第四周

sklearn. 数据集 (4a)

```
from sklearn import datasets
ivis = datasets.load_iris()
x, y = ivis.data, ivis.target
ivis.feature_names
```

```
df = pd.read_csv('....csv')
df['xxx'] → 取xxx列
df[1:3] → 切片取第二行
df.columns=['star';'...';...]
        → 替换表头.
df.loc[1:3, ['star']] → 显示指定行,列.
```

df[df['star']=='力荐']

→ 筛选

df.dropna() → 删除缺失数据

df.groupby('start').sum()

创建新列：

```
star_to_number={
    '力荐':5,
    ...}
```

df['new_star']=df['star'].map(star_to_number)

df[df['new_star']>=3] → 筛选中评以上

Pandas

Series: (列) (index 和 value)

a	0
b	1
c	2

DataFrame: (多行多列)

	index	0
0	a	0
1	b	1
2	c	2

---

pd.series(['a', 'b', 'c'])

S = pd.Series({'a':1, 'b':2})

S = pd.Series([...], index = [...])

S.index S.values.tolist()

```python
# 取出email
emails = pd.Series(['abc at amazom.com', 'admin1@163.com', 'mat@m.at', 'ab@abc.com'])
import re
pattern ='[A-Za-z0-9._]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,5}'
mask = emails.map(lambda x: bool(re.match(pattern, x)))
emails[mask]
```

---

df = pd.DataFrame([['a','b'], ['c','d']])
df.columns = ['...','...']
df.index = ['...','...']
↳ 前2行引用

数据导入.

pd.read_excel(r'1.xlsx')
.read_excel(r'1.xlsx', sheet_name = 0)
pd.read_csv(r'...csv', sep=' ', nrows=10, coding='utf-8')
.read_table(r'...txt', sep=' ')

```
sql = 'SELECT * FROM mytable'
conn = pymysql.connect('ip', 'name', 'pass', 'dbname',
pd.read_sql(sql, conn)          'charset')
```

```
excel.head(3)
    .shape
    .info()
    .describe()
```

数据预处理 (46)

Series:

x.hasnans → 判断有空值.

x.fillna(value = x.mean())

DataFrame:

df.isnull().sum()

.ffill() → 用上一行填充

.ffill(axis=1) → 用上一列填充.

df.info()

df.dropna() → 删掉有na的整行

- fillna('无')

.drop_duplicates() → 去重行

数据调整

df[['A','C']] → 筛选列

df.iloc[:, [0, 2]] → :所有行

[0, 2] → 1, 3列

df.loc[[0, 2]] → 1, 3行

- loc[0:2] → 1→3行

df[(df['A']<5) & (df['C']<4)]

→ A列<5 且 C列<4 的

替换

```
df['c'].replace(4, 40)

df.replace(np.NaN, 0)

df.replace([4,5,8], 1000)
  .replace({4:400, 5:500})
```

排序

```
df.sort_values(by=['A'],
               ascending=False)
  .sort_values(by=['A','C'],
               ascending=[True, False])
```

删除

```
df.drop('A', axis=1) → 删除A列
  .drop(3, axis=0) → 删除3行
df = df[df['A'] < 4] → A列小于4的行
```

行列转置：

df.T

透视表：

df.stack() → 展平

df.unstack() → 反向展平.

df.stack().reset_index()

→ 填充空值

计算：

df['A'] + df['C'] (空值无法计算)

df['A'] + 5

df['A'] < df['C'] (空值永远False)

df.sum()

数据分组聚合.

```
df.groupby('type').groups
for a,b in df.groupby('type'):
    print(a)
    print(b)
```

```
df.groupby('type').count
```

☆ `df.groupby('type').aggregate({'type':'count', 'Teb':'sum'})`

聚合 `.agg()`

```
df.groupby('group').agg('mean')
```

合并去重 ← `.mean().to_dict()`

不合并 ← `.transform('mean')`

透视表：

```
pd.pivot_table(data,
               values='salary',
               columns='groups',
               index='age',
               aggfunc='count',
               margins=True
               ).reset_index()
```

```
37  # 数据透视表
38  pd.pivot_table(data,
39                 values='salary',
40                 columns='group',
41                 index='age',
42                 aggfunc='count',
43                 margins=True
44                ).reset_index()
45
```

输出　终端　调试控制台　问题

```
...                 aggfunc='count',
...                 margins=True
...               ).reset_index()
group  age    x    y    z  All
0       24  NaN  NaN  1.0    1
1       32  NaN  NaN  1.0    1
2       35  2.0  NaN  NaN    2
3       36  1.0  NaN  NaN    1
4       39  1.0  NaN  NaN    1
5       42  1.0  NaN  NaN    1
6       44  NaN  NaN  1.0    1
7       47  NaN  1.0  NaN    1
8       49  1.0  NaN  NaN    1
9      All  6.0  1.0  3.0   10
>>>
```

多表拼接：

pd.merge(data1, data2)

.merge(data1, data2, on='group')

(, , left_on='..',

right_on='..')

how='inner'

- left
- right
- outer

pd.concat([data1, data2])

↓

纵向拼接。

用于有同样列的情况。

存为：

```
df.to_excel(excel_writer
            = r'file.xlsx')
      sheet_name='..',
      index=False,
    columns=['..','..']
    encoding='utf-8',
    na_rep = 0
    inf_rep=0
df.to_csv()
df.to_pickle('xx.pkl')
```

绘图：

```
dates = pd.date_range('20200101', periods=12)
df = pd.DataFrame(np.random.randn(12, 4),
                  index=dates, columns=list('ABCD'))

matplotlib.pyplot as plt

plt.plot(df.index, df['A'],
         color='#BBFFAA',
         linestyle='--',
         linewidth=3, marker='D')

plt.show()
```

横坐标.　　纵坐标.

```
import seaborn as sns.
plt.scatter(df.index, df['A'])
sns.set_style('darkgrid')
plt.show()
```

---

jieba 分词与提取关键词.

(4C)

```
r = jieba.cut(string, cut_all=False)
'/'.join(list(result))
```

cut_all=False → 精确模式

True: 全模式

```
.cut_for_search(...)
```

import jieba.analyse.

jieba.analyse.extract_tags(text,
topk=5, → 权重最大的5个词
withWeight=True)
→ 返回每个词的权重值.

↓
tf-IDF 算法.

stop_words='./.../.../.txt'
jieba.analyse.set_stop_words(stop_words)
.textrank( )
↓
基于 TextRank

user_dict :

... 3 nt

jieba.load_userdict (user_dict)

jieba.add_word('航运大学')

.del_word(...)

jieba.cut(HMM=False)

关闭新词发现识别

jieba.suggest_freq('中出', True)

.suggest_freq('中', '将'), True)

SnowNLP 情感倾向分析

```
from snownlp import SnowNLP

s = SnowNLP(text)
s.words  → 分词. 中文
list(s.tags) → 词性标注
s.sentiments → 情感分析.
```

训练:

```
from snownlp import seg
seg.train('data.txt')
seg.save('seg.marshal')
```

s.pinyin → 转拼音.
s.han → 繁→简体
s.keywords(limit=5)