Feature selection

Obsah

Preco by som mal vyberat len niektore atributy?

- redundancia - skryte zavyslosti medzi nimi
- irelevancia - nemusia mat ziadny vplyv na predikovanu hodnotu
- pretrenovanie - model sa da natrenovat aj na nahodnych datach a na trenovacej sade bude fungovat. Na testovacej ale bude fungovat uplne strasne
- prekliatie dimenzionality - pri velkom pocte atributov potrebujem vela dat na to aby som dostatocne pokryl priestor moznych hodnot
- produktivita / rychlost - moja ako analytika a aj mojich modelov (trenovanie aj predikcia)
- zrozumitelnost - lahsie sa vysvetluje model, ktory ma menej atributov

In [24]:
```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn

plt.rcParams['figure.figsize'] = 9, 6
```

Filter

Vyber atributov bez ohladu na model, ktory sa chystame trenovat.

- rychle
- nezavisle na modeli (to je dobre aj zle)

Najjednoduchsie je vyhodit atributy, ktore maju vsade rovnake hodnoty

pozor, nie malu varianciu. Hlavne pri nevyvazenych triedach mozu byt prave taketo atributy velmi uzitocne

In [25]:
```
from sklearn.feature_selection import VarianceThreshold
```

In [26]:
```
X = np.array([[0, 2, 0, 3], [0, 1, 4, 3], [0, 1, 1, 3]])
X
```

Out[26]:
```
array([[0, 2, 0, 3],
       [0, 1, 4, 3],
       [0, 1, 1, 3]])
```

In [27]:
```
selector = VarianceThreshold(threshold=0.0)
selector.fit_transform(X)
```

Out[27]:
```
array([[2, 0],
       [1, 4],
       [1, 1]])
```

Potom mozeme vyberat atributy na zaklade zavislosti atributu a predikovanej hodnoty

Priklad: vyberieme K vlastnosti, s najvyssou zavislostou s predikovanou hodnotou.

In [28]:
```
from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2 # daju sa pouzit ine metriky
iris = load_iris()
X, y = iris.data, iris.target
X.shape
```

Out[28]: (150, 4)

In [29]:
```
X_new = SelectKBest(chi2, k=2).fit_transform(X, y)
X_new.shape
```

Out[29]: (150, 2)

Daju sa pouzivat rozne metriky

Klasifikacia

- chi2 - nezaporne cisla
- mutual_info_classif - diskretne data
- f_classif - ANOVA medzi predikovanou premennou a atributmi

Regresia

- f_regression - F test medzi predikovanou hodnotou a atributmi
- mutual_info_regression - Mutual information na realnych cislach

Da sa vyberat K najlepsich alebo nejaky percentil alebo nechat pocet atributov na statisticky test

- SelectKBest
- SelectPercentile
- SelectFpr - false positive rate
- SelectFdr - false discovery rate
- SelectFwe - family wise error
- GenericUnivariateSelect - Vsetko dohromady a strategia sa da nastavit parametrom

Vlastnosti filtrov

- vacsinou rychle
- nezavisle na modely (nepotrebujem opakovane trenovat model ale vybrane atributy nemusia byt najvhodnejsie pre kazdy model)
- vacsinou sa pozeraju len na vlastnosti dvojic predikovana premenna - atribut, kombinacie viacerych atributov nezohladnuju

Varovanie, PCA sa casto pouziva na redukciu dimenzionality ale nie na vyber atributov

Je to casta chyba

Nemohol som si odpustit tuto poznamku

Preco je to tak sa mozeme porozpravat v diskusii

Wrapper

Zakladna myslienka

haldame podmnozinu atributov, na ktorej bude model davat najlepsie vysledky

Skusame rozne podmnoziny a vyberame tu najlepsiu

Problem

Ak mame dataset s N atributmi, tak pocet roznych podmnozin je 2^N

To znamena, ze by sme museli nas model natrenovat 2^N krat.

Chcelo by to najst proces, ktory minimalizuje pocet pokusov a zaroven maximalizuje uspesnost modelu

Greedy pristupy

Najcastejsie sa pouzivaju greedy pristupy, ktore postupne zvacsuju sadu atributov (alebo zmensuju) tak, ze pridavaju (odoberaju) atribut tak aby sa co najviac zvysila uspesnost.

Mlxtend

- Sequential Forward Selection (SFS)

  > Postupne zvacsuje mnozinu atributov o ton, ktory najviac prispel k zlepseniu

- Sequential Backward Selection (SBS)

  > Postupne zmensuje mnzoinu atributov o ten, ktory najmenej pomahal

- Sequential Floating Forward Selection (SFFS)

  > SFS s pokusom o vyhodenie uz pridanych atributov ak sa ukaze ze velmi nepomahaju

- Sequential Floating Backward Selection (SFBS)

  > SBS s pokusom o pridanie uz raz vyhodeneho atributu

Scikit-Learn

- RFE - Recursive feature elimination

  > Postupne vyhadzovanie atributov, ktore maju v modeli najnizsiu vahu (potrebujeme aby to model vedel povedat)

- RFECV - RFE with cross-validation

RFE s cross validaciou

Priklad SFS

In [30]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris

iris = load_iris()
X = iris.data
y = iris.target
knn = KNeighborsClassifier(n_neighbors=4)
```

In [31]:
```
from mlxtend.feature_selection import SequentialFeatureSelector as SFS

sfs1 = SFS(knn, k_features=3, forward=True,  floating=False, verbose=2, scoring='accuracy', cv=0)
# pomocou tejto triedy vieme robit SFS, SFFS, SBS aj SFBS a dokonca aj pridat cross-validaciu

sfs1 = sfs1.fit(X, y)
```

```
[2017-04-06 10:29:59] Features: 1/3 -- score: 0.96
[2017-04-06 10:29:59] Features: 2/3 -- score: 0.973333333333
[2017-04-06 10:29:59] Features: 3/3 -- score: 0.973333333333
```

In [32]:
```
sfs1.subsets_
```

Out[32]:
```
{1: {'avg_score': 0.95999999999999996,
  'cv_scores': array([ 0.96]),
  'feature_idx': (3,)},
 2: {'avg_score': 0.97333333333333338,
  'cv_scores': array([ 0.97333333]),
  'feature_idx': (2, 3)},
 3: {'avg_score': 0.97333333333333338,
  'cv_scores': array([ 0.97333333]),
  'feature_idx': (1, 2, 3)}}
```

Viem si vytiahnut zoznam najlepsich vlastnosti a uspesnost modelu pri nich

In [33]:
```
sfs1.k_feature_idx_
```

Out[33]: (1, 2, 3)

In [34]:
```
sfs1.k_score_
```

Out[34]: 0.97333333333333338

Embedded

Hlavna myslienka

Skombinovat vyhody filtrov a wrapprov

Model, ktory sa trenuje si bude priamo vyberat atributy, ktore su pre neho najlepsie

- Linearne modely penalizovnae L1 (Lasso) alebo L1+L2 (Elastic Net) regularizaciou: SVM, Linearna regresia, Logisticka regresia ...
- RandomForest

In [35]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectFromModel
iris = load_iris()
X, y = iris.data, iris.target
X.shape
```

Out[35]: (150, 4)

In [36]:
```
clf = RandomForestClassifier()
clf = clf.fit(X, y)
clf.feature_importances_
```

Out[36]: array([0.15331839, 0.01613406, 0.43087666, 0.39967089])

In [37]:
```
model = SelectFromModel(clf, prefit=True)
X_new = model.transform(X)
X_new.shape
```

Out[37]:
```
(150, 2)
```

Zaver

Zvazte ktory sposob vyberu atributov sa hodi prave pre vas. Zalezi to hlavne od pouziteho algoritmu na vytvorenie modelu.

- Ak pouzivate nejaky linearny model alebo les, tak je zbytocne robit filtre a este viac zbytocne robit wrappre.
- Ak nemate cas na opakovane trenovanie modelu, tak filtre mozu byt dostatocny hotfix. Treba ale zvazit aku vlastnost atributov chcete pouzit na najdenie najdolezitejsich.
- Ak mate cas spustit to trenovanie viac krat, tak asi najlepsia moznost je SFFS alebo SFECV

Zdroje

- http://scikit-learn.org/stable/modules/feature_selection.html (http://scikit-learn.org/stable/modules/feature_selection.html)
- http://jotterbach.github.io/2016/03/24/Principal_Component_Analysis/ (http://jotterbach.github.io/2016/03/24/Principal_Component_Analysis/)
- https://plot.ly/scikit-learn/plot-feature-selection/ (https://plot.ly/scikit-learn/plot-feature-selection/)
- https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/ (https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/)
- http://www.kdnuggets.com/2017/04/must-know-fewer-predictors-machine-learning-models.html?utm_content=buffer42ed6&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer (http://www.kdnuggets.com/2017/04/must-know-fewer-predictors-machine-learning-models.html?utm_content=buffer42ed6&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer)

In [ ]:
```
```