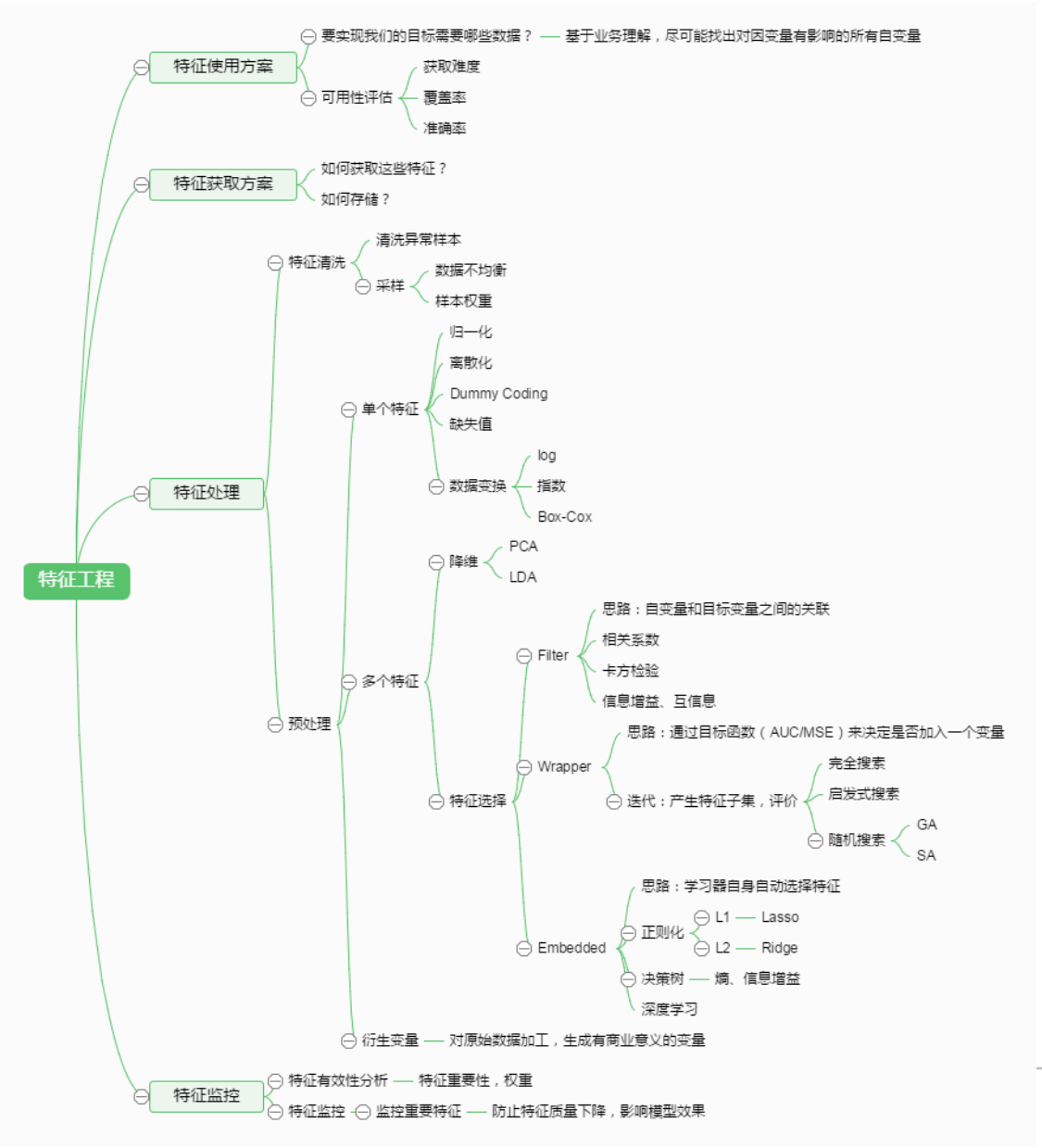
Weekend2：特征选择

CSDN学院


特征工程
特征使用方案
要实现我们的目标需要哪些数据？ — 基于业务理解，尽可能找出对因变量有影响的所有自变量
可用性评估
获取难度
覆盖率
准确率
特征获取方案
如何获取这些特征？
如何存储？
特征处理
特征清洗
清洗异常样本
采样
数据不均衡
样本权重
预处理
单个特征
归一化
离散化
Dummy Coding
缺失值
数据变换
log
指数
Box-Cox
多个特征
降维
PCA
LDA
特征选择
Filter
思路：自变量和目标变量之间的关联
相关系数
卡方检验
信息增益、互信息
Wrapper
思路：通过目标函数（AUC/MSE）来决定是否加入一个变量
迭代：产生特征子集，评价
完全搜索
启发式搜索
随机搜索
GA
SA
Embedded
思路：学习器自身自动选择特征
正则化
L1 — Lasso
L2 — Ridge
决策树 — 熵、信息增益
深度学习
衍生变量 — 对原始数据加工，生成有商业意义的变量
特征监控
特征有效性分析 — 特征重要性，权重
特征监控 — 监控重要特征 — 防止特征质量下降，影响模型效果

特征选择

- 原因
  - 冗余：部分特征之间相关度太高
  - 噪声
- 特征选择 vs.降维
  - 降维（PCA/SVD）：对原始特征进行线性组合，得到低维表示
  - 特征选择：剔除与预测关系不大的特征子集
    - 减少特征数目，防止过拟合，提高模型泛化能力
    - 有助于更好地理解特征的特性及其与响应之间的关系

特征选择

sklearn.feature_selection

- http://scikit-learn.org/stable/modules/feature_selection.html
- 特征选择实现方法
  - 删除方差过小的特征
  - 过滤型(Fillter)
  - 包裹型(Wrapper)
  - 嵌入型
    - 正则化方法（L1）
    - feature_importance_

删除方差过小的特征

- 方差小意味着变化很小（标准差小）意味着该特征包含信息少，特征区分度很低

- *class* sklearn.feature_selection.VarianceThreshold(*threshold=0.0*)
  - 删除方差小于*threshold*的特征
  - *threshold* 缺省值为0，移除所有方差为0的特征，即那些在所有样本中数值完全相同的特征

▶例：风控算法大赛解决方案

2.2 剔除常变量

金融风控大赛解决方案.pdf

原始数据中有 190维数值型特征，通过计算每个数值型特征的标准差，剔除部分变化很小的特征，下表（表1）列出的15个特征是标准差接近于0的，我们剔除了这15维特征。

表 1.剔除数值特征标准差

属性	标准差	属性	标准差	属性	标准差
WeblogInfo_10	0.0707	WeblogInfo_41	0.0212	WeblogInfo_49	0.0071
WeblogInfo_23	0.0939	WeblogInfo_43	0.0372	WeblogInfo_52	0.0512
WeblogInfo_31	0.0828	WeblogInfo_44	0.0166	WeblogInfo_54	0.0946
WeblogInfo_32	0.0834	WeblogInfo_46	0.0290	WeblogInfo_55	0.0331
WeblogInfo_40	0.0666	WeblogInfo_47	0.0401	WeblogInfo_58	0.0609

Recall：Pandas 的describe()函数可得到各数值型变量的标准差

过滤型(Fillter)

- 过滤型特征选择根据单变量的统计测试分数选择最佳特征
  - sklearn.feature_selection.SelectKBest：选择分数最高的k个特征
  - sklearn.feature_selection.SelectPercentile:选择分数最高的百分比特征
  - 缺点：没有考虑到特征之间的关联作用，可能把有用的关联特征误踢掉

- 例： sklearn.feature_selection.SelectKBest(*score_func*=<*function f_classif*>, *k=10*)
  - 分数度量*score_func*：
    - 相关系数
      - 相关系数只对线性关系敏感，不太能发现非线性相关
      - 相关系数只对两个实数型随机变量有定义
    - 互信息 / 最大信息系数：通常用于分类（y为离散值）
    - chi2: 非负特征与分类标签之间的χ^2统计量
    - …

互信息

- 度量联合分布 $p(X,Y)$ 和因式分解形式 $p(X)p(Y)$ 之间的不相似度（KL散度）
  - 越不相似表示X、Y之间越不独立，即越相关

$$\mathbb{I}(X;Y) \triangleq \mathbb{KL}(p(X,Y)||p(X)p(Y)) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

最大信息系数

函数minepy提供了MIC功能

- 连续变量的互信息计算不方便
  - 需先离散化（而互信息的结果对离散化的方式敏感）
- 最大信息系数（**maximal information coefficient**，MIC）
  寻找最优的离散化方式，并将互信息取值转换成到 [0，1]

$$m(x,y)=\frac{\max_{G\in\mathcal{G}(x,y)}\mathbb{I}(X(G);Y(G))}{\log\min(x,y)},\ \mathrm{MIC}\triangleq\max_{x,y:xy<B}m(x,y)$$

  - 其中 $X(G);Y(G)$ 为某种离散方式
- 例：$y=x^2$，MIC算出来的互信息值为1(最大的取值)。

最大信息系数（cont.）

工具包minepy实现了最大信息系数计算

```
from sklearn.feature_selection import SelectKBest
from minepy import MINE

#由于MINE的设计不是函数式的，定义mic方法将其为函数式的，返回一个二元组，二元组的第2项设置成固定的P值0.5
def mic(x, y):
    m = MINE()
    m.compute_score(x, y)
    return (m.mic(), 0.5)

#选择K个最好的特征，返回特征选择后的数据
SelectKBest(lambda X, Y: array(map(lambda x:mic(x, Y), X.T)).T, k=2).fit_transform(X_train, y_train)
```

https://github.com/minepy/minepy

Scikit learn实现


```
from sklearn.datasets import load_iris   #数据集
from sklearn.feature_selection import SelectKBest          #特征选择方法
from sklearn.feature_selection import chi2 #度量指标

iris = load_iris()
X, y = iris.data, iris.target
X.shape   #原始数据集大小：(150, 4)
X_new = SelectKBest(chi2, k=2).fit_transform(X, y)
X_new.shape          #特征选择后数据集大小：(150, 2)，只选择与y最相关的两维特征
```

包裹型(Wrapper)

- 递归特征删除算法
  - ①用所有特征跑一个模型，得到特征系数（如线性模型系数）或特征重要性（如xgboost）
  - ② 根据特征的系数或特征重要性，每次删除最不重要的（5-10%）的特征，观察模型性能变化，直至模型性能出现大的下滑或达到特征数目

Scikit learn实现

- 递归特征淘汰
  - sklearn.feature_selection.RFE(*estimator*, *n_features_to_select=None*, *step=1*, *verbose=0*)
  - sklearn.feature_selection.RFECV(*estimator*, *step=1*, *cv=None*, *scoring=None*, *verbose=0*, *n_jobs=1*)

- 给定一个外部的estimator，为feature分配权重（例如：线性模型的权重系数coefficients、GBDT中的特征重要性_feature_importance），递归特征淘汰
  - 1. 初始化当前feature集：所有特征
  - 2. estimator在当前feature集上进行训练，为每个feature分配权重
  - 3. 将绝对权重最小的features将从当前集中移除
  - 4. 重复2-3过程，直到features数达到目标期望值

- RFECV在一个cross-validation循环上执行RFE查找最优的feature数目。

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression

boston = load_boston()
X = boston["data"]
Y = boston["target"]
names = boston["feature_names"]

#use linear regression as the model
lr = LinearRegression()
#rank all features, i.e continue the elimination until the last one
rfe = RFE(lr, n_features_to_select=1)
rfe.fit(X,Y)

print "Features sorted by their rank:"
print sorted(zip(map(lambda x: round(x, 4), rfe.ranking_), names))
```

特征贡献的排序（最佳特征排在最前面）

```
Features sorted by their rank:
[(1.0, 'NOX'), (2.0, 'RM'), (3.0, 'CHAS'), (4.0, 'PTRATIO'), (5.0, 'DIS'), (6.0,
'LSTAT'), (7.0, 'RAD'), (8.0, 'CRIM'), (9.0, 'INDUS'), (10.0, 'ZN'), (11.0,
'TAX'), (12.0, 'B'), (13.0, 'AGE')]
```

嵌入型

- 模型训练后有系数或特征重要性属性的学习器的模型可用。

- 基于L1正则的特征选择
  - L1正则得到的系数可能是稀疏的，选择非0系数即可实现特征选择
    - Lasso、Logistic回归（L1正则）、SVM（L1正则）
- 基于树的特征选择
  - 基于树的模型（CART、随机森林、GBDT等）可以计算特征的重要性，从而可以根据重要性只选择重要的特征子集

基于L1正则特征选择

```
from sklearn.svm import LinearSVC     #分类器
from sklearn.datasets import load_iris    #数据集
from sklearn.feature_selection import SelectFromModel      #特征选择方式

iris = load_iris()
X, y = iris.data, iris.target
X.shape                  #(150, 4)

lsvc = LinearSVC(C=0.01, penalty="l1", dual=False).fit(X, y)          #L1正则
model = SelectFromModel(lsvc, prefit=True)        #特征选择实例
X_new = model.transform(X)                                #特征选择

X_new.shape          #选择了3维特征：(150, 3)
```

基于特征重要性的特征选择

例：用xgboost特征重要性做特征选择

```
from numpy import sort
from sklearn.feature_selection import SelectFromModel
from xgboost import XGBClassifier

#用全体特征训练xgboost分类器，得到特征重要性
model_XGB=XGBClassifier()
model_XGB.fit(X_train,y_train)

thresholdthresholds = sort(model_XGB.feature_importances_)                #以特征重要性作为候选阈值
for thresh in thresholds:  # select features using threshold
    selection = SelectFromModel(model_XGB, threshold=thresh, prefit=True)  #根据给定阈值选择特征集合
    select_X_train = selection.transform(X_train)

    selection_model = XGBClassifier()
    selection_model.fit(select_X_train, y_train) # 根据选择的特征重新训练分类器
    select_X_test = selection.transform(X_test)
    y_pred = selection_model.predict(select_X_test)

    predictions = [round(value) for value in y_pred]
    accuracy = accuracy_score(y_test, predictions)
    print("Thresh=%.3f, n=%d, Accuracy: %.2f%%" % (thresh, select_X_train.shape[1],      accuracy*100.0))
```

Mushroom_feature_selection.ipynb

如何获取重要特征

- 相关领域的专家知识
  - 如电商场景中加入购物车第二天被购买
- 深度学习自动学习特征
  - 如用CNN网络提取图像特征
- 实验、经验与发现
  - 不断尝试新的特征

CSDN
不 止 于 代 码

THANK YOU

AI100