Pandas扩展知识

1. 加载数据

```
# https://gist.github.com/tijptjik/9408623 wine.csv
import pandas as pd
data = pd.read_csv("wine.csv")
```

```
---------------------------------------------------------------------------

FileNotFoundError                         Traceback (most recent call last)

<ipython-input-2-5ef77fd86430> in <module>()
      1 # https://gist.github.com/tijptjik/9408623 wine.csv
      2 import pandas as pd
----> 3 data = pd.read_csv("wine.csv")
```

```
D:\programs\Anaconda3\envs\base_cp\lib\site-packages\pandas\io\parsers.py in
parser_f(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols,
squeeze, prefix, mangle_dupe_cols, dtype, engine, converters, true_values,
false_values, skipinitialspace, skiprows, nrows, na_values, keep_default_na, na_filter,
verbose, skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col,
date_parser, dayfirst, iterator, chunksize, compression, thousands, decimal,
lineterminator, quotechar, quoting, escapechar, comment, encoding, dialect,
tupleize_cols, error_bad_lines, warn_bad_lines, skipfooter, skip_footer, doublequote,
delim_whitespace, as_recarray, compact_ints, use_unsigned, low_memory, buffer_lines,
memory_map, float_precision)
    707                     skip_blank_lines=skip_blank_lines)
    708
--> 709         return _read(filepath_or_buffer, kwds)
    710
    711     parser_f.__name__ = name
```

```
D:\programs\Anaconda3\envs\base_cp\lib\site-packages\pandas\io\parsers.py in
_read(filepath_or_buffer, kwds)
    447
    448     # Create the parser.
--> 449     parser = TextFileReader(filepath_or_buffer, **kwds)
    450
    451     if chunksize or iterator:
```

```
D:\programs\Anaconda3\envs\base_cp\lib\site-packages\pandas\io\parsers.py in
__init__(self, f, engine, **kwds)
    816             self.options['has_index_names'] = kwds['has_index_names']
    817
--> 818         self._make_engine(self.engine)
    819
    820     def close(self):
```

```
D:\programs\Anaconda3\envs\base_cp\lib\site-packages\pandas\io\parsers.py in
_make_engine(self, engine)
   1047     def _make_engine(self, engine='c'):
   1048         if engine == 'c':
-> 1049             self._engine = CParserWrapper(self.f, **self.options)
   1050         else:
   1051             if engine == 'python':
```

```
D:\programs\Anaconda3\envs\base_cp\lib\site-packages\pandas\io\parsers.py in
__init__(self, src, **kwds)
   1693         kwds['allow_leading_cols'] = self.index_col is not False
   1694
-> 1695         self._reader = parsers.TextReader(src, **kwds)
   1696
   1697         # XXX
```

```
pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()
```

```
pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._setup_parser_source()
```

```
FileNotFoundError: File b'wine.csv' does not exist
```

```
#data
data.head()
```

```
print(data.head())
print(data.tail(3))
print(data.describe())
print("***")
print(data.columns)
```

```
data.head()
```

2. Pandas中的统计和汇总方法

我们可以加载自己的数据集到panda的DataFrame. 以此为例.

```
import pandas as pd

df = pd.read_csv("mydataset.csv")

print(df.head())
print(df.tail(3))
# 统计方法: describe() 针对Series和DataFrame的各列计算汇总统计
print(df.describe())
print("各列的最小值/最大值:")
print(df.min())
print(df.max())
# 求值的总和: 各列的总和
print(df.sum())
print("各列的平均值:")
print(df.mean())
print("各列的方差:")
print(df.var())
print("各列的标准差:")
print(df.std())
print("计算样本的累计和:")
print(df.iloc[:,1:].cumsum())
print("计算一阶差分:")
print(df.iloc[:,1:].diff())
# 上面遇到NaN,可以用fillna()填充缺失数据, 也可以用dropna()丢弃含有缺失值的行
print("计算一阶差分:")
print(df.iloc[:,1:].diff().fillna(0))
print("计算一阶差分:")
print(df.iloc[:,1:].diff().fillna(method='bfill'))
print("计算一阶差分:")
print(df.iloc[:,1:].diff().dropna())
print("***")
print(df.columns)
```

3. 列索引

```
pwd
```

```
import pandas as pd
import os

# 获取当前工作的文件夹名.(pwd)
# file = os.path.dirname(__file__)
#print(file)

file = 'C:\\Users\\huang\\algorithms_2nd_edition\\lec_research\\kNN_Classification'
# use read_csv to read data in the dataset as a data frame
#df = pd.read_csv(file+"/DATA/Module2/Datasets/direct_marketing.csv")
df =
pd.read_csv(r'C:\\Users\\huang\\algorithms_2nd_edition\\lec_research\\kNN_Classificatio
n\\DATA\\Module2\\Datasets\\direct_marketing.csv')
```

```
print(df.tail(2))
print("******")

#print(df.recency)
print(df.recency.head(2))

print(df['recency'].head(2))

print("******")
print(df.loc[:,'recency'].head(2))

# 数值索引: pandas中的“iloc”通过数来选择行和列
print(df.iloc[:, 0].head(2))
```

4. 行索引

```
# 当前工作的文件夹
file = 'C:\\Users\\huang\\algorithms_2nd_edition\\lec_research\\kNN_Classification'

# use read_csv to read data in the dataset as a data frame
df = pd.read_csv(file+"\\DATA\\Module2\\Datasets\\direct_marketing.csv")
# 行索引
print(df[0:3])
print("******索引: 头两行,所有列***")
print(df.iloc[0:2, :])
print("******索引: 头两行,头5列***")
print(df.iloc[0:2, :5])

print("******索引: 头两行,自选3列***")
print(df.iloc[0:2, [0,1,3]])
```

5. 布尔型索引

```
# 布尔型索引 (boolean index)
print(df.recency < 7)

#feed back boolean series to regular df
print("feed back boolean series to regular dataframe")
print(df[df.recency<7])

# 将多个布尔型索引组合
print("combine multiple boolean indexing conditions")
print(df[(df.recency<7)&(df.newbie==0)])
```

```
# df can take in a list of parameters
print(df[['recency']].head(3)) # we will get back a dataframe
print(df.loc[:,['recency']].head(3)) # we will get back a dataframe

print(df.iloc[:, [0]].head(3)) # we will get back a dataframe.
```

```
# 赋值给切片
df[df.recency<7] = -100
print(df.head())
```

6. 编码方法

```
# 简单编码方法
df = pd.DataFrame({
        "哺乳动物":[
            "fish",
            "human",
            "bird",
            "dog"
]})

print("编码前:\n",df)

df.哺乳动物 = df.哺乳动物.astype("category").cat.codes
print("编码后: \n",df)
```

```
df = pd.DataFrame({
        "vertebrates":[
            "fish",
            "human",
            "bird",
            "dog"
]})

print(df)

# 注意: 不要忘了中括号 [ ]
df = pd.get_dummies(df, columns=["vertebrates"])

print(df)
```

```
# for nominal features: option 1 -- fast and dirty coding method
# 对名义上的特征: 可选方法1: 不那么好但快速的编码方法
df = pd.DataFrame({'vertebrates':[
'Bird',
'Bird',
'Mammal',
'Fish',
```

```
'Amphibian',
'Reptile',
'Mammal']})

df['vertebrates'] = df.vertebrates.astype("category").cat.codes
print("不那么好但很快的编码方法:\n",df)

# 可选方法2: 更精确的编码方法
# more accurate coding method

df = pd.get_dummies(df,columns=['vertebrates'])

print('With more accurate method:')
print(df)
```

扩展阅读

Pandas: http://pandas.pydata.org/pandas-docs/stable/cookbook.html

数据处理技术: https://chrisalbon.com/#Python

处理缺失数据: http://pandas.pydata.org/pandas-docs/stable/missing_data.html

Scikit模块家族: https://scikits.appspot.com/scikits

提取特征的技术:--关于词袋模型更多介绍 http://scikit-learn.org/stable/modules/feature_extraction.html#the-bag-of-words-representation

可视化 http://pandas.pydata.org/pandas-docs/stable/visualization.html

```
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3'],
                          'C': ['C0', 'C1', 'C2', 'C3'],
                          'D': ['D0', 'D1', 'D2', 'D3']},
                          index=[0, 1, 2, 3])
```

```
df2 = pd.DataFrame({'A': ['A4', 'A5', 'A6', 'A7'],
                      'B': ['B4', 'B5', 'B6', 'B7'],
                       'C': ['C4', 'C5', 'C6', 'C7'],
                        'D': ['D4', 'D5', 'D6', 'D7']},
                         index=[4, 5, 6, 7])
```

```
df3 = pd.DataFrame({'A': ['A8', 'A9', 'A10', 'A11'],
                          'B': ['B8', 'B9', 'B10', 'B11'],
                          'C': ['C8', 'C9', 'C10', 'C11'],
                       'D': ['D8', 'D9', 'D10', 'D11']},
                         index=[8, 9, 10, 11])
```

```
frames= [df1, df2,df3]
```

```
result = pd.concat(frames)
```

```
result
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

```
df4 = pd.DataFrame({'B': ['B2', 'B3', 'B6', 'B7'],
                    'D': ['D2', 'D3', 'D6', 'D7'],
                    'F': ['F2', 'F3', 'F6', 'F7']},
                    index=[2, 3, 6, 7])
```

```
result = pd.concat([df1, df4], axis=1)
```

```
result
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3
6	NaN	NaN	NaN	NaN	B6	D6	F6
7	NaN	NaN	NaN	NaN	B7	D7	F7

```
result = pd.concat([df1, df4], axis=1, join='inner')
```

```
result
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3

```
result = pd.concat([df1, df4], axis=1, join_axes=[df1.index])
```

```
result
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3

```
result = df1.append(df2)
```

```
result
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

```
s1 = pd.Series(['X0', 'X1', 'X2', 'X3'], name='X')
pd.concat([df1, s1], axis=1)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	A	B	C	D	X
0	A0	B0	C0	D0	X0
1	A1	B1	C1	D1	X1
2	A2	B2	C2	D2	X2
3	A3	B3	C3	D3	X3

```
pd.concat([df1, s1], axis=1, ignore_index=True)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	0	1	2	3	4
0	A0	B0	C0	D0	X0
1	A1	B1	C1	D1	X1
2	A2	B2	C2	D2	X2
3	A3	B3	C3	D3	X3

```
#merge
left = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                     'A': ['A0', 'A1', 'A2', 'A3'],
                     'B': ['B0', 'B1', 'B2', 'B3']})

right = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                      'C': ['C0', 'C1', 'C2', 'C3'],
                     'D': ['D0', 'D1', 'D2', 'D3']})

pd.merge(left, right, on='key')
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	A	B	key	C	D
0	A0	B0	K0	C0	D0
1	A1	B1	K1	C1	D1
2	A2	B2	K2	C2	D2
3	A3	B3	K3	C3	D3

```
left = pd.DataFrame({'key1': ['K0', 'K0', 'K1', 'K2'],
                     'key2': ['K0', 'K1', 'K0', 'K1'],
                'A': ['A0', 'A1', 'A2', 'A3'],
                     'B': ['B0', 'B1', 'B2', 'B3']})
```

```
right = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                      'key2': ['K0', 'K0', 'K0', 'K0'],
                       'C': ['C0', 'C1', 'C2', 'C3'],
                        'D': ['D0', 'D1', 'D2', 'D3']})
```

```
pd.merge(left, right, on=['key1', 'key2'])
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	A	B	key1	key2	C	D
0	A0	B0	K0	K0	C0	D0
1	A2	B2	K1	K0	C1	D1
2	A2	B2	K1	K0	C2	D2

```
pd.merge(left, right, how='left', on=['key1', 'key2'])
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	A	B	key1	key2	C	D
0	A0	B0	K0	K0	C0	D0
1	A1	B1	K0	K1	NaN	NaN
2	A2	B2	K1	K0	C1	D1
3	A2	B2	K1	K0	C2	D2
4	A3	B3	K2	K1	NaN	NaN

```
pd.merge(left, right, how='right', on=['key1', 'key2'])
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	A	B	key1	key2	C	D
0	A0	B0	K0	K0	C0	D0
1	A2	B2	K1	K0	C1	D1
2	A2	B2	K1	K0	C2	D2
3	NaN	NaN	K2	K0	C3	D3