极客时间笔记3 数据挖掘的过程

原创

任任任任小娇在学python的路上 于 2020-05-25 16:42:26 发布 47 收藏

文章标签： 数据挖掘

本文链接：https://blog.csdn.net/qq_42907167/article/details/106325951

1 数据挖掘的过程

数据挖掘的过程可以分成以下 6 个步骤。

> 商业理解：数据挖掘不是我们的目的，我们的目的是更好地帮助业务，所以第一步我们要从商业的角度理解项目需求，在这个基础上，再对数据挖掘的目标进行定义
>
> 数据理解：尝试收集部分数据，然后对数据进行探索，包括数据描述、数据质量验证等。这有助于你对收集的数据有个初步的认知。
>
> 数据准备：开始收集数据，并对数据进行清洗、数据集成等操作，完成数据挖掘前的准备工作。
>
> 模型建立：选择和应用各种数据挖掘模型，并进行优化，以便得到更好的分类结果。
>
> 模型评估：对模型进行评价，并检查构建模型的每个步骤，确认模型是否实现了预定的商业目标。
>
> 上线发布：模型的作用是从数据中找到金矿，也就是我们所说的“知识”，获得的知识需要转化成用户可以使用的方式，呈现的形式可以是一份报告，也可以是实现一个比较复杂的、可重复的数据挖掘过程。

数据挖掘结果如果是日常运营的一部分，那么后续的监控和维护就会变得重要

Apriori 是一种挖掘关联规则（association rules）的算法，它通过挖掘频繁项集（frequent item sets）来揭示物品之间的关联关系，被广泛应用到商业挖掘和网络安全等领域中。频繁项集是指经常出现在一起的物品的集合，关联规则暗示着两种物品之间可能存在很强的关系

运行结果：

```
['a', 'b', 'c', 'd']
4
['mm', 'a', 'b', 'c']
```

列表是 Python 中常用的数据结构，相当于数组，具有增删改查的功能，我们可以使用 len() 函数获得 lists 中元素的个数；使用 append() 在尾部添加元素，使用 insert() 在列表中插入元素，使用 pop() 删除尾部的元素。

元组 (tuple)

```
tuples = ('tupleA','tupleB')
print tuples[0]
```

运行结果：

```
tupleA
```

元组 tuple 和 list 非常类似，但是 tuple 一旦初始化就不能修改。因为不能修改所以没有 append(), insert() 这样的方法，可以像访问数组一样进行访问，比如 tuples[0]，但不能赋值。

Python基础语法，字典

```
# -*- coding: utf-8 -*
#定义一个dictionary
score = {'guanyu':95,'zhangfei':96}
#添加一个元素
score['zhaoyun'] = 98
print score
#删除一个元素
score.pop('zhangfei')
#查看key是否存在
print 'guanyu' in score
#查看一个key对应的值
print score.get('guanyu')
print score.get('yase',99)
```

python 求结构体数组的最大值

```
# -*- coding=utf-8 -*-
import numpy as np
from functools import cmp_to_key

persontype = np.dtype({ 'names':['name', 'chinese', 'english', 'math'], 'formats':['S32', 'i', 'i', 'f']})
peoples = np.array([("ZhangFei",60,65,30),("ZhangFei",95,85,98), ("ZhangFei",93,92,96),("HuangZhong",90,88,77),(
"Dianwei",80,90,90)], dtype=persontype)
print(peoples.dtype)
Chineses = peoples[:]['chinese']
english=peoples[:]['english']
math=peoples[:]['math']
summary=[]
for i, person in enumerate(peoples):
    person = list(person)
    # print(person[:])
    total = (np.sum(person[1:]))
    # print(total)
    summary = np.append(summary, [total], axis=0)
print(summary)
```

Series 和 DataFrame 两个核心数据结构，

他们分别代表着一维的序列和二维的表结构。基于这两种数据结构，Pandas 可以对数据进行导入、清洗、处理、统计和输出。

Series 有两个基本属性：index 和 values。
在 Series 结构中，index 默认是 0,1,2,……递增的整数序列，当然我们也可以自己来指定索引，比如 index=[‘a’, ‘b’, ‘c’, ‘d’]。

```
import pandas as pd
from pandas import Series, DataFrame
x1 = Series([1,2,3,4])
x2 = Series(data=[1,2,3,4], index=['a', 'b', 'c', 'd'])
print x1
print x2
```

输出结果为：

```
0    1
1    2
2    3
3    4
dtype: int64
a    1
b    2
c    3
d    4
dtype: int64
```

数据间的空格

有时候我们先把格式转成了 str 类型，是为了方便对数据进行操作，这时想要删除数据间的空格，我们就可以使用 strip 函数：

```
#删除左右两边空格
df2['Chinese']=df2['Chinese'].map(str.strip)
#删除左边空格
df2['Chinese']=df2['Chinese'].map(str.lstrip)
#删除右边空格
df2['Chinese']=df2['Chinese'].map(str.rstrip)
```

如果数据里有某个特殊的符号，我们想要删除怎么办？同样可以使用 strip 函数，比如 Chinese 字段里有美元符号，我们想把这个删掉，可以这么写：

```
df2['Chinese']=df2['Chinese'].str.strip('$')
```


1. 删除 DataFrame 中的不必要的列或行

Pandas 提供了一个便捷的方法 drop() 函数来删除我们不想要的列或行。比如我们想把“语文”这列删掉。

```
df2 = df2.drop(columns=['Chinese'])
```


想把“张飞”这行删掉。

```
df2 = df2.drop(index=['ZhangFei'])
```


2. 重命名列名 columns，让列表名更容易识别

如果你想对 DataFrame 中的 columns 进行重命名，可以直接使用 rename(columns=new_names, inplace=True) 函数，比如我把列名 Chinese 改成 YuWen，English 改成 YingYu。

```
df2.rename(columns={'Chinese': 'YuWen', 'English': 'Yingyu'}, inplace = True)
```

3. 去重复的值

数据采集可能存在重复的行，这时只要使用 drop_duplicates() 就会自动把重复的行去掉。

```
df = df.drop_duplicates() #去除重复行
```


4. 格式问题

大小写转换

大小写是个比较常见的操作，比如人名、城市名等的统一都可能用到大小写的转换，在 Python 里直接使用 upper(), lower(), title() 函数，方法如下：

```
#全部大写
df2.columns = df2.columns.str.upper()
#全部小写
df2.columns = df2.columns.str.lower()
#首字母大写
df2.columns = df2.columns.str.title()
```

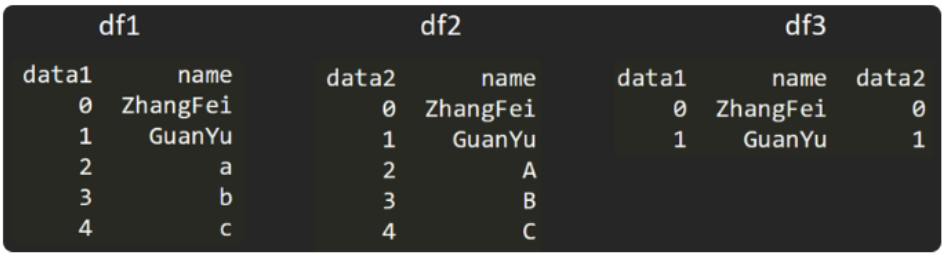

更改数据格式

这是个比较常用的操作，因为很多时候数据格式不规范，我们可以使用 astype 函数来规范数据格式，比如我们把 Chinese 字段的值改成 str 类型，或者 int64 可以这么写：

```
df2['Chinese'].astype('str')
df2['Chinese'].astype(np.int64)
```


连接两个数据结构

```
df3 = pd.merge(df1, df2, on='name')
```



df1

data1	name
0	ZhangFei
1	GuanYu
2	a
3	b
4	c

df2

data2	name
0	ZhangFei
1	GuanYu
2	A
3	B
4	C

df3

data1	name	data2
0	ZhangFei	0
1	GuanYu	1

2. inner 内连接

inner 内链接是 merge 合并的默认情况，inner 内连接其实也就是键的交集，在这里 df1, df2 相同的键是 name，所以是基于 name 字段做的连接：

```
df3 = pd.merge(df1, df2, how='inner')
```


apply 函数是 Pandas 中**自由度非常高的函数**，使用频率也非常高。

比如我们想对 name 列的数值都进行大写转化可以用：

```
df['name'] = df['name'].apply(str.upper)
```

我们也可以定义个函数，在 apply 中进行使用。比如定义 double_df 函数是将原来的数值 *2 进行返回。然后对 df1 中的“语文”列的数值进行 *2 处理，可以写成：

```
def double_df(x):
            return 2*x
df1[u'语文'] = df1[u'语文'].apply(double_df)
```

如何用 SQL 方式打开 PandasPandas 的 DataFrame 数据类型可以让我们像处理数据表一样进行操作，比如数据表的增删改查，都可以用 Pandas 工具来完成。不过也会有很多人记不住这些 Pandas 的命令，相比之下还是用 SQL 语句更熟练，用 SQL 对数据表进行操作是最方便的，它的语句描述形式更接近我们的自然语言。事实上，在 Python 里可以直接使用 SQL 语句来操作 Pandas。这里给你介绍个工具：pandasql。pandasql 中的主要函数是 sqldf，它接收两个参数：一个 SQL 查询语句，还有一组环境变量 globals() 或 locals()。这样我们就可以在 Python 里，直接用 SQL 语句中对 DataFrame 进行操作，举个例子：

```
import pandas as pd
from pandas import DataFrame
from pandasql import sqldf, load_meat, load_births
df1 = DataFrame({'name':['ZhangFei', 'GuanYu', 'a', 'b', 'c'], 'data1':range(5)})
pysqldf = lambda sql: sqldf(sql, globals())
sql = "select * from df1 where name ='ZhangFei'"
print pysqldf(sql)
```

在敲代码的过程中，老是会遇到在Dataframe中新添加一列的情况，每次都要重新google，这次做个记录。

- 其实在Dataframe中新添加一列很简单，直接指明列名，然后赋值就可以了。

```
1 import pandas as pd
2
3 data = pd.DataFrame(columns=['a','b'], data=[[1,2],[3,4]])
4 data
```

```
1 >>> data
2    a  b
3 0  1  2
4 1  3  4
```

下面我们添加一列’ c ‘，赋值为空白值。打印出来，我们可以看到已经成功添加了一列’ c'。

```
1 data['c'] = ''
2 data
```


2. 重命名列名 columns，让列表名更容易识别

如果你想对 DataFrame 中的 columns 进行重命名，可以直接使用 rename(columns=new_names, inplace=True) 函数，比如我把列名 Chinese 改成 YuWen，English 改成 YingYu。


```
1 df2.rename(columns={'Chinese': 'YuWen', 'English': 'Yingyu'}, inplace = True)
```


答案：

```python
# -*- coding=utf-8 -*-
import pandas as pd
import  numpy as np
from pandas import Series, DataFrame
df1=DataFrame({'Chinese': [66, 95, 95, 95,80,80],'English': [None,98, 96, 90, 90,90],'Math': [65, 85, 92, 90, 90
,90]},index=['Zhangfei','Guanyu','Zhaoyun','Huangzhong','Dianwei','Dianwei'])
# print(df1)
#去除重复的行
df2 = df1.drop_duplicates() #去除重复行
# print(df2)
# print(df2.isnull().any())
#求每一行的平均值
# df2['总成绩'] = df2.apply(lambda x: x.sum(),axis=1)
#重命名列名
df2.rename(columns={'Chinese':'语文','English':'英语','Math':'数学'},inplace=True)
print(df2)

# 打印显示成绩单信息，张飞有空值
df2['英语'].fillna(df2['英语'].mean(), inplace=True)
#重新求一次总成绩
df2['总成绩'] = df2.apply(lambda x: x.sum(),axis=1)
#将总成绩进行排序
df2.sort_values(['总成绩'], ascending=[False], inplace=True)
print(df2.isnull().sum())
print(df2.describe())
print(df2)
```

结果

	语文	英语	数学	总成绩
Zhaoyun	95	96.0	92	283.0
Guanyu	95	98.0	85	278.0
Huangzhong	95	90.0	90	275.0
Dianwei	80	90.0	90	260.0
Zhangfei	66	93.5	65	224.5