Ecole Nationale Supérieure d'Informatique

Module ML

Option SIL

Rapport du TP 4

Comparaison entre les approches (méthodes) de sélection d'attributs

« Feature Selection »

Réalisé par les étudiants

1. HEBBACHE Loucif

2. BOUZARA Aimene Hichem

Année universitaire 2018/2019

Table des matières

1. Introduction :

La sélection des attributs (Feature Selection) est un processus qui appartient à la phase de prétraitement des données, dans lequel on sélectionne automatiquement les attributs (entités) de nos données qui contribuent le plus à la variable de prédiction (la variable dépendante) ou à la sortie qui nous intéressent ;

La présence des attributs non pertinents dans nos données peut réduire la précision de nombreux modèles, en particulier les algorithmes linéaires tels que la régression linéaire et logistique.

La sélection des attributs avant l'entrainement du modèle de nos données présente trois avantages:

- Réduit les sur-ajustements: Moins de données redondantes signifie moins d'opportunités pour prendre des décisions basées sur le bruit.
- Améliore la précision: moins de données trompeuses signifie que la précision de la modélisation est améliorée.
- Réduit le temps d'apprentissage (entrainement): moins de données signifie que les algorithmes s'entraînent plus rapidement ;

On trouve 3 grandes approches qui sont les plus utilisées dans ce processus :

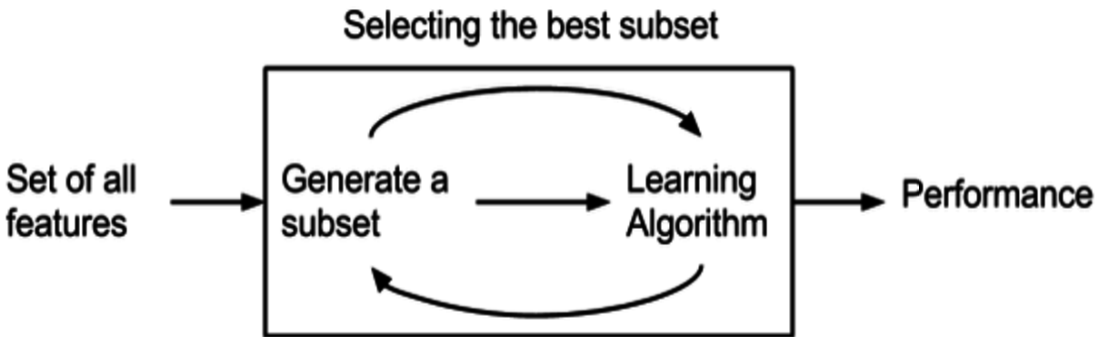
- **Approche enveloppante : Wrapper approach ;**
- **Approche filtre : Filter approach ;**
- **Approche intégrée : Embedded approach.**

Dans ce qui va suivre, on va présenter un exemple de méthode pratique de chaque approche citée précédemment appliqué sur le même dataset (house-votes-84.data) avec le même modèle d'entrainement (la régression logistique) afin de voir la précision (qualité) de chaque méthode ainsi son temps d'exécution (performances) et par conséquent conclure une comparaison.

2. Principales approches pour la résolution du problème de sélection d'attributs :

2.1 Approche enveloppante (wrapper) :

La sélection est partie prenante dans l'algorithme de classification ;



REF (Recursive Feature Elimination), sequential feature selection algorithms et genetic algorithms sont les méthodes les plus utilisées dans cette approche ; on a pris comme exemple pratique la méthode **REF** (**Recursive Feature Elimination**) ;
Cette dernière fonctionne en supprimant récursivement les attributs et en construisant un modèle sur les attributs restants ; Il utilise la précision du modèle pour identifier les attributs (et la combinaison d'attributs) qui contribuent le plus à la prédiction de l'attribut cible ; l'application de cette méthode dans notre programme donne :

In [157]:
```
lr = LogisticRegression(solver='lbfgs')
rfe = RFE(lr,4)
import time
start = time.time()
fit=rfe.fit(X_train,y_train)
end = time.time()
print("Execution time with Recursive feature elimination REF is: %f"%(float(end)- float(start)))
y_pred = fit.predict(X_test)
confusion_matrix(y_test,y_pred)
```

```
Execution time with Recursive feature elimination REF is: 0.228878
```

Out[157]:
```
array([[37,  0],
       [ 2, 31]], dtype=int64)
```

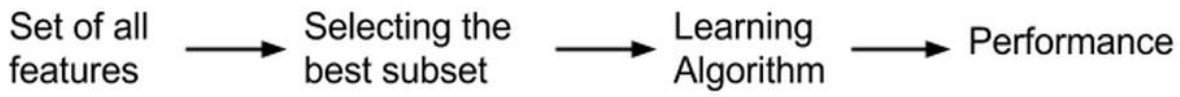
In [158]:
```
print("Num Features: ",fit.n_features_)
print("Selected Features:",fit.support_)
print("Feature Ranking: ", fit.ranking_)
print ("Accuracy with Recursive feature elimination is ", accuracy_score(y_test,y_pred)*100)
```

```
Num Features:  4
Selected Features: [False False  True  True False False False False False False  True False
 False  True False]
Feature Ranking:  [11  8  1  1  5  6  9 10  4  3  1  2 12  1  7]
Accuracy with Recursive feature elimination is  97.14285714285714
```

Donc la précision avec cette méthode est de 97.14 et les 4 attributs sélectionnés sont d'index : 2,3,10 et 13 (sont marqués par « True » dans selected feature et par « 1 » dans feature ranking) ; pour son temps d'exécution est de : 0.228 seconde !

2.2 Approche filtre (filter) :

Les attributs sont sélectionnés ensuite utilisées dans l'algorithme de classification ;



On trouve dans cette approche les méthodes : information gain, chi-square test, fisher score, correlation coefficient et variance threshold ; on a pris comme exemple pratique la méthode **chi-square test** ;on peut utiliser un autre type de test de la classe SelectKBest qui est offert par le framework sklearn pour sélectionner un nombre spécifique d'attributs ;cette méthode d'utiliser les tests statistiques pour la sélection d'attributs est appelée aussi « **Univariate Selection** » ; les résultats obtenues avec le programme sont les suivants :

```
numpy.set_printoptions(precision=3)
print(fit2.scores_)

X_trans=fit2.transform(X)
X_train_trans, X_test_trans, y_train_trans, y_test_trans = train_test_split(X_trans,y, test_size=0.3)


start = time.time()
lr.fit(X_train_trans,y_train_trans)
end = time.time()
print("Execution time with chi-square test is: %f"%(float(end)- float(start)))

y_pred = lr.predict(X_test_trans)
confusion_matrix(y_test_trans,y_pred)
```

```
[1.970e+01 5.316e-02 5.296e+01 1.052e+02 5.918e+01 1.637e+01 2.674e+01
 5.242e+01 5.029e+01 1.829e-01 2.058e+01 6.479e+01 3.216e+01 3.621e+01
 3.673e+01]
Execution time with chi-square test is: 0.015992
```

Out[174]:
```
array([[30,  2],
       [ 1, 37]], dtype=int64)
```

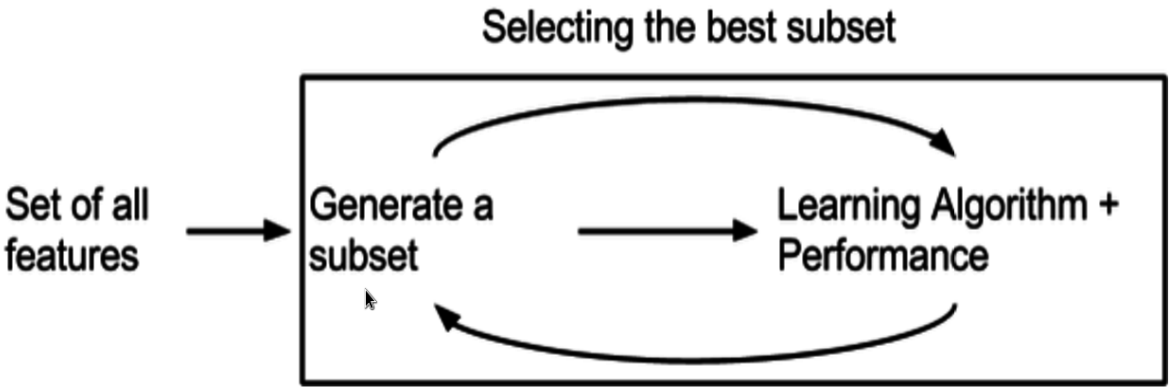
In [175]:
```
print ("Accuracy with chi-square test is ", accuracy_score(y_test_trans,y_pred)*100)
```

```
Accuracy with chi-square test is  95.71428571428572
```

La précision est : 95.71 et les 4 attributs sélectionnés sont d'index : 2,4,7 et 11 (sont ceux qui ont un score grand); pour son temps d'exécution est de : 0.0159 seconde !

2.3 Approche intégrée (embedded) :

Un algorithme de classification est appliqué sur l'ensemble de données afin d'identifier les meilleures caractéristiques ;



Y'a 2 méthodes qui peuvent être classées dans cette catégorie d'approche : L1 (LASSO) regularization et **decision tree** ; on a utilisé cette dernière dans notre exemple ; les arbres de décisions sont utilisés à l'origine pour la classification ; mais on peut en profiter pour estimer le poids (l'importance) des attributs afin de sélectionner les plus importants ; le programme nous a donné :

In [180]:
```
#3ème expérience : calculer la précision et le temps d'éxécution avec la méthode decision tree pour la séléction d'attributs
et = ExtraTreesClassifier()
et.fit(X, y)
print(et.feature_importances_)

sfm = SelectFromModel(et, threshold=0.01)
fit3=sfm.fit(X,y)

X_trans=fit3.transform(X)
X_train_trans, X_test_trans, y_train_trans, y_test_trans = train_test_split(X_trans,y, test_size=0.3)

start = time.time()
lr.fit(X_train_trans, y_train_trans)
end = time.time()
print("Execution time with decision tree is: %f"%(float(end)- float(start)))
print("Shape of the dataset ",shape)


pre = lr.predict(X_test_trans)
print ("Accuracy with tree based feature selection is ", accuracy_score(y_test_trans,pre)*100)
```

```
[0.011 0.02  0.038 0.367 0.128 0.007 0.036 0.062 0.016 0.013 0.053 0.096
 0.016 0.098 0.038]
Execution time with decision tree is: 0.015988
Shape of the dataset  (162, 11)
Accuracy with tree based feature selection is  97.14285714285714
```

La précision est la même avec celle trouvée avec **REF** : 97.14 et les 4 attributs sélectionnés sont d'index : 3,4,11 et 13 (sont ceux qui ont un score grand); pour son temps d'exécution est de : 0.0159 seconde ! (Le même trouvé avec **chi-square test)➔cette méthode est un compromis entre les 2 précédentes ;**

2.4 Sans la sélection des attributs :

Cette fois, si on applique directement l'algorithme d'apprentissage sans aucune sélection, on trouve

In [187]:
```
#4eme expérience : calculer la précision et le temps d'éxécution sans la séléction d'attributs
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size = 0.3)

start = time.time()
fit=lr.fit(X_train,y_train)
end = time.time()
print("Execution time without any feature selection is: %f"%(float(end)- float(start)))
y_pred = fit.predict(X_test)
confusion_matrix(y_test,y_pred)
```

```
Execution time without any feature selection is: 0.033982
```

Out[187]:
```
array([[36,  3],
       [ 0, 31]], dtype=int64)
```

In [188]:
```
print ("Accuracy without any feature selection is ", accuracy_score(y_test,y_pred)*100)
```

```
Accuracy without any feature selection is  95.71428571428572
```

In [ ]:

La précision est la même avec celle de **chi-square test**: 95.71 ; pour son temps d'exécution est de : 0.0339 seconde !

3. Conclusion : Récapitulatif :

	Qualité de classification (la précision)	Performance (le temps d'exécution)
Sans sélection	**95.71**	0.0339
Avec sélection : méthode REF(wrapper approach)	**97.14**	0.0159
Avec sélection : méthode chi-square test(filter approach)	**95.71**	0.0159
Avec sélection : méthode decision tree (embedded approach)	**97.14**	0.228

D'après ce qui précède, on trouve que **decision tree (embedded approach)** est le compromis idéal dans cet exemple ;
En générale, la sélection des attributs améliore les 2 facteurs (qualité et performance) **mais ce n'est pas toujours le cas**, tout dépend des données traitées (dataset);