【华子】机器学习000-特征工程之特征抽取 -

(【本文所使用的Python库和版本号】: Python 3.5, Numpy 1.14, scikit-learn 0.19, matplotlib 2.2)

特征工程包含内容

- 特征抽取
- 特征预处理
- 特征选择
- 特征降维

特征抽取

- 字典特征提取(特征离散化)
- 文本特征提取
- 图像特征提取（暂不介绍）

一：字典特征提取

```
import pandas as pd
from sklearn.feature_extraction import DictVectorizer
dv = DictVectorizer(sparse=False) #不产生稀疏矩阵
data = dv.fit_transform([{'city': '北京','temperature':100},
                         {'city': '上海','temperature':60},
                         {'city': '深圳','temperature':30}])
print(data) #提取后
result = dv.inverse_transform([[1,0,0,400]]) #解码
print(result)
```

```
[[  0.   1.   0. 100.]
 [  1.   0.   0.  60.]
 [  0.   0.   1.  30.]]
[{'city=上海': 1, 'temperature': 400}]
```

二：文本特征提取

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
#1.提取english内容的特征

data = ["life is short,i like python","life is too long,i dislike python"]
```

```
cv = CountVectorizer()
data = cv.fit_transform(data)
print(data.toarray())  #toarray()
print(cv.get_feature_names())
print('---'*10)
# 2.提取中文内容的特征
def cutword():
    # 调用jieba.lcut处理
    contetn1 = jieba.lcut("我学会了python，java和php")
    contetn2 = jieba.lcut("python包括flask，django和数据库")
    contetn3 = jieba.lcut("java和php要求熟练掌握数据库")
    # 将分词后内容连接成一个串，空格隔开
    c1 = ' '.join(contetn1)
    c2 = ' '.join(contetn2)
    c3 = ' '.join(contetn3)
    return c1, c2, c3

def getFeature():
    # 实例化conunt，并设置停用词
    count = CountVectorizer(stop_words=['要求','学会','包括','熟练掌握'])
    # 定义一个分词的函数
    c1, c2, c3 = cutword()
    data = count.fit_transform([c1, c2, c3])
    # 打印特征
    print(count.get_feature_names())
    # 特征抽取结果
    print(data.toarray())
getFeature()
```

```
[[0 1 1 1 0 1 1 0]
 [1 1 1 0 1 1 0 1]]
['dislike', 'is', 'life', 'like', 'long', 'python', 'short', 'too']
------------------------------
['django', 'flask', 'java', 'php', 'python', '数据库']
[[0 0 1 1 1 0]
 [1 1 0 0 1 1]
 [0 0 1 1 0 1]]
```

Tf-idf文本特征提取

- TF-IDF的主要思想是：如果**某个词或短语在一篇文章中出现的概率高，并且在其他文章中很少出现**，则认为此词或者短语具有很好的类别区分能力，适合用来分类。
- TF-IDF作用：用以评估字词对于一个文件集中一份文件的重要程度。

公式：

- 词频（term frequency，tf）指的是某一个给定的词语在该文件中出现的频率
- 逆向文档频率（inverse document frequency，idf）是一个词语普遍重要性的度量。某一特定词语的idf，可以**由总文件数目除以包含该词语之文件的数目，再将得到的商，取以10为底的对数得到**

$$\mathrm{tfidf}_{i,j} = \mathrm{tf}_{i,j} \times \mathrm{idf}_i$$

最终得出结果可以理解为重要程度。

> 注：假如一篇文件的总词语数是100个，而词语"非常"出现了5次，那么"非常"一词在该文件中的词频就是5/100=0.05。而计算文件频率（IDF）的方法是以文件集的文件总数，除以出现"非常"一词的文件数。所以，如果"非常"一词在1,000份文件出现过，而文件总数是10,000,000份的话，其逆向文件频率就是lg（10,000,000 / 1,0000）=3。最后"非常"对于这篇文档的tf-idf的分数为0.05 * 3=0.15

```python
# tf-idf文本特征提取
from sklearn.feature_extraction.text import TfidfVectorizer
def tfidfvec():
    # 实例化提取器
    tfidf = TfidfVectorizer()
    # 调用分词函数
    c1, c2, c3 = cutword()
    data = tfidf.fit_transform([c1, c2, c3])
    # 特征名
    print(tfidf.get_feature_names())
    # 特征提取后
    print(data.toarray())
tfidfvec()
```

```
['django', 'flask', 'java', 'php', 'python', '包括', '学会', '数据库', '熟练掌握', '要求']
[[0.         0.         0.45985353 0.45985353 0.45985353 0.
  0.60465213 0.         0.         0.        ]
 [0.49047908 0.49047908 0.         0.         0.37302199 0.49047908
  0.         0.37302199 0.         0.        ]
 [0.         0.         0.3935112  0.3935112  0.         0.
  0.         0.3935112  0.51741994 0.51741994]]
```

思考：将爬取到的职位名称进行特征抽取？打标签并进行回归分析？