ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

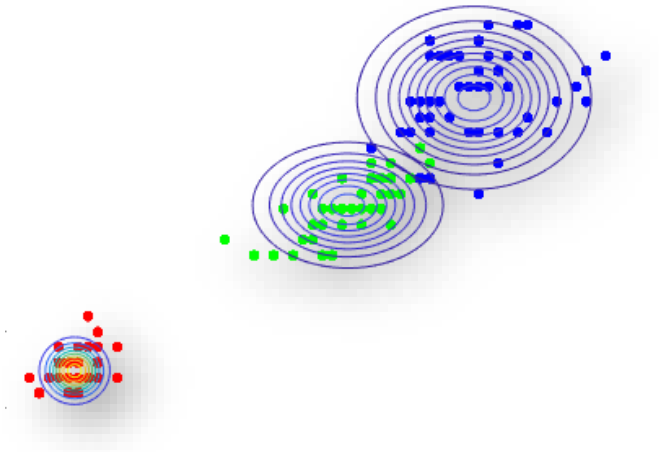
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

Αναγνώριση Προτύπων – Εργασία 1η

Εξέταση της απόδοσης ταξινομητών (kNN, Bayes, Naive Bayes) και χρήση δικτύων Bayes για συμπερασμό πιθανοτήτων



Ον/μο: Μέτσης Γεώργιος

Α.Μ.: Μ1574

Ον/μο: Δοξαστάκης Γεώργιος

Α.Μ.: Μ1613

Περιεχόμενα

Εξέταση της απόδοσης ταξινομητών (kNN, Bayes, Naive Bayes) και χρήση δικτύων Bayes για συμπερασμό πιθανοτήτων

1. ΕΙΣΑΓΩΓΗ

Στα πλαίσια του μαθήματος «Αναγνώριση Προτύπων» ζητήθηκε η υλοποίηση και εξέταση της απόδοσης ταξινομητών κοντινότερων γειτόνων (KNN), Bayes καθώς και Naive Bayes στα γνωστά από τη βιβλιογραφία προβλήματα:

1. ταξινόμησης φυτών Iris ανάμεσα σε 3 είδη
2. ταξινόμηση εγκύων ινδιάνων της φυλής Pima σε άτομα που έχουν ή δεν έχουν διαβήτη

Καθώς και η σχεδίαση και υλοποίηση ενός δικτύου Bayes για κάποιο πρόβλημα επιλογής μας, και στη συνέχεια παρουσίαση παραδειγμάτων συμπερασμού τόσο με ευθύγραμμη όσο και ανάδρομη διάδοση και σύγκριση των αποτελεσμάτων.

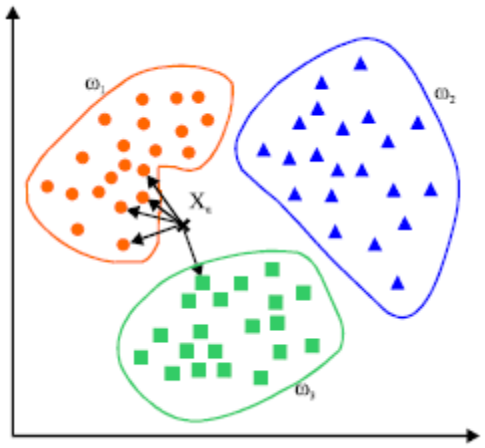
2. ΔΕΔΟΜΕΝΑ ΚΑΙ ΜΕΘΟΔΟΙ

2.1 Ταξινομητές KNN

Οι ταξινομητές Κ κοντινότερων γειτόνων (KNN) αποτελούν μια μη παραμετρική μέθοδο ταξινόμησης.

Τα δείγματα εκπαίδευσης λογίζονται ως διανύσματα σε ένα πολυδιάστατο χώρο και συνοδεύονται από την κλάση στην οποία ανήκουν.

Τα δείγματα ελέγχου ταξινομούνται με βάση μια μετρική απόστασης, και τους ανατίθεται η κλάση που είναι πιο συνήθης μεταξύ των Κ κοντινότερων δειγμάτων εκπαίδευσης. Ο αριθμός Κ είναι θετικός ακέραιος (συνήθως μικρός) και παίρνει τιμές από 1 έως και το πλήθος των δειγμάτων εκπαίδευσης.



Εικόνα 1: *Ταξινόμηση του δείγματος Χ στη κλάση ω_1 με βάση τους 5 κοντινότερους γείτονες.*

Για δείγματά σε συνεχής χώρους συνήθως χρησιμοποιούνται μετρικές όπως η Ευκλείδεια απόσταση, η απόσταση Manhattan , η απόσταση Mahalanobis κ.α. , ενώ για διακριτούς χώρους η απόσταση Hamming, η απόσταση Kulsinski κ.α.

Ένα σημαντικό μειονέκτημα της μεθόδου προκύπτει όταν δείγματα εκπαίδευσης μιας πιο συνήθης κλάσης κυριαρχούν στην απόφαση ταξινόμησης του δείγματος ελέγχου επειδή τείνουν να είναι περισσότερα ανάμεσα στους Κ κοντινότερους γείτονες, λόγω του μεγάλου τους αριθμού. Ένας από τους τρόπους να το αποφύγουμε είναι η χρήση βαρών στην ταξινόμηση, λαμβάνοντας υπόψη την απόσταση του δείγματος ελέγχου από τον καθένα από τους Κ κοντινότερους γείτονες. Η κλάση του κάθε γείτονα πολλαπλασιάζεται με ένα βάρος αντιστρόφως ανάλογο της απόστασής του από το δείγμα ελέγχου.

2.2 Ταξινομητές Bayes

Η οικογένεια ταξινομητών που βασίζονται στην εφαρμογή του θεωρήματος του Bayes για την εκτίμηση της κλάσης ενός δείγματος ονομάζονται ταξινομητές Bayes.

Τα δείγματα εκπαίδευσης συνοδεύονται από τη κλάση στην οποία ανήκουν και έτσι υπολογίζονται οι apriori πιθανότητες $P(C_k)$ καθώς και οι συναρτήσεις πυκνότητας πιθανότητας για κάθε κλάση (class conditional PDF) $p(x|C_k)$.

Έτσι η απόφαση ταξινόμησης εφαρμόζοντας το θεώρημα του Bayes στηρίζεται στην μεγαλύτερη aposteriori πιθανότητα μεταξύ των κλάσεων δεδομένου ενός δείγματος ελέγχου x, $P(C_k \mid x)$:

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

***Εικόνα 2**: A posteriori πιθανότητα της κλάσης C_k δεδομένου του δείγματος X*

και συνοψίζεται στη παρακάτω παράσταση, ακόμα και για δείγματα που ανήκουν σε πολυδιάστατους χώρους:

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

***Εικόνα 3:** Εκτίμηση κλάσης δεδομένου δείγματος X με εφαρμογή του θεωρήματος Bayes*

Με x_i τη τιμή του δείγματος ελέγχου x στη κάθε διάσταση i, n το πλήθος των διαστάσεων του x, C_k τη κλάση και K το πλήθος των κλάσεων.

Ένα από τα προτερήματα των ταξινομητών Bayes είναι πως απαιτούν λίγα δείγματα εκπαίδευσης για να εκτιμήσουν τις παραμέτρους που απαιτούνται για τη ταξινόμηση, αρκεί να είναι αντιπροσωπευτικά του προβλήματος.

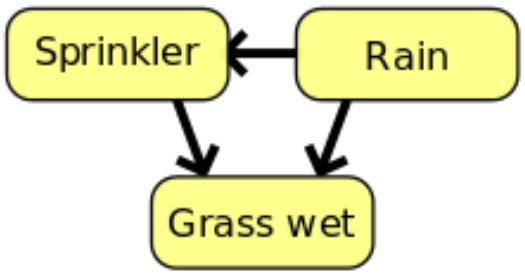
2.3 Ταξινομητές Naive Bayes

Οι απλοϊκοί (naive) ταξινομητές Bayes είναι μια υποπερίπτωση των ταξινομητών Bayes κατά την οποία ο πίνακας συνδιασποράς της κανονικής κατανομής που χρησιμοποιείται για κάθε κλάση είναι μηδενικός σε όλα τα στοιχεία εκτός της διαγωνίου του. Έτσι κατ' απλούστευση τα δείγματα θεωρούνται ανεξάρτητα μεταξύ τους.

Παρά τη παραδοχή ανεξαρτησίας και το υπεραπλουστευμένο μοντέλο οι απλοϊκοί ταξινομητές Bayes αποδίδουν αρκετά αποτελεσματικά σε ένα μεγάλο σύνολο προβλημάτων.

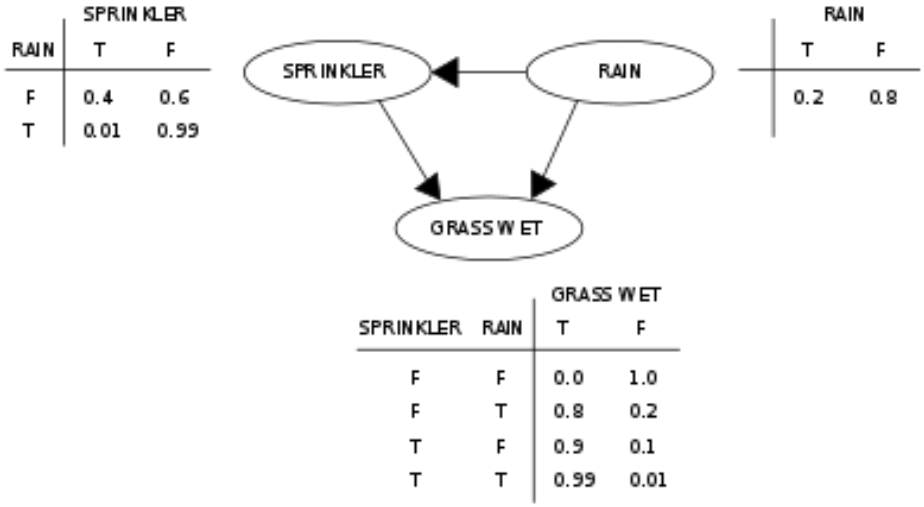
2.4 Δίκτυα Bayes

Τα δίκτυα Bayes είναι πιθανοτικοί κατευθυντηκοί γράφοι που χρησιμοποιούνται για τον συμπερασμό πιθανοτήτων αλληλεξαρτώμενων γεγονότων. Τα χαρακτηριστικά που χρησιμοποιούνται είναι οι κόμβοι του γραφήματος οι οποίοι λαμβάνουν συγκεκριμένες καταστάσεις. της εφαρμογής. Στην εικόνα 4 φαίνεται ένα παράδειγμα ενός απλού δικτύου Bayes.



Εικόνα 4: Παράδειγμα δικτύου Bayes

Τα δίκτυα Bayes εκπαιδεύονται υπολογίζοντας τις πιθανότητες κάθε κατάστασης των κόμβων μέσω διαθέσιμων δεδομένων. Το δίκτυο αφού έχει εκπαιδευτεί όπως φαίνεται στην εικόνα 5, μπορεί να χρησιμοποιηθεί εισάγοντας γνωστές καταστάσεις (evidence) για τον συμπερασμό πιθανοτήτων των επόμενων (ευθεία διάδοση) και των προηγούμενων (ανάδρομη διάδοση) κόμβων.

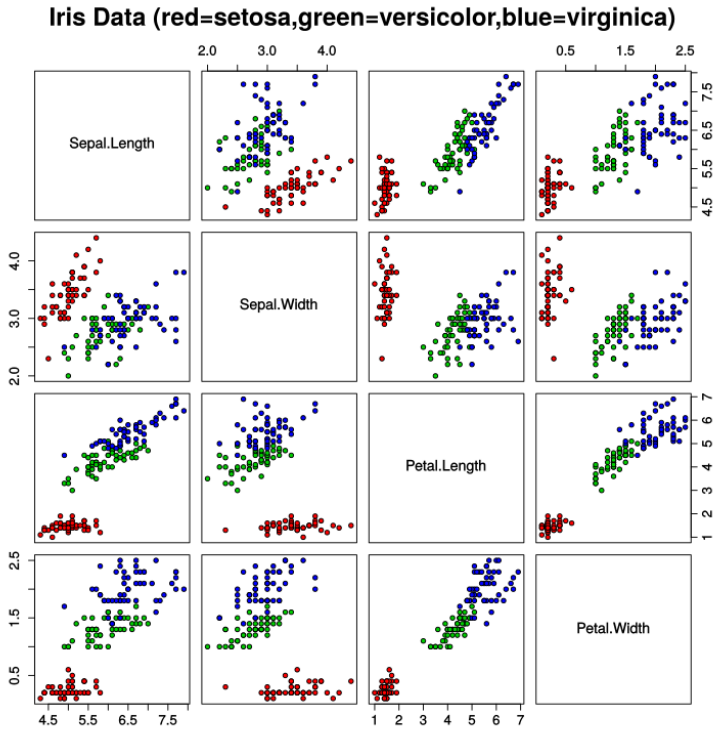


Εικόνα 5: Εκτίμηση πιθανοτήτων σε ένα δίκτυο Bayes

2.5 Σύνολα Δεδομένων

2.5.1 Fisher Iris

Το dataset περιλαμβάνει μετρήσεις μορφολογικών χαρακτηριστικών για τρία είδη ανθών ίριδας. Συνολικά υπάρχουν 50 δείγματα για κάθε είδος (Iris setosa, Iris virginica and Iris versicolor). Το dataset συνήθως χρησιμοποιείται για την δοκιμή στατιστικών τεχνικών μηχανικής μάθησης.



Εικόνα 6:. Απεικόνιση του Fisher's Iris flower dataset

2.5.2 Pima Indians Diabetes

Το pima Indians diabetes dataset περιλαμβάνει χαρακτηριστικά για γυναίκες άνω των 21 οι οποίες κατάγονται από την φυλή pima και ο στόχος του είναι η πρόβλεψη για το εάν η ασθενής πάσχει από διαβήτη.

1	Number of times pregnant
2	Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3	Diastolic blood pressure (mm Hg)
4	Triceps skin fold thickness (mm)
5	2-Hour serum insulin (mu U/ml)
6	Body mass index (weight in kg/(height in m)^2)
7	Diabetes pedigree function
8	Age (years)
9	Class variable (0 or 1)

Εικόνα 7:. Χαρακτηριστικά του Pima Indians Diabetes Dataset

Εξέταση της απόδοσης ταξινομητών (kNN, Bayes, Naive Bayes) και χρήση δικτύων Bayes για συμπερασμό πιθανοτήτων

2.5.3 Census Income

Το census income dataset περιλαμβάνει δεδομένα απογραφής ενηλίκων του 1994 από το U.S. Census Bureau. Το σύνολο δεδομένων έχει ως στόχο την πρόβλεψη της τάξης εισοδήματος (<=50K ή >50K) με βάση αλλά χαρακτηριστικά των απογραφέντων. Το σύνολο περιλαμβάνει 48842 δείγματα.

1	workclass
2	education
3	maritalstatus
4	occupation
5	race
6	sex
7	nativecountry
8	class

Εικόνα 8:. Χαρακτηριστικά που χρησιμοποιούνται από το Census Income Dataset

3. ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

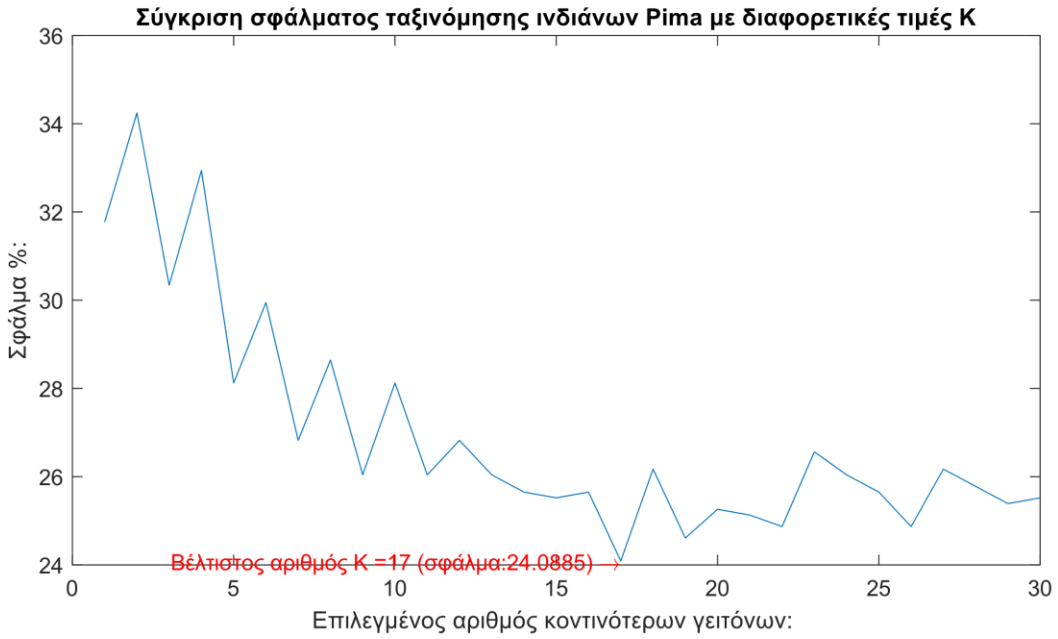
3.1 Ταξινομητής KNN

Στο θέμα 1 ερώτημα α υποερώτημα 1 ζητήθηκε η υλοποίηση ταξινομητή κοντινότερων γειτόνων για ταξινόμηση δειγμάτων του Fisher Iris data set σε 3 κλάσεις - είδη. Χρησιμοποιώντας τη μέθοδο 10 – fold cross validation παρατηρήθηκε πως η επιλογή του αριθμού των κοντινότερων γειτόνων επηρεάζει αρκετά το ποσοστό ορθής ταξινόμησης των δειγμάτων με βέλτιστη τιμή το K = 10 και ποσοστό σφάλματος 2,66% (ποσοστό επιτυχίας 97,33%). Οι κλάσεις είναι εύκολα διαχωρίσιμες.

***Εικόνα 9**: Σύγκριση σφάλματος ταξινόμησης KNN Iris για διάφορες τιμές K*

Να σημειωθεί πως η χρήση της μεθόδου 10-fold cross validation επηρεάζει τα αποτελέσματα ανάλογα με τη κατανομή των δειγμάτων εκπαίδευσης σε train (9/10) και validation (1/10) sets οπότε στον επισυναπτόμενο κώδικα Matlab πάντα αρχικοποιείται η γεννήτρια ψευδοτυχαίων αριθμών με τη μέθοδο «`rng(10);`*».*

Για υποερώτημα 2 του ίδιου ερωτήματος υλοποιήθηκε ταξινομητής κοντινότερων γειτόνων για τη ταξινόμηση δειγμάτων του Pima Indians data set με την ίδια μέθοδο όπως και παραπάνω. Το σφάλμα ταξινόμησης σε αυτή τη περίπτωση ήταν μεγαλύτερο με βέλτιστο αριθμό γειτόνων K = 17 και ελάχιστο σφάλμα 24,08% (ποσοστό επιτυχίας 75,92%). Τα χαρακτηριστικά του συγκεκριμένου data set επικαλύπτονται μερικώς μεταξύ δειγμάτων διαφορετικών κλάσεων με αποτέλεσμα η ταξινόμηση να δυσχεραίνει.



Εικόνα 10: Σύγκριση σφάλματος ταξινόμησης KNN Pima για διάφορες τιμές K

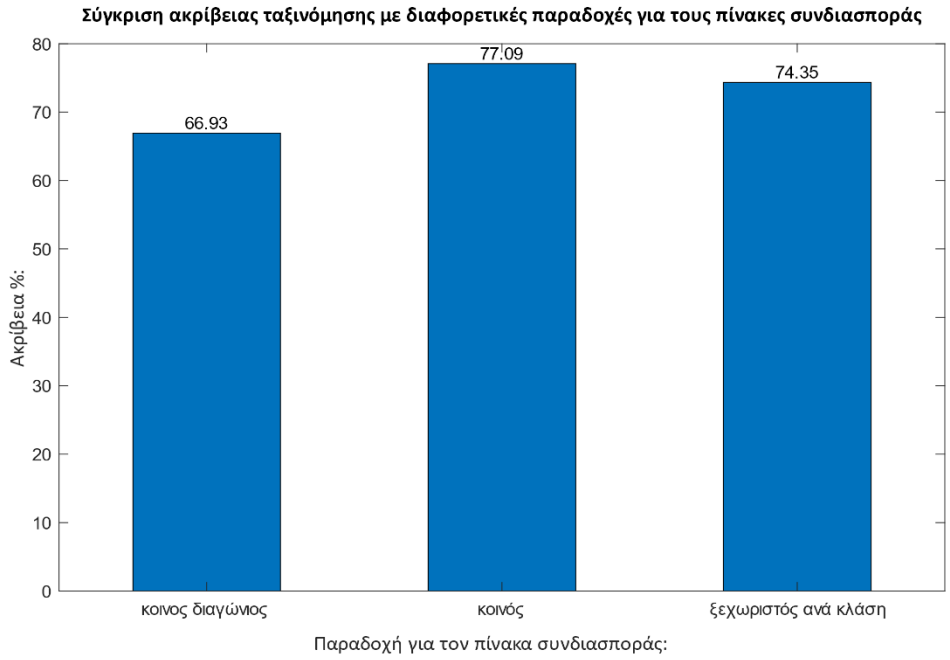
3.2 Ταξινομητές Bayes

Για το ερώτημα β του 1ου Θέματος υλοποιήθηκαν ταξινομητές Bayes με υποκείμενες γκαουσιανές κατανομές για τις πυκνότητες πιθανότητας των 2 κατηγοριών του Pima Indians data set (διαβητικός ή όχι), με 3 διαφορετικές παραδοχές:

1. Κοινός πίνακας συνδιασποράς για τις κατηγορίες, πολλαπλάσιος του μοναδιαίου πίνακα
2. Κοινός πίνακας συνδιασποράς για τις κατηγορίες ίσος με τον πίνακα συνδιασποράς του συνόλου των δεδομένων εκπαίδευσης
3. Διαφορετικοί πίνακες συνδιασποράς για τις κατηγορίες.

Εξέταση της απόδοσης ταξινομητών (kNN, Bayes, Naive Bayes) και χρήση δικτύων Bayes για συμπερασμό πιθανοτήτων

Παρατηρήθηκε πως μεταξύ των τριών παραδοχών, η χρήση κοινού πίνακα συνδιασποράς ίσου με τον πίνακα συνδιασποράς του συνόλου των δειγμάτων εκπαίδευσης ήταν η βέλτιστη επιλογή, με ποσοστό επιτυχούς ταξινόμησης 77,09%.



Εικόνα 11: *Σύγκριση ποσοστού επιτυχούς ταξινόμησης Bayes Pima με διάφορες παραδοχές*

Σε σύγκριση με τον ταξινομητή KNN ο βέλτιστος Bayes (εκ των παραπάνω παραδοχών) απέδωσε καλύτερα κατά σχεδόν 2 μονάδες %.

3.3 Ταξινομητής Naive Bayes

Για το γ ερώτημα του 1ου θέματος υλοποιήθηκε απλοϊκός ταξινομητής Bayes και δοκιμάστηκε έναντι του ίδιου data set (Pima Indians). Το αποτέλεσμα ήταν ποσοστό επιτυχούς ταξινόμησης 75.13%, πολύ κοντά με στις επιδόσεις του γκαουσιανού Bayes με την παραδοχή των διαφορετικών πινάκων συνδιασποράς ανά κλάση (74,35%), και σχεδόν ίδια επίδοση με τον KNN ταξινομητή (75,92%).

Επιβεβαιώνεται , σε σύγκριση με τους προαναφερθέντες ταξινομητές, πως ο απλοϊκός Bayes καταφέρνει καλά ποσοστά με ένα πολύ απλουστευμένο μοντέλο πιθανοτήτων λιγότερης πολυπλοκότητας και απαιτήσεων σε πόρους.

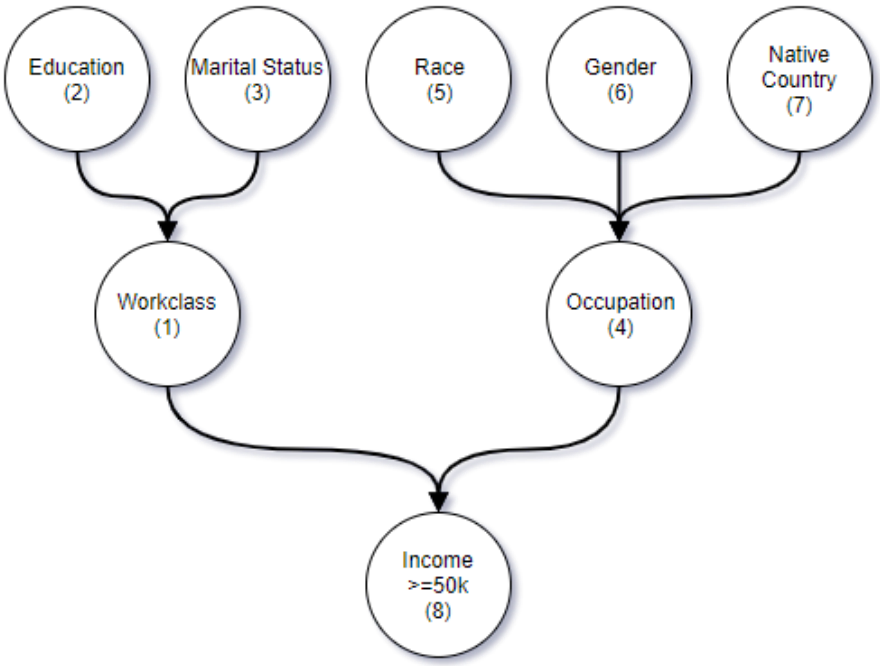
3.4 Δίκτυο Bayes

Με την χρήση του Bayes Net Toolbox for Matlab δημιουργείται το δίκτυο της εικόνας 12 . Στην συνέχεια το δίκτυο εκπαιδεύεται με την χρήση των δεδομένων του Census Income dataset (Παράρτημα 1.4 – bayesNetIncome.m). Το δίκτυο χρησιμοποιείται για μία περίπτωση συμπερασμού πιθανοτήτων με ευθύγραμμη διάδοση:

p(income='>50k'| education='Bachelors', maritalstatus='Never-married', race='White', gender='Male', nativecountry='Greece') = 0.3214

Επίσης χρησιμοποιείται για μια περιπτωση συμπερασμού πιθανοτήτων με ανάδρομη διάδοση:

p(education='Highschool grad' | workclass='never-worked') = 0.1611



Εικόνα 12: Γράφος Δικτύου Bayes

4. ΣΥΜΠΕΡΑΣΜΑΤΑ

Από την πειραματική διαδικασία φαίνεται ότι οι ταξινομητές KNN επιτυγχάνουν τα μέγιστα ποσοστά ακρίβειας αν επιλεγεί ο κατάλληλος αριθμός γειτόνων, καθώς αυτός καθορίζει την πολυπλοκότητα του μοντέλου. Επίσης παρατηρείται ότι η χρήση ταξινομητών Bayes με κοινό μητρώο συνδιασποράς για όλες τις κλάσεις εμφανίζει την καλύτερη ακρίβεια με αμέσως επόμενη αυτή του απλοϊκού ταξινομητή Bayes . Τέλος από την χρήση των δικτύων Bayes φαίνεται ότι αυτά μπορούν να πετύχουν καλές εκτιμήσεις των αγνώστων χαρακτηριστικών εφόσον το μέγεθος του συνόλου επαρκεί και περιέχει αντιπροσωπευτικό δείγμα.

Παράρτημα Α.Κώδικας Matlab

nfoldDataset.m

```
function [ dataset ] = nFoldDataset( database, folds )
%NFOLDDATASET Summary of this function goes here
%   Detailed explanation goes here
    [m,n] = size(database);
    database(randperm(m),:)=database;
    seg=uint32(round(m/folds));
    for i=1:folds
        validx=((folds-i)*seg)+1:(folds-i+1)*seg;
        dataset(i).train=database;
        dataset(i).train(validx,:)=[];
        dataset(i).val=database(validx,:);
    end
end
```

1. KNN Classifier

knnclassifier.m

```
%KNN Classifier function

function accur = knnclassifier(traindata, testdata, K)

%Find distance with all training datapoints, sort and poll
for i = 1 : size(testdata)
x = testdata(i,:);
dist = sqrt((traindata(:, 1) - x(1)) .^ 2 + (traindata(:, 2) - x(2)) .^ 2 +
(traindata(:, 3) - x(3)) .^ 2 + (traindata(:, 4) - x(4)) .^ 2);
classes = traindata(:, 5);
dist(:, 2) = classes;
poll = sortrows(dist, 1);

%For tiebreak in case of even K
if (mod(K, 2) == 1)
expclass(i) = mode(poll(1 : K, 2));
else
    temp = poll(1 : K, 2);
    uniq = unique(temp);
    p = size(uniq);
    bincounts = histc(temp, uniq);
    q = max(bincounts);
    %if number of unique elements = 2 && highest frequency is K/2, then there
is tie
    M = (p == 2) & (q == K/2);
    %Alloted the class which is at closest distance
    expclass(i) = mode(poll(1 : K - M, 2));
end
```

```
end

%Error percentage calculation
error = transpose(expclass) - testdata(:,5);
accur = ((size(error, 1) - nnz(error))/size(error, 1));

end
```

nfold_iris.m

```
clear;
iris = datatable2mat(dataset2table(dataset('File', 'iris.data',...
    'Delimiter', ',', 'ReadVarNames', false)));
folds = 10;
iris = nFoldDataset(iris,folds);
for K=1:9
    for f=1:folds
         err(f)=knnclassifier(iris(f).train,iris(f).val, K);
    end
    sub = subplot(3,3, K);
    plot(err);
    title(['Plot for K = ', num2str(K)])
end

function dat = datatable2mat(datatable)
    %Converts table with numerical and categorical data to matrix
    c= grp2idx(table2array(datatable(:,end)));
    dat = table2array(datatable(:,1:end-1));
    dat(:,end+1)=c;
end
```

2. Bayes Classifier

bayes_classifier.m

```
% Bayes Classifier
% Assumes last column of input TRAIN and TEST matrices is the class
%
% set covarianceMode to 1 for common diagonal covariance matrix
% set covarianceMode to 2 for common covariance matrix
% set covarianceMode to 3 for a seperate covariance matrix for each class

function testval = bayes_classifier(XTRAIN, XTEST, covarianceMode)
if nargin < 3
    fprintf('No covarianceMode specified : ');
    covarianceMode = 3;
end
fprintf('Fitting Bayes Classifier :\n');
```

```
trainDataSize  = size(XTRAIN,1);
testDataSize  = size(XTEST,1);

% Separate all the training samples into separate classes
[C,~,idx] = unique(XTRAIN(:,end));
classes = accumarray(idx,1:trainDataSize,[],@(r){XTRAIN(r,1:end-1)});

% Generate the summary map from all the classes
classMap = containers.Map(C, classes);

covMap = containers.Map('KeyType','int32','ValueType','any');
meanMap = containers.Map('KeyType','int32','ValueType','any');

% -------------------Calculate covariance and mean-------------------------
switch covarianceMode
    case 1 % Use a common DIAGONAL covariance regardless of class
        fprintf('using COMMON DIAGONAL covariance matrix for all classes
:\n');

        % First gather all features
        allClassesFeatures = [];
        for i = 1:size(C,1)
            c = C(i);
            allClassesFeatures = [allClassesFeatures; classMap(c)];
        end

        % Calculate one covariance matrix regardless of class and multiply
        % it by the diagonal matrix
        covariance = var(allClassesFeatures(:)) * eye
(size(allClassesFeatures,2));

        for i = 1:size(C,1)
            c = C(i);
            covMap(c) = covariance; % Use the same covariance matrix for all
classes
            meanMap(c) = mean(classMap(c),1);
        end
    case 2 % Use a common covariance matrix regardless of class
        fprintf('using COMMON covariance matrix for all classes :\n');

        allClassesFeatures = [];
        for i = 1:size(C,1)
            c = C(i);
            allClassesFeatures = [allClassesFeatures; classMap(c)];
        end

        % Calculate one covariance matrix regardless of class
        covariance = cov(allClassesFeatures);

        for i = 1:size(C,1)
            c = C(i);
            covMap(c) = covariance;
            meanMap(c) = mean(classMap(c),1);
        end
    case 3
        % Use seperate covariance matrices for each class
```

```
        fprintf('using SEPERATE covariance matrix for each class :\n');

        for i = 1:size(C,1)
            c = C(i);
            if size(classMap(c),1) ~= 0
                covMap(c) = cov(classMap(c)); % Calculate the covariance
matrix of THIS class
                meanMap(c) = mean(classMap(c),1);
            end
        end
end

% -------------Find the accuracy of training and testing data--------------

% First get the a priori probabilities for each class
priors = []; % Initialise some space to store the priors
 for i = 1:size(C,1) % Loop  over all classes

        c = C(i); % Get the current class

        % Get the a priori probability of this class
        priorProb = size(classMap(c),1)/trainDataSize;

        % Store it for later use
        priors = cat(2, priors, priorProb);
 end

% Training ----------------------------------------------------------------

% Initialise some space to store the estimated class for each x
calculatedY = [];

% Do the estimate!
for trainIndex=1:trainDataSize % Loop  over all x in the train set

    prob = []; % Initialise some space to store the resulting probabilities

    % Calculate the a posteriori probabilities for all classes
    % for current x
    for i = 1:size(C,1) % Loop  over all classes

        c = C(i); % Get the current class

        % Get the current class' a posteriori probability given current x
        % using mean and covariance for this class.
        % (Note: depending on the covarianceMode chosen, common and
diagonal,
        % common or seperate covariance matrices might be used. See above.)
        % (Note: no need to normalize by p(x), as its the same everywhere)
        condProb = mvnpdf(XTRAIN(trainIndex,1:end-1), meanMap(c),
covMap(c))*priors(i);

        % Put the a posteriori and a priori probabilities of current class
        % for current x side by side for
        % later comparison
        prob = cat(2, prob, condProb);
```

```
    end

    % Get the greatest a posteriori probability (estimate of which class the
    % x falls under)
    [maxval,argmax] = max(prob);

    % Store the estimated class for later use
    calculatedY = [calculatedY;argmax];
end

% Now count how many estimates were right by comparing them with the labels
accuratePredictions = (calculatedY == XTRAIN(:,end));
% Get the accuracy percentage
trainingAccuracy = sum(accuratePredictions)*100/size(accuratePredictions,1);
fprintf('Training Accuracy = %4.2f\n',trainingAccuracy);


%Testing ----------------------------------------------------------------

% Initialise some space to store the estimated class for each x
calculatedY = [];

% Do the estimate!
for testIndex=1:testDataSize % Loop  over all x in the train set

    prob = []; % Initialise some space to store the resulting probabilities

    % Calculate the a posteriori probabilities for all classes
    % for current x
    for i = 1:size(C,1) % Loop  over all classes

        c = C(i); % Get the current class

        % Get the current class' a posteriori probability given current x
        % using mean and covariance for this class.
        % (Note: depending on the covarianceMode chosen, common and
diagonal,
        % common or seperate covariance matrices might be used. See above.)
        % (Note: no need to normalize by p(x), as its the same everywhere)
        condProb = mvnpdf(XTEST(testIndex,1:end-1), meanMap(c),
covMap(c))*priors(i);

        % Put the a posteriori and a priori probabilities of current class
        % for current x side by side for
        % later comparison
        prob = cat(2, prob, condProb);
    end

    % Get the greatest a posteriori probability (estimate of which class the
    % x falls under)
    [maxval,argmax] = max(prob);

    % Store the estimated class for later use
    calculatedY = [calculatedY;argmax];
end
```

```
% Now count how many estimates were right by comparing them with the labels
accuratePredictions = (calculatedY == XTEST(:,end));
% Get the accuracy percentage
testAccuracy = sum(accuratePredictions)*100/size(accuratePredictions,1);
fprintf('Test Accuracy = %4.2f\n\n',testAccuracy);


% return the accuracy percentage to the caller
testval = testAccuracy;
```

gaussian_bayes.m

```
clear;
close all;

tic

rng(8); % For reproducibility
load('../datasets.mat');

% Randomize dataset
dataSetSize = size(pimaindiansdiabetes,1);
data_rand = table2array(pimaindiansdiabetes(randperm(dataSetSize),:));
data_rand(:,end) = data_rand(:,end) + 1; %Shift class one value up to help
with later calculations

accuracies = [];

% Calculate accuracy rates for all different covariance modes using
% 10fold cross-validation method
for i = 1:3 % Loop over all covariance modes

    % mode 1 for common diagonal covariance matrix
    % mode 2 for common covariance matrix
    % mode 3 for a seperate covariance matrix for each class
    covarianceMode = i;

    % Wrap the classifier function inside another so we can pass the mode as
a
    % parameter and keep matlab happy.
    % That is because matlab's crossval function takes as first
    % parameter a function who's signature is as follows:
    % @(XTRAIN, XTEST)(success_rate)
    % The first being the train data set and the seconf the test data set.
    % Our bayes_classifier accepts a third parameter as well.
    classifier = @(XTRAIN, XTEST)(bayes_classifier(XTRAIN, XTEST,
covarianceMode));

    % 10fold cross-validate
    overallAccuracy = mean(crossval(classifier, data_rand));
    fprintf('Bayes Overall Accuracy = %4.2f\n',overallAccuracy);
    accuracies = [accuracies; overallAccuracy];
end
figure;
```

```
x = categorical({'common diagonal covariance matrix','common covariance
matrix','seperate covariance matrix for each class'});
bar(x, accuracies);
title('Comparison of classifier accuracy with different covariance modes');
xlabel('Covariance Mode:'); % x-axis label
ylabel('Accuracy %:'); % y-axis label
for i=1:3
    text(x(i),accuracies(i),num2str(accuracies(i),'%0.2f'),...
                'HorizontalAlignment','center',...
                'VerticalAlignment','bottom')
end
toc
```

3. Naïve Bayes Classifier

naivebayesindians.m

```
clear;
load('../datasets.mat');
dat = datatable2mat(pimaindiansdiabetes);
folds = nFoldDataset(array2table(dat),10);

for i=1:10
    dat = table2array(folds(i).train);
    class1 = dat(dat(:,9) == 1, 1:8);
    mu1 = mean(class1);
    sig1 = var(class1).*eye(8);
    class2 = dat(dat(:,9) == 2, 1:8);
    mu2 = mean(class2);
    sig2 = var(class2).*eye(8);

    dat = table2array(folds(i).val);
    p=[mvnpdf(dat(:,1:8),mu1,sig1) mvnpdf(dat(:,1:8),mu2,sig2)];
    [x,pred] = max(p,[],2);
    acc(i) =1-(sum(abs(pred-dat(:,9)))/length(pred));
end
disp(['Accuracy: ' num2str(mean(acc))]);
function dat = datatable2mat(datatable)
    %Converts table with numerical and categorical data to matrix
    c= grp2idx(table2array(datatable(:,end)));
    dat = table2array(datatable(:,1:end-1));
    dat(:,end+1)=c;
end
```

4. Bayesian Net

bayesNetIncome.m

```
clear;
load('income-usable.mat');
N=8;
for i=1:N
    classes{i}=unique(income(:,i));
    classnum(i)=height(classes{i});
    data(:,i)=grp2idx(table2array(income(:,i)));
end
% data = data(1:300,:);

%Create network
% 2 3  5 6 7
% \ /  \ | /
%  1     4
%   \   /
%     8

dag = zeros(N,N);
dag(2,1)=1; dag(3,1)=1;
dag(5,4)=1; dag(6,4)=1; dag(7,4)=1;
dag(1,8)=1; dag(4,8)=1;

%Create network
onodes = [2,3,5,6,7];
node_sizes = classnum;
bnet = mk_bnet(dag, node_sizes, 'observed', onodes);
% use random params
for i=1:N
    bnet.CPD{i} = tabular_CPD(bnet, i);
end

%train network
ncases = size(data, 1);
cases = num2cell(data');
engine = jtree_inf_engine(bnet);
bnet = learn_params_em(engine, cases);

%inference forward
engine = jtree_inf_engine(bnet);
evidence = cell(1,N);
evidence(onodes) = num2cell([10 5 5 2 12]);
[engine, ll] = enter_evidence(engine, evidence);
p8 = marginal_nodes(engine, 8);

%inference inverse
nodes = [1];
engine = jtree_inf_engine(bnet);
evidence = cell(1,N);
evidence(nodes) = num2cell([3]);
[engine, ll] = enter_evidence(engine, evidence);
p2 = marginal_nodes(engine, 2);
```