SELECTION D'ATTRIBUTS

Machine Learning

Travail fait par :

- BOUADI Mohamed
- BOUZENIA Islem

Groupe: SIQ 4

Enseignante du module :

- Mme. BENATCHBA

JUNE 27, 2019

ESI

OUED SEMAR

CONTENTS

INTRODUCTION

En tant que technique couramment utilisée dans le prétraitement des données pour l'apprentissage automatique, la sélection de caractéristiques identifie les caractéristiques importantes et supprime les caractéristiques non pertinentes, redondantes ou sonores afin de réduire la dimensionnalité de l'espace des fonctions. Il améliore l'efficacité, la précision et la compréhensibilité des modèles construits par l'apprentissage automatique. Les techniques de sélection des caractéristiques ont été largement utilisées dans une variété d'applications, telles que l'analyse génomique, la récupération d'informations et la catégorisation de texte.

Les chercheurs ont introduit de nombreux algorithmes de sélection de caractéristiques avec différents critères de sélections. Cependant, il a été découvert qu'aucun critère unique ne convient le mieux pour toutes les applications.

Dans ce rapport nous allons présenter les approches de sélection des données ainsi que les algorithmes les plus connus de chaque approche. Enfin, nous terminerons par une application comparative sur un exemple de classification binaire. Il est à noter que ce document va concentrer plus sur le principe des méthodes ainsi que les avantages et les inconvénients de chaque méthode et cela va comprendre aussi les cas d’utilisation de chaque algorithme d’une approche. Par contre les détails d’implémentation et le code entier vous le trouverez dans le fichier attaché à ce papier (On se limite ici à présenter des parties du code).

SELECTION D'ATTRIBUTS

La sélection des attributs (également appelée sélection de sous-ensembles ou sélection de variables) est un processus couramment utilisé dans l'apprentissage automatique pour résoudre le problème de haute dimensionnalité. Il sélectionne un sous-ensemble de fonctionnalités importantes et supprime les fonctionnalités non pertinentes, redondantes et bruyantes pour des représentations des données plus concises. Les avantages de la sélection des attributs sont multiples. Tout d'abord, La sélection permet d'économiser beaucoup de temps dans le processus d'apprentissage en supprimant les attributs redondants. Deuxièmement, sans l'interférence de caractéristiques non pertinentes, redondantes et bruyantes, Les algorithmes d'apprentissage peuvent se concentrer sur les aspects les plus importants des données et créer des processus plus simples mais plus efficaces, modèles de données précises, et donc, les performances de classification sont améliorées. Troisièmement, La sélection peut nous aider à construire un modèle plus simple et plus général et à mieux comprendre le concept sous-jacent de la tâche.

OBJECTIFS DE LA SELECTION

Différents algorithmes de sélection d'attributs peuvent avoir différents objectifs à atteindre. Le suivant est une liste d'objectifs communs entre eux :

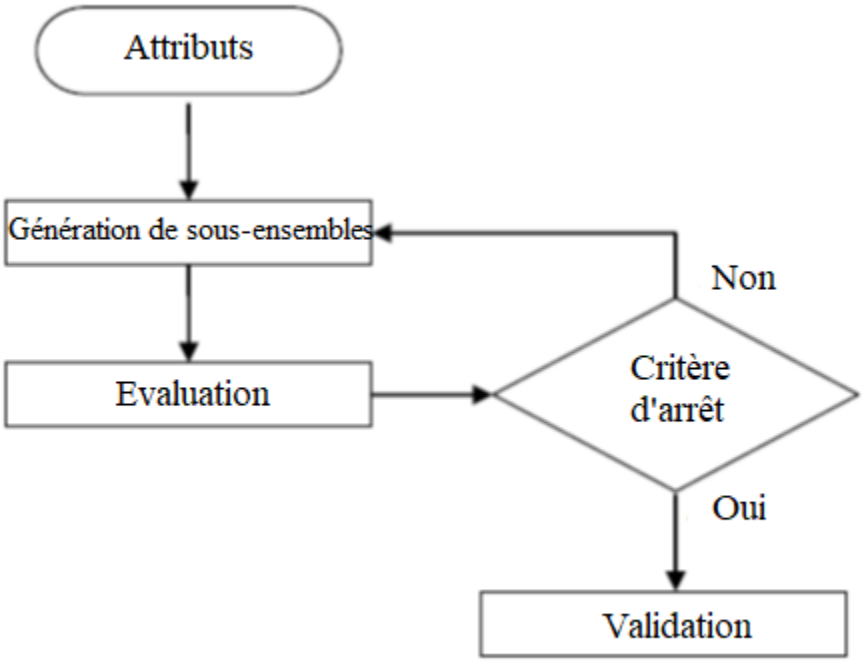
1. Trouver le sous-ensemble de caractéristiques de taille minimale nécessaire et suffisant au concept cible.
2. Sélectionner un sous-ensemble de N entités parmi un ensemble de M entités, N < M, telles que la valeur de la fonction objective est optimisée pour tous les sous-ensembles de taille N.
3. Choisir un sous-ensemble de fonctionnalités pour améliorer la précision des prévisions ou réduire la taille de la structure sans diminuer de manière significative la précision de prédiction du classificateur construit en utilisant seulement les caractéristiques sélectionnées.
4. Sélectionner un sous-ensemble tel que la distribution résultante, à partir des valeurs des caractéristiques sélectionnées, est aussi proche que possible de la distribution d'origine.

PROCEDURE DE SELECTION

Une procédure typique de sélection des caractéristiques comprend quatre étapes de base :

1. Génération d’un sous-ensemble
2. Evaluation du sous-ensemble
3. Critère d'arrêt
4. Validation des résultats

Le processus commence par la génération d’un sous-ensemble qui utilise une certaine stratégie de recherche pour produire des candidats (sous-ensemble d’attributs). Ensuite, chaque sous-groupe candidat est évalué et comparé avec le meilleur précédent. Le processus de génération et d'évaluation de sous-ensemble est répété jusqu'à ce qu'un critère d'arrêt donné soit satisfait. Enfin, le meilleur sous-ensemble d’attributs sélectionné est validé par des connaissances préalables ou par des tests.



APPROCHES DE SELECTION

Typiquement, ils existent trois approches pour la sélection des attributs, chacune utilisant différents principes et implémentant une variété d'algorithmes. La première approche est celle du filtrage qui est purement statistique, facile à l'implémentation et rapide pendant l'exécution mais qui ne donne pas de bons résultats en comparant avec les deux autres approches : Wrapper et Embedded.

APPROCHE PAR FILTRAGE

Comme son nom l'indique cette méthode permet de filtrer les attributs et de prendre qu'un sous-ensemble des fonctionnalités les plus pertinentes. C'est une méthode statistique qui analyse les données et selon une propriété statistique (la moyenne, la variance, la corrélation, la contribution …) affecte un poids à un attribut. C'est une méthode simple et très rapide en exécution, cependant elle ne garantit pas la meilleure exactitude en termes d'apprentissage, mais elle permet quand même de dégager dans un premier temps l'attribut le plus important et aussi l'attribut le moins important.

Nous allons utiliser le Data Set « The built of Boston » qui contient les résultats du vote pour des maison, la figure suivante représente une partie de ce Data Set :

	Class Name	handicapped-infants	water-project-cost-sharing	adoption-of-the-budget-resolution	physician-fee-freeze	el-salvador-aid	religious-groups-in-schools
0	republican	n	y	n	y	y	y
1	republican	n	y	n	y	y	y
2	democrat	?	y	y	?	y	y
3	democrat	n	y	y	n	?	y
4	democrat	y	y	y	n	y	y
5	democrat	n	y	y	n	y	y
6	democrat	n	y	n	y	y	y
7	republican	n	y	n	y	y	y
8	republican	n	y	n	y	y	y
9	democrat	y	y	y	n	n	n
10	republican	n	y	n	y	y	n
11	republican	n	y	n	y	y	y
12	democrat	n	y	y	n	n	n
13	democrat	y	y	y	n	n	y
14	republican	n	y	n	y	y	y
15	republican	n	y	n	y	y	y
16	democrat	y	n	y	n	n	y
17	democrat	y	?	y	n	n	n

Nous commençons d'abord par importer et nettoyer le Data Set, Pour cela nous allons éliminer toutes les lignes qui ne contiennent pas une information qui est dans notre cas 'y' ou 'n'.

Le Snippet suivant montre une manière de le faire en Python :

```
path = 'C:\\Users\\yato\\PycharmProjects\\Selection\\house-votes-84.data'
names = ['Class Name',
         'handicapped-infants',
         'water-project-cost-sharing',
         'adoption-of-the-budget-resolution',
         'physician-fee-freeze',
         'el-salvador-aid',
         'religious-groups-in-schools',
         'anti-satellite-test-ban',
         'aid-to-nicaraguan-contras',
         'mx-missile',
         'immigration',
         'synfuels-corporation-cutback',
         'education-spending',
         'superfund-right-to-sue',
         'crime',
         'duty-free-exports',
         'export-administration-act-south-africa']

x = pd.read_csv(path, names=names)

header = [""]
X = x.drop(columns='Class Name')
drops = []
for i in range(len(X)):
    for value in X.values[i, :]:
        if value == '?':
            drops.append(i)
data = x.drop(index=drops)
X = data.drop(columns='Class Name')
```

Ensuite nous allons remplacer les données par des valeurs numériques : 1 pour 'y' et 0 pour 'n' et récupérer les valeurs des classes : 1 pour 'democrat' et 0 pour 'public'.

```
y = data.values[:, 0]
for i in range(len(y)):
    if y[i] == 'democrat':
        y[i] = 1
    else:
```

```
        y[i] = 0

for i in range(len(X.values)):
    for j in range(len(X.values[i, :])):
        if X.values[i, j] == 'y':
            X.values[i, j] = 1
        else:
            X.values[i, j] = 0
```

L'exécution du code précédent va donner le résultat suivant :

	handicapped-infants	water-project-cost-sharing	adoption-of-the-budget-resolution	physician-fee-freeze	el-salvador-aid	religious-groups-in-schools	anti-satellite-test-ban
5	0	1	1	0	1	1	0
8	0	1	0	1	1	1	0
19	1	1	1	0	0	0	1
23	1	1	1	0	0	0	1
25	1	0	1	0	0	0	1
26	1	0	1	0	0	0	1
27	1	1	1	0	0	0	1
28	1	0	0	1	1	0	1
29	1	1	1	0	0	0	1
30	0	1	0	1	1	1	0
32	1	1	1	0	0	0	1
33	0	1	0	1	1	1	0
34	1	1	1	0	0	0	1
35	0	1	0	1	1	1	0
37	1	1	0	1	1	1	0
38	0	1	0	1	1	1	0
39	1	0	1	0	0	0	1
42	1	0	1	0	0	0	1
43	1	0	1	0	0	0	1
46	1	1	1	0	0	0	1
48	1	1	1	0	0	0	1

METHODE1 : CORRELATION DE PEARSON

Une fois le Data Set est nettoyé et formaté nous allons essayer d'extraire les attributs les plus importants, le filtrage ici est effectué en utilisant une matrice de corrélation plus exactement une corrélation de Pearson.

Nous calculons d'abord de corrélation de Pearson entre chaque attribut et la variable de sortie. Nous ne sélectionnerons les entités qui sont fortement corrélées (en valeur absolue) avec la variable de sortie.

Le code suivant montre la fonction qui calcule la corrélation entre les variables indépendantes avec la sortie et affiche celles qui sont fortement corrélées :

```python
def cor_selector(X, y):
    cor_list = []
    # On calcule la correlation entre chaque x et y
    for i in X.columns.tolist():
        cor = np.corrcoef(list(X[i].values), list(y))
        cor_list.append(cor[0,1])

     # feature name
    cor_feature = X.iloc[:,np.argsort(np.abs(cor_list))[-3:]].columns.tolist()

    # feature selection? 0 for not select, 1 for select
    cor_support = [True if i in cor_feature else False for i in names]

    return cor_support, cor_feature
```

L'exécution de cette fonction donne le résultat suivant :

```python
cor_support, cor_feature = cor_selector(X, y)
print(str(len(cor_feature)), 'selected features')
print(cor_feature)
```

```
3 selected features
['education-spending', 'el-salvador-aid', 'physician-fee-freeze']
```

METHODE 2 : CHI2

Le test du khi-deux est un test statistique d'indépendance permettant de déterminer la dépendance de deux variables. Il partage des similitudes avec le coefficient de détermination, R^2. Cependant, le test du khi-deux ne s'applique qu'aux données catégoriques ou nominales, tandis que R^2 ne s'applique qu'aux données numériques

Le test du chi carré est utilisé pour les entités qualitatives dans un jeu de données. Nous calculons le Chi-carré entre chaque caractéristique et la cible et sélectionnons le nombre souhaité de caractéristiques avec les meilleurs scores du Chi-carré. Il détermine si l'association entre deux variables catégoriques de l'échantillon refléterait leur association réelle dans la population.

Le score de chi carré est donné par :

$$X^2 = \frac{(Observed\ frequency - Expected\ frequency)^2}{Expected\ frequency}$$

Le code python est le suivant :

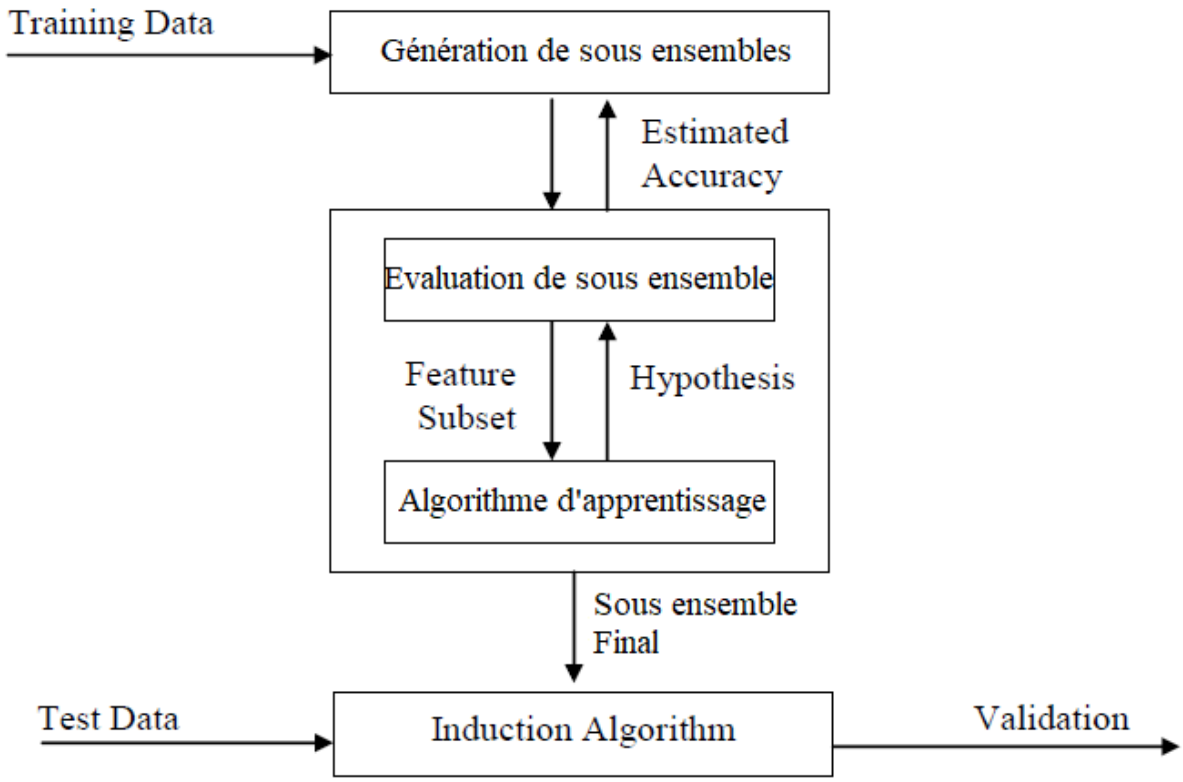
```
#We will select the features using chi square

test = SelectKBest(score_func=chi2, k=3)

#Fit the function for ranking the features by score

fit = test.fit(X_train, Y_train)

#Summarize scores

np.set_printoptions(precision=3)

print(fit.scores_)

#Apply the transformation on to dataset

features = fit.transform(X_train)

#Summarize selected features

print(features)
```

L’exécution de cette fonction donne le résultat suivant :

```
3 selected features
['education-spending', 'el-salvador-aid', 'physician-fee-freeze']
```

APPROCHE PAR WRAPPER

Dans les méthodes Wrapper les performances d'un algorithme d'apprentissage sont utilisés pour évaluer les sous ensemble des attributs. Le sous ensemble qui a le meilleur score selon l'algorithme d'apprentissage est sélectionné pour représenter tout le Data Set. Le schéma suivant résume les étapes de sélection dans l'approche Wrapper.



Dans un premier temps nous allons appliquer un algorithme de régression linéaire pour voir l'exactitude du modèle en considérant tous les attributs dans l'apprentissage. Puis nous utilisons deux méthodes pour obtenir une meilleure performance en réduisant le nombre d'attributs.

```
cols = list(X.columns)
X_1 = X[cols]
X_1 = sm.add_constant(X_1)
model = LinearRegression().fit(X=X_1, y=y)
score = model.score(X=X_1, y=y)
print(score)
```

L'algorithme de régression linéaire termine avec une exactitude de 90.42%.

METHODE 1 : ELIMINATION PAR PENALITE

Cette méthode consiste à calculer une pénalité pour chaque attribut, à chaque itération l'attribut de plus grande pénalité va être éliminé. La notion de pénalité est par rapport aux performances de l'algorithme d'apprentissage selon un attribut candidat.

Le code suivant montre une manière de le faire en acceptant que les attributs qui ont une pénalité < 5%. Le modèle utilisé est OLS (Ordinary Least Mean) qui est un algorithme de régression linéaire avec calcule de pénalité.

```
pmax = 1
while len(cols) > 0:
    p= []
    X_1 = X[cols]
    X_1 = sm.add_constant(X_1)
    model = sm.OLS(y.astype(float),X_1.astype(float)).fit()
    p = pd.Series(model.pvalues.values[1:], index=cols)
    pmax = max(p)
    print(pmax)
    feature_with_p_max = p.idxmax()
    if pmax > 0.05:
        cols.remove(feature_with_p_max)
    else:
        break
print(cols)
```

A la fin de l'exécution, l'algorithme retourne trois attributs :

- Adoption-of-the-budget-resolution
- Physician-fee-freeze
- Synfuels-corporation-cutback

En considérant ces trois attributs, nous allons évaluer l'exactitude de l'algorithme de régression linaire une autre fois :

```
 X_1 = X[cols]
X_1 = sm.add_constant(X_1)
model = LinearRegression().fit(X=X_1, y=y)
score = model.score(X=X_1, y=y)
print(score)
```

Le résultat de l'exécution donne une exactitude de **89.99%** ce qui est proche de la première valeur (sans élimination d'attribut) qui était 90.42%. Ceci est un bon gain puisqu'avec

seulement 3 attributs au lieu de 16 nous sommes arrivé à avoir presque la même exactitude et bien sûre avec un temps d'exécution plus rapide.

On peut jouer sur le paramètre de pénalité pour avoir de différents résultats, le tableau suivant donne les valeurs obtenues pour une multitude de value de pénalité à accepter :

Pénalité	Les attributs sélectionnés	L'exactitude
<100%	Tous les attributs	90.42%
<10%	'adoption-of-the-budget-resolution', 'physician-fee-freeze', 'immigration', 'synfuels-corporation-cutback', 'export-administration-act-south-africa'	90.19%
<5%	'adoption-of-the-budget-resolution', 'physician-fee-freeze', 'synfuels-corporation-cutback'	89.99%
<0.1%	'physician-fee-freeze', 'synfuels-corporation-cutback'	89.43%
<0.00001%	'physician-fee-freeze'	88.43%

On remarque qu'avec un seul attribut qui est 'physician-fee-freeze' on peut avoir une exactitude de 88.43% qui simplifie largement la représentation du Data Set sur un seul axe qui est 'physician-fee-freeze'. ***(Il est à noter que le Data Set d'apprentissage est réutilisé pour le test, cela influe les résultats des tests, dans la deuxième méthode nous allons trancher le Data Set ce qui va améliorer l'exactitude pendant le test).***

METHODE 2 : RECURSIVE FEATURES ELIMINATION

L'élimination récursive des attributs est l'une des meilleures méthodes, si vous avez par exemple un Data Set à n attributs et vous voulez le réduire à m attribut, le RFE va essayer de trouver toutes les décompositions possibles de l'ensemble des attributs en sous-ensembles de m attributs. L'algorithme est le suivant :

$RFE\ (liste\_attributs, m)$:

- $Si\ taille(liste\_attributs) = m\ alors\ renvoie\ liste\_attributs$
- $Sinon$
  - $Générer\ tous\ les\ sous\ ensembles\ de\ taille$: $[taille(liste_{attributs}) - 1]$, $soi - disant\ S_1, S_2, S_3 \dots S_{taille(liste\_attributs)-1}$.

- $Renvoie\ meilleur\ (RFE(S_1, m), RFE(S_2, m), \ldots, RFE\ (S_{dernier}, m))$

Dans l'implémentation nous allons dans un premier temps faire une boucle pour obtenir le meilleur nombre d'attributs que nous allons avoir puis exécuter RFE pour obtenir cet ensemble.

Avant de présenter le code en python pour obtenir le nombre optimal et aussi avoir l'attribut(s) le(s) plus pertinent(s), il est nécessaire de noter ceci :

Dans la méthode précédente nous avons évalué les attributs selon l'exactitude d'apprentissage seulement et puis nous avons utilisé un autre Data set pour le test ce qui peut donner des résultats différents par rapport à la valeur de l'exactitude. Néanmoins, dans cette méthode nous allons trancher le Data Set en données d'apprentissage et données de tests. Le nombre d'attributs sélectionnés dépend fortement de la manière de trancher le Data Set.

Le code de sélection est le suivant :

```
selected_features_BE = cols
print(selected_features_BE)

nof_list=np.arange(1,16)
high_score=0
nof=0
score_list =[]
for n in range(len(nof_list)):
    X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.6,
random_state = 0)
    model = LinearRegression()
    rfe = RFE(model,nof_list[n])
    X_train_rfe = rfe.fit_transform(X_train,y_train)
    X_test_rfe = rfe.transform(X_test)
    model.fit(X_train_rfe,y_train)
    score = model.score(X_test_rfe,y_test)
    score_list.append(score)
    if score > high_score:
        high_score = score
        nof = nof_list[n]

print("Nombre optimal: %d" %nof)
print("score: %f" %  high_score)

#liste des attributs
cols = list(X.columns)
model = LinearRegression()
```

```
#Initializing RFE model
rfe = RFE(model, nof)

X_rfe = rfe.fit_transform(X,y)
model.fit(X_rfe,y)
temp = pd.Series(rfe.support_, index=cols)
selected_features_rfe = temp[temp is True]
print(selected_features_rfe)
print(selected_features_BE)
```

Les résultats selon la taille du Data Set :

Taille de données de test	Nombre d'attribut	exactitude
20% des données	4	78.29%
40% des données	2	88.21%
60% des données	1	91.09%

CONCLUSION

Bien que les différentes approches s'accordent sur les variables les plus importantes de l'ensemble de données, elles ne concordent pas sur les points les plus fins. Il en résulte des sélections de colonnes parfois différentes, parfois très différentes, en fonction de l'algorithme utilisé qui dépend fortement du cas d'utilisation ainsi que jeu de données et des paramètres à passer à l'algorithme. Dans le cas ou le DATA SET contient un grand nombre d'attributs il est préférable d'utiliser les méthodes par filtrage parce qu'elles sont très rapides (souvent polynomiales) et donnent un résultat acceptable. Si on est exigeant par rapport à l'exactitude d'apprentissage et par rapport à la représentation du DATA SET, il faut travailler avec les méthodes Wrapper ou Embeded qui sont coûteuses en termes de temps d'exécution et des fois difficiles à paramétrer.

ANNEX : OUTILS DE TRAVAIL

Dans ce petit TP nous avons travaillé avec **Python 3.7**, les outils de développement sont :

- Jupyter Notebook
- Pycharm (Outil professionnel de JetBrains)

Les bibliothèques utilisées :

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE
from sklearn.linear_model import RidgeCV, LassoCV, Ridge, Lasso
from sklearn.ensemble import RandomForestClassifier
```