Analyse comparative des modèles Naive Bayes et Tree Augmented Naive Bayes(TAN)

Puwei Liao Shurong Zhang

Avril 2022

Contexte

Datasets :

- Iris : 150 individus qui contient 4 variables continues (Sepal.Length, Sepal.Width, Petal.Length et Petal.Width) pour prédire la variable catégorielle Species (setosa, versicolor et virginica).
- Pima : 768 individus qui contient 8 variables continues (pregnant, glucose, pressure, triceps, insulin, mass, pedigree et age) pour prédire la variables catégorielle diabetes (pos et neg)

Le but de ce projet est de programmer deux classifieurs Naive Bayes et TAN pour faire la prédiction et puis comparer leur performance avec la méthode de validation-croisée.

Naive Bayes modèle

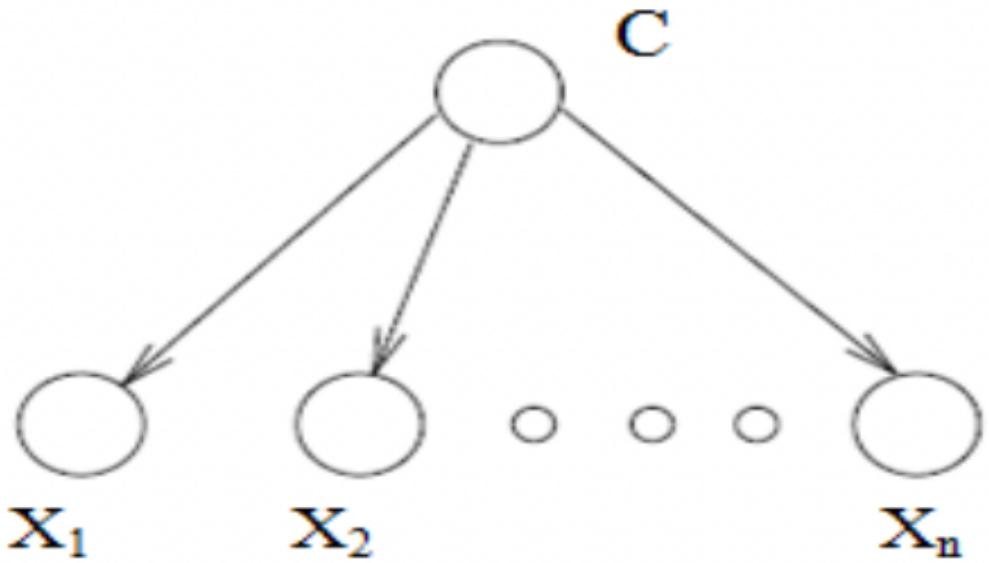


Figure – Graphe de Naive Bayes modèle

C : Classe Label

$X_1 ... X_n$: Variables Aléatoires

On remarque que tous les variables aléatoires sont indépendantes entre eux.

Naive Bayes modèle

La loi jointe de ces variables aléatoires sont réduits à
$P(X_1, \ldots, X_n) = P(C) \prod_i P(X_i \mid C)$

La formule bayesian nous donne :
$P(C \mid X_1, \ldots, X_n) = P(X_1, \ldots, X_n \mid C) \cdot P(C) / P(X_1, \ldots, X_n)$

Comme le dénominateur $P(X_1, \ldots, X_n)$ ne joue pas de rôle dans la maximisation de $P(C \mid X_1, \ldots, X_n)$ sur C, il peut être ignoré et seul le numérateur peut être considéré pour classer les instances.

Implémentation de Naive Bayes modèle

Initialisation :

- μ_{setosa}, μ_{ver} et μ_{vir} trois listes, chaque élément de liste correspond à la moyenne pour chaque variable.
- σ_{setosa}, σ_{ver} et σ_{vir} trois listes, chaque élément de liste correspond à la variance pour chaque variable.

Entraînement : Utiliser nos données d'apprentissage (training set),

- pour les valeurs de chaque élément des vecteurs μ, calculer sa moyenne des chaque variables pour chaque classe.
- Pour chaque élément des vecteurs σ, calculer sa variance des chaque variables pour chaque classe.

Implémentation de Naive Bayes modèle

```
for (i in 1:nb_var) {
  mu_setosa[i]=mean(train[which(train$Species=="setosa"),i])
  mu_ver[i]=mean(train[which(train$Species=="versicolor"),i])
  mu_vir[i]=mean(train[which(train$Species=="virginica"),i])

  sigma_setosa[i]=var(train[which(train$Species=="setosa"),i])
  sigma_ver[i]=var(train[which(train$Species=="versicolor"),i])
  sigma_vir[i]=var(train[which(train$Species=="virginica"),i])
}
p1=1
p2=1
p3=1

for (i in 1:nb_var) {
  p1=p1*dnorm(as.numeric(test[i]),mean=mu_setosa[i],sd=sqrt(sigma_setosa[i]))
  p2=p2*dnorm(as.numeric(test[i]),mean=mu_ver[i],sd=sqrt(sigma_ver[i]))
  p3=p3*dnorm(as.numeric(test[i]),mean=mu_vir[i],sd=sqrt(sigma_vir[i]))
}


p1=p1*(length(which(train$Species=="setosa"))/dim(train)[1])
p2=p2*(length(which(train$Species=="versicolor"))/dim(train)[1])
p3=p3*(length(which(train$Species=="virginica"))/dim(train)[1])
```

Tree augmented Naive Bayes

But : Prendre le compte les corrélations entre les variables pour s'améliorer la performance de classification.

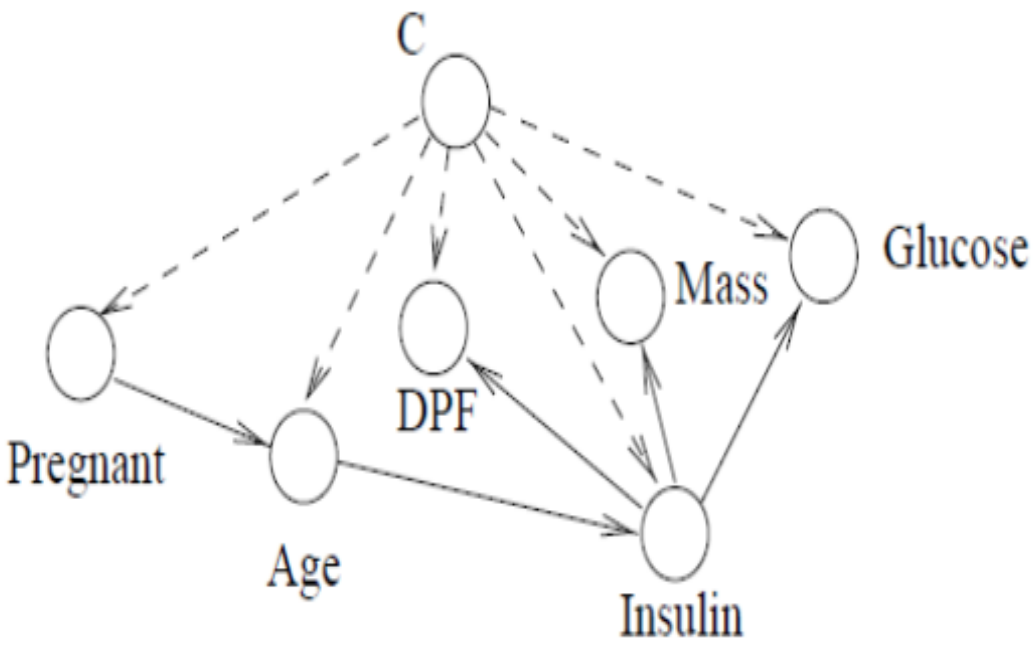
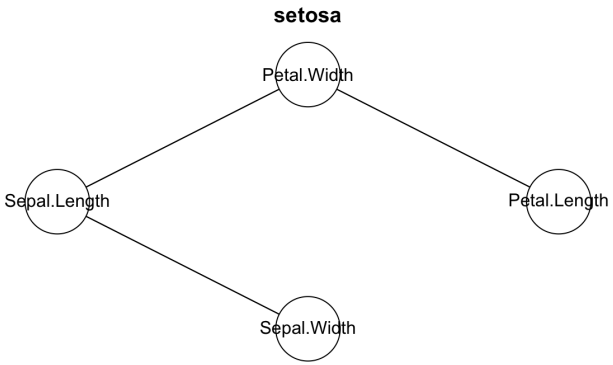


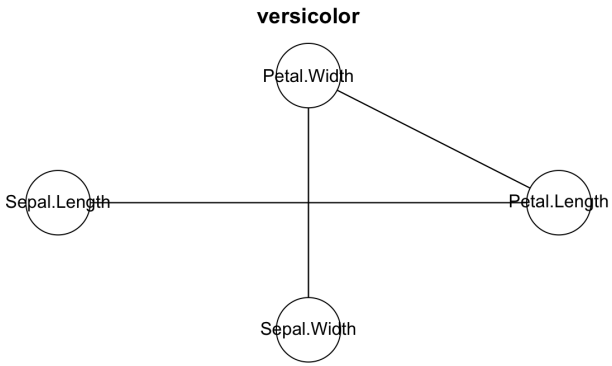
Figure – Exemple de graphe d'un modèle TAN

- Arbre non dirigé pour chaque classification
- Choisissez la racine et définissons les orientations
- Matrice adjacente
- Calcul de probabilité conditionnel

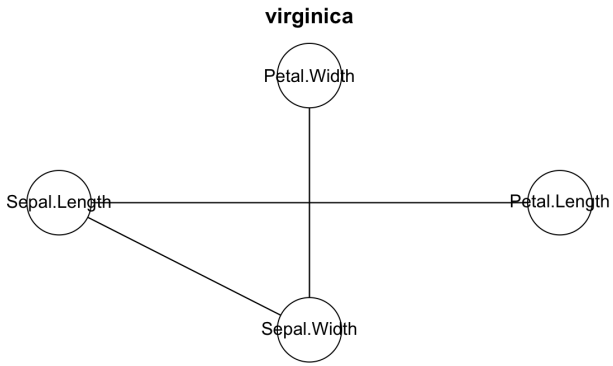
Arbre des variables de Iris



setosa

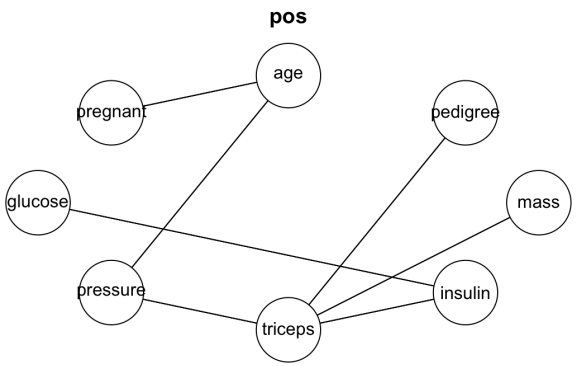


versicolor

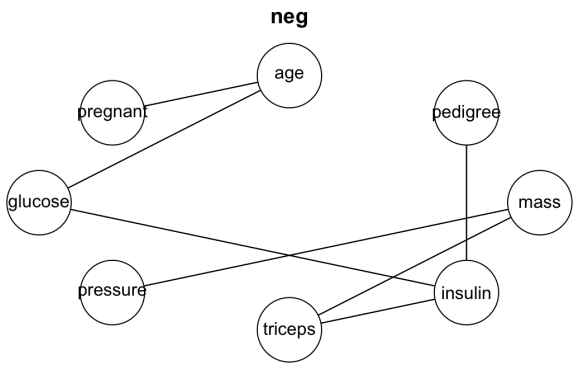


virginica

Arbre des variables de Pima



pos



neg

Averaged One-Dependent Estimator(AODE)

$$\mathrm{P}(C \mid X_1, \ldots, X_n) = P(C) \cdot P\left(X_{\text{root}} \mid C\right) \prod_i P\left(X_i \mid C, X_{\text{parent}}\right)$$

$$\mathrm{P}(\mathrm{c} \mid \mathbf{x}) \propto \sum_{\substack{i=1 \\ |Dx_i| \geqslant m'}}^{d} P\left(c, X_{\text{root}}\right) \prod_{j=1}^{d} P\left(x_j \mid c, X_{\text{parent}}\right)$$

$$P\left(\hat{c}, x_i\right) = \frac{\left|D_{c,x_i}\right|+1}{|D|+N_i} \quad P\left(\hat{x_j} \mid c, x_i\right) = \frac{\left|D_{c,x_i,x_j}\right|+1}{\left|D_{c,x_i}\right|+N_j}$$

Tree augmented Naive Bayes

```
  for (i in 1:nb_var) {
    p1_1=1
    for (j in (which(Mat_adj_setosa[i,]==1))) {
      p1_1=p1_1*((length(which(train_1[,i]==test[,i]&&train_1[,j]==test[,j]))+1)/(
length(which(train_1[,i]==test[,i]))+length(unique(train_1[,j]))))

    }
    p1=p1+p1_1*(length(which(train_1[,i]==test[,i]))+1)/(length(train_1[,1])+lengt
h(unique(train_1[,i])))

  }
```

Algorithme de validation croisée

On va utiliser la méthode K-Fold pour $K = 10$.

On coupe les données en 10 morceaux puis s'entraîner notre modèle avec les 9 morceaux puis faire le test avec l'un reste.

Compare avec les vraies valeurs, puis on constate qu'il y a combien bonnes prédictions.

Validation croisée avec deux méthodes

Dataset iris :

Nombre de classification correct			
K	Naive Bayes	TAN	Total
1	14	13	15
2	14	12	15
3	14	11	15
4	14	13	15
5	15	12	15
6	14	13	15
7	13	11	15
8	15	12	15
9	15	14	15
10	15	14	15

Validation croisée avec deux méthodes

Dataset Pima :

Nombre de classification correct			
K	Naive Bayes	TAN	Total
1	61	46	76
2	56	45	77
3	62	44	77
4	54	45	77
5	61	50	77
6	57	48	77
7	55	45	77
8	62	46	76
9	49	45	77
10	57	42	77