ΜΥ002: ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ

ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ 1 – Αλγόριθμοι Ταξινόμησης (ή κατηγοριοποίησης) δεδομένων

Ονοματεπώνυμο: Νίκη Αριστειδοπούλου, Νικόλαος Βόσιος
ΑΜ: 2937, 1643

Επεξεργασία αρχείων

Στην παρούσα σειρά ασκήσεων δόθηκαν 2 αρχεία για να γίνουν τα πειράματα. Το ένα είναι το *spam dataset* και το άλλο είναι το *credit card clients dataset*. Πριν ξεκινήσουν τα προγράμματα πρέπει να γίνει μια επεξεργασία στα αρχεία, οπότε πρώτα πρέπει να τρέξει το πρόγραμμα *make_the_file.py*. Το συγκεκριμένο πρόγραμμα διαβάζει τα αρχεία που πρέπει να κάνουμε πειράματα (*spambase.data*, *default of credit card clients.csv*). Συγκεκριμένα, στο *spambase.data* τοποθετεί στην πρώτη γραμμή τα χαρακτηριστικά x και την κατηγορία, τέλος το γράφει σε ένα νέο αρχείο με το όνομα *spambase.csv*. Στο αρχείο *default of credit card clients.csv* βγάζει την πρώτη γραμμή, διότι δεν χρειάζονται οι πληροφορίες της πρώτης γραμμής και την πρώτη στήλη γιατί δε χρειάζεται και αυτή (έχει το id της κάθε γραμμής), τέλος το γράφει σε ένα νέο αρχείο με το όνομα *default of credit card clients.csv*.

Αλγόριθμος

Στην αρχή αρχικοποιεί τις μεταβλητές που θα χρειαστούν. Την *a* ποσό μεταβολής των παραμέτρων, *lenoffolds* τα fold που χρειάζονται, *M* ο ακέραιος αριθμός, *TP* το True Positive, *TN* το True Negative, *FP* το False Positive, *FN* το False Negative, *P* το Positive, *N* το Negative και τον πίνακα *results* με τα αποτελέσματα των μεθόδων. Έπειτα, δημιουργεί μια συνάρτηση στην οποία αρχικοποιεί το μέσον $\boldsymbol{\mu_j}$ και την ακτίνα $\boldsymbol{\sigma_j}$. Διαβάζει το αρχείο που θα χρησιμοποιήσει στον αλγόριθμο και το μετατρέπει σε numpy array. Ταξινομεί σύμφωνα με την κατηγορία, διότι θα πρέπει τα δεδομένα να χωριστούν σε κάθε fold με την ίδια αναλογία. Έπειτα, φτιάχνει τα 10 folds που θα χρησιμοποιηθούν. Επειδή τα παραδείγματα δεν χωράνε ακριβώς σε 10 folds, τότε το τελευταίο fold θα έχει όσα μένουν. Τέλος, φτιάχνει τα δεδομένα που θα χρειαστούν οι υπόλοιπες μέθοδοι ταξινόμησης.

Πρώτα θα τρέξει τις μεθόδους ταξινόμησης και τέλος αυτή που δημιουργήσαμε. Η πρώτη μέθοδος που θα τρέξει είναι η **Nearest Neighbor k-NN**. Το *k* το αρχικοποιεί στην αρχή, εκπαιδεύει το σύστημα, τυπώνει και αποθηκεύει το **Accuracy** και το **F1 score**. Η δεύτερη μέθοδος που τρέχει είναι **Neural Network**s. Η συνάρτηση ενεργοποίησης είναι η σιγμοειδής, το κρυμμένο επίπεδο το αρχικοποιεί με το *hidden*, τους νευρώνες τους αρχικοποιεί με *k* και τα εναλλακτικά σχήματα με *solv*. Τέλος, εκπαιδεύει το σύστημα, τυπώνει και αποθηκεύει το **Accuracy** και το **F1 score**. Η τρίτη μέθοδος που θα τρέξει είναι η **Support Vector Machines**. Η συνάρτηση πυρήνα αρχικοποιείται με την μεταβλητή *tyker*. Εκπαιδεύει το σύστημα, τυπώνει και αποθηκεύει το **Accuracy** και το **F1 score**. Η τέταρτη μέθοδος που θα τρέξει είναι η **Naive Bayes classifier**. Εκπαιδεύει το σύστημα, τυπώνει και αποθηκεύει το **Accuracy** και το **F1 score**.

Έπειτα, θα τρέξει την μέθοδο ταξινόμησης που δημιουργήσαμε. Για *M* επαναλήψεις αρχικοποιεί τα $\boldsymbol{\mu_j}$ και την $\boldsymbol{\sigma_j}$. Για κάθε *fold* κρατάει το *test* που είναι ίδιο με την επανάληψη *M*, διότι χρειάζεται να μετρηθεί η επίδοση της μεθόδου και πρέπει να είναι σε διαφορετικό *fold* σε κάθε *M* επανάληψη. Τα υπόλοιπα *fold* εκπαιδεύονται. Για κάθε παράδειγμα που υπάρχει στο *fold* κρατάει τα χαρακτηριστικά *x* και την πραγματική του κατηγορία *y*. Για όλες τις κατηγορίες που υπάρχουν βρίσκει την Γκαουσιανή περιοχή. Οι περιοχές είναι όσες είναι και τα $\boldsymbol{\mu_j}$ και $\boldsymbol{\sigma_j}$ που υπάρχουν στα *fold* που εκπαιδεύονται. Για όλες αυτές τις περιοχές βρίσκει την Γκαουσιανή συνάρτηση ομοιότητας. Η νικήτρια είναι αυτή που θα είναι η μεγαλύτερη και την κρατάει αυτή την περιοχή στη μεταβλητή *catj*. Έπειτα, παίρνει όλα τα δεδομένα του παραδείγματος και βλέπει αν είναι η νικήτρια κατηγορία (*catj*) ίση με την πραγματική (*y*). Αν είναι τότε επιβραβεύει την απόφαση και μεταβάλει τα $\boldsymbol{\mu_j}$ και $\boldsymbol{\sigma_j}$ της περιοχής. Αν δεν είναι τότε τιμωρεί την απόφαση και μεταβάλει τα $\boldsymbol{\mu_j}$ και $\boldsymbol{\sigma_j}$ της περιοχής και δημιουργεί μια καινουργία Γκαουσιανή περιοχή με κέντρο *x* και ακτίνα μια μικρή τιμή $\boldsymbol{\sigma_{init}}$.

Μόλις τελειώσει με την εκπαίδευση, πηγαίνει να μετρήσει την επίδοση της μεθόδου με το *fold* που κράτησε στην αρχή, δηλαδή το *testdata*. Για αυτό το *fold* θα κάνει την ίδια διαδικασία με την εκπαίδευση μόνο που θα κρατήσει τις μεταβλητές για να βρει το **Accuracy** και το **F1 score**. Δηλαδή αν είναι να επιβραβεύσει και αν η

κατηγορία είναι 1 τότε αυξάνει τα *Negative* και *True Negativ*e, αλλιώς είναι η κατηγορία 0 και αυξάνει τα *Positive* και *True Positive*. Αν είναι να τιμωρήσει και αν η κατηγορία είναι 1 τότε αυξάνει τα *Negative* και *False Negative*, αλλιώς είναι η κατηγορία 0 και αυξάνει τα *Positive* και *False Positive*.

Τέλος, βρίσκει το **Accuracy** (*Acc*) και το **F1 score** (*F1*), τα τυπώνει και τα αποθηκεύει. Ο αλγόριθμος τελικά τυπώνει τα αποτελέσματα με αύξουσα σειρά ως προς το F1 score και τυπώνει την βέλτιστη μέθοδο.

Αποτελέσματα Δοκιμών

- **[Method 1]** Nearest Neighbor k-NN

  Για *k = 3* **Accuracy:** 0.951063829787234 και **F1 score:** 0.9514780543074107

- **[Method 2]** Neural Networks

  (a) Για ένα κρυμμένο επίπεδο

  - 5 νευρώνες
    i. Gradient Descent **Accuracy :** 0.6085106382978723 και **F1 score:** 0.4604075199819881
    ii. Stochastic Gradient Descent **Accuracy :** 0.9702127659574468 και **F1 score:** 0.9702410136900619

  (b) Για δύο κρυμμένα επίπεδα

  - 5 νευρώνες
    i. Gradient Descent **Accuracy :** 0.7787234042553192 και **F1 score:** 0.7767049566441664
    ii. Stochastic Gradient Descent **Accuracy :** 0.9744680851063829 και **F1 score:** 0.9745154523572458

- **[Method 3]** Support Vector Machines

  (a) Γραμμική συνάρτηση πυρήνα **Accuracy:** 0.9574468085106383 και **F1 score:** 0.9574046774400675

  (b) Gaussian συνάρτηση πυρήνα **Accuracy:** 0.7148936170212766 και **F1 score:** 0.7148936170212766

- **[Method 4]** Naive Bayes classifier

  Accuracy: 0.8574468085106383και **F1 score:** 0.8591411441290984

- **[Method 5]** Αλγόριθμος

  Για M = 10 **Accuracy:** 0.3913279721799609 και **F1 score:** 0.002848495638241054

Βέλτιστη Μέθοδος

Σύμφωνα με τα αποτελέσματα των μεθόδων και με σύγκριση το **F1 score** κάθε μεθόδου καταλήγουμε στο συμπέρασμα ότι η βέλτιστη μέθοδος για την ταξινόμηση προβλημάτων που αφορούν δύο κατηγορίες είναι η μέθοδος 2 **Neural Networks** (Νευρωνικά Δίκτυα) με *2* κρυμμένα επίπεδα, *5* νευρώνες σε κάθε επίπεδο και *Stochastic Gradient Descent* εναλλακτικό σχήμα.

Βιβλιοθήκες

- pandas (version 1.0.3)
- matplotlib (version 3.2.1)
- numpy (version 1.18.3)
- sklearn (0.22.2.post1)
- math

Οι βιβλιοθήκες που έχει το πρόγραμμα πρέπει να είναι αναβαθμισμένες.