλύτερο accuracy.

- Για τα Neural Networks με 1 hidden layer μετά από δοκιμές κατέληξα να επιλέξω ως **learning_rate** : invscaling, **learning_rate_init:** 0.1 το οποίο δηλαδή ξεκινάει με μεγάλη τιμή παραμέτρου και σιγά σιγά με το χρόνο μειώνετε προκειμένου να μην γίνονται τόσες μεγάλες ενημερώσεις στα weights καθώς περνάει ο χρόνος το οποίο είχε μεγαλύτερο accuracy από constant learning rate.

- Για τα Neural Networks με 2 hidden layer παρά τις δοκιμές με διάφορες τιμές παραμέτρων σε όλες τις εκτελέσεις υπήρχαν αρκετές επαναλήψεις με no true samples (όλα τα test data classified ως 0) με αποτέλεσμα πολλά False-Negative, αλλά σχεδόν κανένα True-Positive το οποίο (εκτός από το λάθος classification) δεν μας επιτρέπει να υπολογίσουμε το F1-score (ill-defined) παρόλα αυτά τα έχω προσθέσει στον πίνακα.

- Για τα SVM με rbf kernel μετά από δοκιμές κατέληξα να επιλέξω ως gamma = auto και γενικά διαλέγοντας μικρότερες τιμές αυξάνουμε το accuracy και το F1- score και καλύτερη γενικευτική ικανότητα.

Συμπέρασμα:

Μπορούμε να κατατάξουμε τις μεθόδους σε αύξουσα σειρά με βάση accuracy

Naive Bayes << kNN (k=4) << Neural Networks (w 1HL) << << SVM (linear kernel) << SVM (rbf kernel) [best results]

Για Occupancy Detection (8143 Instances)

Από τα 3 dataset επέλεξα να χρησιμοποιήσω το datatraining.txt

Occupancy Detection Dataset			
Method	Accuracy	F1 score	Notes
kNN (k=1)	0.99029875	0.97716486	
kNN (k=4)	0.98968420	0.97592563	
kNN (k=13)	0.99042085	0.97788478	
Naive Bayes	0.97789541	0.95045270	
Neural Networks (1HL: K1 = 150)	0.92472315	0.84093929	constant with INIT 0.01 & MAX_ITER 700 ConvergenceWarning & no true samples with invscaling
Neural Networks 2HL: (150, 100)	0.92963293	0.85374354	constant with INIT 0.01 & MAX_ITER 700
SVM (linear kernel)	0.98624877	0.96835783	Slow
SVM (rbf kernel)	0.95443645	0.902705049	Much faster than linear kernel

Συμπέρασμα:

- Για το συγκεκριμένο dataset καλύτερο performance είχε ο kNN (k=4), και μετά SVM (linear kernel) που όμως χρειάστηκε αρκετά λεπτά περισσότερο από όλες τις υπόλοιπες μεθόδους και σε συνδυασμό με το γεγονός ότι ο Naive Bayes μας δίνει αρκετά ικανοποιητικά αποτελέσματα, που είναι αμέσως μετά για το συγκεκριμένο dataset.

- Γενικά για το συγκεκριμένο dataset όλες οι μέθοδοι σημείωσαν υψηλότερα f1-score (& accuracy) ακόμα και μετά από μείωση των Instances σε 2665 χρησιμοποιώντας το datatest.txt,
    - πχ accuracy:(kNN : 0.9846, Naive Bayes: 0.96249, Neural Networks (1HL:0.978985)
        - άρα συμπεραίνουμε ότι δεν οφείλετε στον μεγαλύτερο αριθμό Instances αλλά στην μορφή των δεδομένων (dimensionality, deviation)

- SVM (rbf kernel) έκαναν το classification αρκετά γρηγορότερα από SVM (linear kernel) αλλά με χαμηλότερο accuracy και f1-score (using default values for c, g)

<u>Γενικές παρατηρήσεις για τις μεθόδους:</u>

- kNN, naive Bayes είχαν σχεδόν σταθερό χρόνο ανεξαρτήτου dataset ή μορφής των δεδομένων, αριθμό features. Και για τα instances με μικρό dimensionality επιστρέφουν αρκετά πιο accurate results.
- Τα Neural Networks μας επιτρέπουν να κάνουμε αρκετούς διαφορετικούς συνδυασμούς παραμέτρων (hidden layers, activation function, learning rate, solver) , αλλά αν δεν γίνει προσεκτική επιλογή μπορεί να οδηγήσει σε αρκετά μειωμένο performance.
- Τα SVM είχαν σταθερά καλύτερη απόδοση και στα δυο dataset από τις υπόλοιπες μεθόδους, (Μετά και τα Neural Networks)