ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2019-2020

ΜΥΕ002 - ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ

ΔΙΔΑΣΚΩΝ: ΜΠΛΕΚΑΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΑΝΑΦΟΡΑ ΠΡΩΤΗΣ ΕΡΓΑΣΙΑΣ

ΓΚΙΤΣΑΚΗΣ ΔΗΜΟΣ Α.Μ. :2425

ΚΡΟΜΜΥΔΑΣ ΓΕΩΡΓΙΟΣ Α.Μ.:3260

ΙΩΑΝΝΙΝΑ, 2020

ΕΙΣΑΓΩΓΗ

Στην παρούσα αναφορά αναλύεται η υλοποίηση 5 μεθόδων ταξινόμησης δεδομένων στα πλαίσια του μαθήματος της Αναγνώρισης Προτύπων. Για τις μεθόδους αυτές χρησιμοποιήθηκαν 2 διαφορετικά datasets, το spam dataset και το credit card clients dataset. Για την υλοποίηση των μεθόδων χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python, καθώς και κάποιες βιβλιοθήκες της. Είναι σημαντικό να αναφερθεί πως λόγω των συνθηκών έχουμε δουλέψει στους προσωπικούς μας υπολογιστές και κάποιες από τις βιβλιοθήκες που έχουμε χρησιμοποιήσει, ενδέχεται να μην είναι εγκατεστημένες στους υπολογιστές των εργαστηρίων του τμήματος.

ΜΕΡΟΣ Α: ΠΡΟΕΤΟΙΜΑΣΙΑ ΔΕΔΟΜΕΝΩΝ

Πριν την υλοποίηση των μεθόδων χρησιμοποιήσαμε την στρατηγική 10-folds cross validation προκειμένου να χωρίσουμε τα δεδομένα σε 10 ομοιόμορφα υποσύνολα. Η διαδικασία αυτή χρησιμοποιήθηκε και για τα 2 datasets ξεχωριστά. Για την στρατηγική αυτή, δημιουργήσαμε την συνάρτηση organiseFolds, η οποία παίρνει σαν παράμετρο το όνομα του αρχείου που περιέχει το σύνολο δεδομένων που θέλουμε να χωρίσουμε. Για τον διαχωρισμό του συνόλου χρησιμοποιούμε λεξικά 10 θέσεων και κάθε θέση τους δείχνει σε μία λίστα με ένα υποσύνολο (fold). Για να πετύχουμε έναν «δίκαιο» διαχωρισμό κάναμε χρήση της mod(pos, N), όπου pos ο αριθμός της γραμμής του παραδείγματος στο αρχικό αρχείο και N το 10. Ο αριθμός που επιστρέφεται από το mod είναι ο αριθμός του fold στο οποίο κατατάσσεται το παράδειγμα. Με αυτόν τον τρόπο πετυχαίνουμε ίδιο ποσοστό από κάθε κατηγορία σε κάθε fold, καθώς τα αρχεία είναι ταξινομημένα ως προς τις κατηγορίες και κάθε fold δεν παίρνει συνεχόμενες γραμμές του αρχείου, αλλά μία ανά 10. Ουσιαστικά, κάθε fold έχει μέγεθος το 1/10 του αρχικού αρχείου αλλά το ποσοστό των κατηγοριών είναι ίδιο.

ΣΗΜΕΙΩΣΗ: Το αρχείο που περιείχε το credit card clients dataset ήταν της μορφής excel, αλλά αντιμετωπίσαμε ένα θέμα κατά το άνοιγμα του στην Python και για αυτό το λόγο το μετατρέψαμε σε csv και χρησιμοποιούμε αυτό. Το csv αρχείο συμπεριλαμβάνεται στα παραδοτέα.

ΜΕΡΟΣ Β: ΥΛΟΠΟΙΗΣΗ ΜΕΘΟΔΩΝ

Είναι σημαντικό να αναφερθεί ότι όλες οι μέθοδοι έχουν υλοποιηθεί σε συναρτήσεις και οι κλήσεις των συναρτήσεων αυτών έχουν μπει σε σχόλια ώστε να τρέχουμε κάθε φορά την μέθοδο που θέλουμε βγάζοντας την από σχόλιο και όχι όλες μαζί, κάτι που εξοικονομεί αρκετό χρόνο. Αρχικά, όλες οι κλήσεις των συναρτήσεων είναι σε σχόλια κάτω από την εκάστοτε συνάρτηση.

Β.1: ΠΑΡΑΛΛΑΓΗ ΤΗΣ ΜΕΘΟΔΟΥ LVQ

Για την μέθοδο αυτή δημιουργήσαμε την συνάρτηση LVQ(), η οποία παίρνει σαν παράμετρο το όνομα του αρχείου που θέλουμε να χρησιμοποιήσουμε. Στην συνέχεια, αφού ορίζει τα training_set και test_set, αρχικοποιεί τα κέντρα των κατηγοριών ως το μέσο όρο των χαρακτηριστικών των παραδειγμάτων των 2 κατηγοριών. Επομένως, τα κέντρα (centers) είναι διανύσματα μεγέθους ίσου με τον αριθμό των χαρακτηριστικών των παραδειγμάτων. Ως ακτίνα της κάθε κατηγορίας, αρχικά χρησιμοποιούμε την διακύμανση των παραδειγμάτων κάθε κατηγορίας. Έπειτα, ξεκινά η διαδικασία του training για την οποία διαλέγουμε ένα τυχαίο παράδειγμα από το σύνολο του training_set και για αυτό υπολογίζουμε την ομοιότητα σε σχέση με την κάθε κατηγορία σύμφωνα με την Γκαουσιανή συνάρτηση ομοιότητας. Επομένως, υπολογίζονται 2 τιμές που κάθε μία από αυτές αντιστοιχεί σε μία κατηγορία και η μεγαλύτερη τιμή υποδεικνύει και την νικήτρια κατηγορία. Στην συνέχεια γίνεται έλεγχος αν η νικήτρια κατηγορία είναι και η σωστή. Η σωστή κατηγορία μας είναι γνωστή εξαρχής, αφού έχουμε μάθηση με επίβλεψη. Αν η νικήτρια κατηγορία είναι και η σωστή, έχουμε επιτυχία και επιβραβεύουμε το σύστημα μετατοπίζοντας το κέντρο της κατηγορίας πιο κοντά στο τρέχον παράδειγμα και μεγαλώνοντας την ακτίνα της. Αν η νικήτρια κατηγορία είναι λάθος, τιμωρούμε το σύστημα απομακρύνοντας το κέντρο της από το τρέχον παράδειγμα και μειώνοντας την ακτίνα της. Επιπλέον, αυξάνουμε τον αριθμό των κατηγοριών κατά 1 και η νέα κατηγορία έχει ως κέντρο το τρέχον παράδειγμα και ακτίνα ένα μικρό ποσοστό (20%) της ακτίνας της νικήτριας κατηγορίας. Η παραπάνω διαδικασία γίνεται τόσες φορές όσες και το μέγεθος του training_set. Ένας τέτοιος κύκλος είναι μία εποχή και στην συγκεκριμένη μέθοδο ως εκπαίδευση εκτελούμε 10 εποχές. Αφού ολοκληρωθεί η εκπαίδευση, ξεκινά η διαδικασία του testing, όπου για κάθε παράδειγμα του test_set υπολογίζουμε την ομοιότητα με όλες τις K κατηγορίες που έχουν προκύψει και αφού περάσουμε όλα τα παραδείγματα του test_set, υπολογίζουμε το Accuracy και το F1 Score της μεθόδου για να την αξιολογήσουμε.

ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕΘΟΔΟΥ LVQ

	Accuracy	**F1 Score**
Spambase	62.17%	0.53
Credit card	57.43%	0.28

ΣΗΜΕΙΩΣΕΙΣ

1**)**Η μέθοδος είναι πολύ χρονοβόρα. Ενδεικτικά, για ένα μόλις fold για το spambase σύνολο δεδομένων, μεγέθους 460 γραμμών χρειάζεται πάνω από μισή ώρα προκειμένου να λάβουμε αποτελέσματα. Όπως γίνεται αντιληπτό, είναι χρονικά αδύνατο να το τρέξουμε για 9 folds σαν σύνολο δεδομένων 10 φορές για να βγάλουμε το μέσο όρο. Για αυτό το λόγο τα παραπάνω αποτελέσματα προκύπτουν μετά από την εκτέλεση της μεθόδου για training_set = fold0 και test_set = fold9 (spambase) και training_set = credit_fold0 και test_set = credit_fold9 (credit card) .

2) Μετά από διευκρινίσεις που δόθηκαν σε διάλεξη του μαθήματος, κανονικοποιήσαμε τα δεδομένα του $2^{ου}$ σετ (credit card) διαιρώντας τις τιμές κάθε στήλης με την μεγαλύτερη, μειώσαμε τα χαρακτηριστικά του κάθε παραδείγματος που χρησιμοποιούμε στην εκπαίδευση και αυξήσαμε ελαφρώς το σ_{init}. Αυτό είχε ως αποτέλεσμα την κατακόρυφη βελτίωση των αποτελεσμάτων της μεθόδου για το $2^{ο}$ σετ και μια μικρή βελτίωση για το $1^{ο}$ σετ. Τα κανονικοποιημένα δεδομένα του credit_card είναι τα norm_credit_fold0, norm_credit_fold1, … , norm_credit_fold9, ενώ τα μη κανονικοποιημένα είναι τα credit_fold0, credit_fold1, … , credit_fold9.

ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕΘΟΔΟΥ LVQ (ΜΕ ΚΑΝΟΝΙΚΟΠΟΙΗΜΕΝΟ ΤΟ CREDIT CARD)

	Accuracy	**F1 Score**
Spambase	66.3%	0.63
Credit card	99.2%	0.98

Β.2: k-NN Nearest Neighbors με Ευκλείδεια Απόσταση

Για την μέθοδο αυτή δημιουργήσαμε την συνάρτηση nearestNeighbors(), η οποία δέχεται σαν παραμέτρους έναν αριθμό k που υποδηλώνει τον αριθμό τον k γειτόνων που θέλουμε καθώς και ένα filename που αντιστοιχεί στο αρχείο για το οποίο θέλουμε να εφαρμόσουμε την μέθοδο. Έπειτα, για κάθε παράδειγμα του αρχείου, υπολογίζουμε την ευκλείδεια απόσταση από τα υπόλοιπα παραδείγματα και αποθηκεύουμε τις αποστάσεις στην λίστα distances. Στην συνέχεια, ταξινομούμε την λίστα αυτή σε αύξουσα σειρά και κρατάμε τις k πρώτες αποστάσεις που είναι και οι k μικρότερες. Τέλος, κρατάμε 2 μετρητές που μας δείχνουν πόσα από τα k παραδείγματα είναι της κατηγορίας 0 και πόσα της κατηγορίας 1. Το παράδειγμά μας ταξινομείται στην κατηγορία του μεγαλύτερου μετρητή και υπολογίζουμε τα Accuracy και F1 Score για όλα τα παραδείγματα.

ΣΗΜΕΙΩΣΗ

Η μέθοδος, αντίστοιχα με αυτή της LVQ, είναι αρκετά χρονοβόρα και για αυτό το λόγο για τα αποτελέσματα που παρουσιάζονται στον παρακάτω πίνακα χρησιμοποιούμε ένα υποσύνολο του κάθε dataset και συγκεκριμένα το fold0 για το spambase σύνολο δεδομένων και το credit_fold0 για το credit card σύνολο δεδομένων.

ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕΘΟΔΟΥ k-NN Nearest Neighbor

	Accuracy	**F1 Score**
Spambase	80.04%	0.74
Credit card	73.57%	0.27

B.3: Neural Networks

Για την μέθοδο αυτή υλοποιήσαμε την συνάρτηση NeuralNetworks(), η οποία παίρνει σαν παραμέτρους το filename που είναι το όνομα του αρχείου το οποίο θέλουμε να ταξινομήσουμε, το hidden που είναι ο αριθμός των κρυμμένων επιπέδων που θα έχει το νευρωνικό δίκτυο και το trainingMethod που είναι η μέθοδος η οποία χρησιμοποιείται για την εκπαίδευση του δικτύου. Έτσι, η συνάρτηση καλείται 8 φορές, 4 για κάθε αρχείο, από 2 με Gradient Descent και από 2 με Stochastic Gradient Descent, με 1 και 2 κρυμμένα επίπεδα η κάθε μία. Στην συνάρτηση, χωρίζουμε τις σωστές κατηγορίες από τα δεδομένα εκπαίδευσης και τα υπόλοιπα τα εισάγουμε στο νευρωνικό δίκτυο. Το δίκτυο μας υλοποιείται με συναρτήσεις της βιβλιοθήκης PyTorch. Όλοι οι νευρώνες του δικτύου έχουν σιγμοειδή συνάρτηση ενεργοποίησης και ο αριθμός των νευρώνων είναι 70 όταν χρησιμοποιούμε 1 κρυμμένο επίπεδο ενώ όταν έχουμε 2 κρυμμένα επίπεδα, 60 και 30 αντίστοιχα. Για τον υπολογισμό του σφάλματος χρησιμοποιούμε την συνάρτηση MSELoss με παράμετρο “sum” για να υπολογίσει αθροιστικό τετραγωνικό σφάλμα και όχι μέσο τετραγωνικό σφάλμα. Για την μέθοδο εκπαίδευσης χρησιμοποιούμε την συνάρτηση optim.SGD με ρυθμό εκπαίδευσης ίσο με 0.001. Η διαφορά μεταξύ Gradient Descent και Stochastic Gradient Descent είναι ότι στην πρώτη χρησιμοποιούμε όλα τα δεδομένα για μία εποχή, ενώ στην δεύτερη μόνο ένα υποσύνολο 50 παραδειγμάτων που διαλέγονται τυχαία. Για αυτό το λόγο οι εποχές στην $2^{η}$ μέθοδο είναι πολύ περισσότερες (30 και 3000 αντίστοιχα).

Μετά το πέρας της εκπαίδευσης εισάγουμε στο δίκτυο το test_set και με βάση τα αποτελέσματα τα οποία δίνει υπολογίζουμε τα Accuracy και F1 Score.

ΣΗΜΕΙΩΣΕΙΣ

1)Πολλές φορές υπάρχει μια δυσκολία το δίκτυο να εντοπίσει TruePositives και για αυτό το λόγο το F1 Score βγαίνει ίσο με 0. Δεν καταλήξαμε σε κοινές παραμέτρους και για τα 2 αρχεία ώστε να παίρνουμε καλά αποτελέσματα και για τα 2 σύνολα δεδομένων, παρά το γεγονός ότι το Accuracy είναι υψηλό και για τα 2 σύνολα.

2)Το 2ο σύνολο δεδομένων τις περισσότερες φορές δυσκολεύεται να πέσει κάτω από ένα τοπικό ελάχιστο με αποτέλεσμα οι περισσότερες εποχές να επιστρέφουν ίδιο σφάλμα.

3)Οι παραπάνω παράμετροι δίνουν ικανοποιητικά αποτελέσματα για το spambase σύνολο δεδομένων με την μέθοδο Stochastic Gradient Descent.

ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕΘΟΔΟΥ Neural Networks

	Accuracy	**F1 Score**
Spambase 1 Hidden Layer Gradient Descent	60.65%	0.00
Spambase 2 Hidden Layers Gradient Descent	60.65%	0.00
Spambase 1 Hidden Layer Stochastic Gradient Descent	75.87%	0.65
Spambase 2 Hidden Layers Gradient Descent	77.61%	0.74
Credit Card 1 Hidden Layer Gradient Descent	77.87%	0.00
Credit Card 2 Hidden Layers Gradient Descent	77.87%	0.00

Credit Card 1 Hidden Layer Stochastic Gradient Descent	77.87%	0.00
Credit Card 2 Hidden Layers Stochastic Gradient Descent	77.87%	0.00

B.4: Support Vector Machines (SVM)

Για την μέθοδο αυτή έχουμε υλοποιήσει την συνάρτηση SVM(), η οποία δέχεται την παράμετρο fileName που υποδηλώνει το όνομα του αρχείου το οποίο περιέχει τα παραδείγματα που θέλουμε να ταξινομήσουμε καθώς και την παράμετρο kernel, η οποία υποδηλώνει τον τύπο του kernel που θέλουμε να έχουμε (Linear ή Gaussian). Για την μέθοδο αυτή χρησιμοποιούμε συναρτήσεις της βιβλιοθήκης sklearn και πιο συγκεκριμένα, αφού δηλώσουμε training set και test set, την συνάρτηση SVC για την δημιουργία της μεθόδου με Linear και Gaussian Kernel ανάλογα με την παράμετρο που δίνεται στην κλήση της συνάρτησης, καθώς και την συνάρτηση fit για το πέρασμα των δεδομένων του training set. Μετά από αυτό, ελέγχουμε την αποδοτικότητα της μεθόδου με την συνάρτηση predict που παίρνει σαν παράμετρο το test set. Τέλος, τυπώνουμε τα Accuracy, F1 Score και confusion matrix που δίνεται έτοιμος από την sklearn.

ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕΘΟΔΟΥ SVM

	Accuracy	**F1 Score**
Spambase Linear Kernel	93.48%	0.95
Spambase Gaussian Kernel	70.87%	0.79
Credit Card Linear Kernel	77.87%	0.88
Credit Card Gaussian Kernel	77.87%	0.88

B.5: Naïve Bayes Classifier

Για την μέθοδο αυτή υλοποιήσαμε την συνάρτηση NaïveBayes(), η οποία παίρνει σαν παράμετρο το fileName που αντιστοιχεί στο όνομα του αρχείου που περιέχει τα παραδείγματα που θέλουμε να ταξινομήσουμε. Για την υλοποίηση της μεθόδου χρησιμοποιήσαμε συναρτήσεις της βιβλιοθήκης sklearn και πιο συγκεκριμένα, μετά τον καθορισμό του training set και test set, χρησιμοποιήσαμε τη GaussianNB().fit για να δημιουργήσουμε την μέθοδο με ανεξάρτητες κατανομές και του περνάμε σαν παράμετρο το training set και έπειτα ελέγχουμε την αποδοτικότητα της μεθόδου με την συνάρτηση predict που παίρνει σαν παράμετρο το test_set. Τέλος, υπολογίζουμε και τυπώνουμε τα Accuracy, F1 Score και τον confusion matrix που δίνεται έτοιμος από την sklearn.

ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕΘΟΔΟΥ Naïve Bayes Classifier

	Accuracy	**F1 Score**
Spambase	80.87%	0.82
Credit Card	43.10%	0.46

ΜΕΡΟΣ Γ: ΣΥΜΠΕΡΑΣΜΑΤΑ

Συγκρίνοντας τα αποτελέσματα των μεθόδων η αποδοτικότερη μέθοδος με βάση το F1 Score για το spambase σύνολο δεδομένων είναι η SVM με Linear Kernel με F1 Score = 0.95, ενώ για το credit cards σύνολο δεδομένων η καλύτερη μέθοδος είναι το LVQ αλλά με κανονικοποιημένα δεδομένα. Χωρίς την κανονικοποίηση, η καλύτερη μέθοδος είναι η SVM είτε με Linear είτε με Gaussian Kernel με F1 Score = 0.88.