网销案件数据预处理

目的：对获取到的网销案件数据进行预处理，包括理解特征，对数据类型、缺失值处理，相应字段的筛选、加工，无关字段的删除等，为下一步的探索性数据分析做准备。

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from  pyecharts.charts import Bar
import warnings
warnings.filterwarnings('ignore')
```

1. 导入数据

```
data = pd.read_csv('data.csv')
```

```
data.info()
```

```
RangeIndex: 1428 entries, 0 to 1427
Data columns (total 29 columns):
中支公司          1428 non-null object
机构            1428 non-null object
保单号           1428 non-null object
投保人           1428 non-null object
身份证           1428 non-null object
出生日期          1428 non-null object
与被保险人关系       1428 non-null object
险种代码          1428 non-null int64
报案号           1428 non-null int64
赔案号           1428 non-null int64
出险日期          1428 non-null object
保险起期          1428 non-null object
报案时间          1428 non-null object
核赔(已决)时间     1428 non-null object
赔付类型          1428 non-null object
损失类型          1428 non-null object
已决金额          1428 non-null object
重开次数          1428 non-null int64
业务来源          1428 non-null object
终端来源          1428 non-null object
渠道细分          1428 non-null object
险种            1428 non-null object
出险原因          1428 non-null object
出险省份          1426 non-null object
出险出险市         1426 non-null object
出险镇           1382 non-null object
出险路           1428 non-null object
被保险人          1428 non-null object
```

```
事故经过          1428 non-null object
dtypes: int64(4), object(25)
```

2. 数据预处理

2.1 各字段处理

```
# “事故经过”字段的提取与删除；“出险原因”字段的处理
df =  data[data['出险原因']=='其他']['事故经过']
index1 = df[df.str.contains('病')].index
data['出险原因'].loc[index1] = '普通疾病'

index2 = df[df.str.contains('炎')].index
data['出险原因'].loc[index2] = '普通疾病'

index3 = data[data['出险原因']=='其他']['事故经过'].index
data['出险原因'].loc[index3] = '意外'

data.drop(columns='事故经过', inplace=True)
```

```
# 字段名称修改
col =
['company','organization','insurance_policy_number','policy_holder','id','birthd
ay','relationship_insured','insurance_code','case_number','claim_number','date_c
ase','start_date_insurance','report_date','decision_time','pay_type','loss_type'
,'pay_money','pay_counts','business_source','terminal_source','channel','insuran
ce_type','case_reason','case_province','case_city','case_town','case_road','insu
red_name']
data.columns = col
```

```
# 字典映射替换中支公司名称
dict = {
    '010201广东分公司营业一部（虚拟）'              : '营业一部',
    '010222广东分公司阳江中心支公司'                : '阳江',
    '010211广东分公司广州市分公司'                  : '广州市分',
    '010218广东分公司茂名中心支公司'                : '茂名',
    '010215广东分公司佛山中心支公司'                : '佛山',
   '010206广东分公司佛山市顺德支公司（虚拟）'      : '顺德',
    '010223广东分公司清远中心支公司'                : '清远',
    '010217广东分公司湛江中心支公司'                : '湛江',
    '010216广东分公司江门中心支公司'                : '江门',
    '010212广东分公司韶关中心支公司'                : '韶关',
    '010214广东分公司汕头中心支公司'                : '汕头',
    '010220广东分公司惠州中心支公司'                : '惠州',
    '010227广东分公司广州市黄埔中心支公司（虚拟）'  : '黄埔',
    '010224广东分公司东莞中心支公司'                : '东莞',
    '010252广东分公司潮州中心支公司'                : '潮州',
    '010221广东分公司梅州中心支公司'                : '梅州',
    '010204广东分公司广州市番禺支公司（虚拟）'      : '番禺',
    '010205广东分公司广州市花都中心支公司（虚拟）'  : '花都',
    '010226广东分公司河源中心支公司'                : '河源'
}
data.company = data.company.apply(lambda x: dict[x])
```

```
# 筛选policy_holder中包含 ... 的记录 => None
data.policy_holder[data.policy_holder.str.contains('\.')]
```

```
# birthday 转化为日期格式
data.birthday = pd.to_datetime(data.birthday, format='%Y-%m-%d')
```

```
# 映射relationship_insured
data.relationship_insured.value_counts()
dict = {
    '01-本人': '本人',
    '50-父母': '父母',
    '99-其他': '其他',
    '10-配偶': '配偶',
    '40-子女': '子女'
}
data.relationship_insured = data.relationship_insured.apply(lambda x : dict[x])
data.relationship_insured.value_counts()
```

```
本人    686
父母    434
其他    294
配偶     11
子女      3
```

```
# insurance_code, case_number, claim_number 由int64 改为object
data.insurance_code = data.insurance_code.astype('object')
data.case_number = data.case_number.astype('object')
data.claim_number = data.claim_number.astype('object')
```

```
# date_case改为去掉时分的时间格式，并去掉AM, PM
# 同理start_date_insurance，report_date，decision_time
data.date_case = data.date_case.str.replace('AM', '')
data.date_case = data.date_case.str.replace('PM', '')
data.date_case = pd.to_datetime(data.date_case).dt.normalize()
```

```
import time
def to_time(x):
    x = x.replace('AM', '')
    x = x.replace('PM', '')
    x = pd.to_datetime(x)
    return x.strftime('%Y-%m-%d')
```

```
data.start_date_insurance = data.start_date_insurance.apply(to_time)
data.report_date = data.report_date.apply(to_time)
data.decision_time = data.decision_time.apply(to_time)
```

```
# 转化为日期格式
data.start_date_insurance = data.start_date_insurance.astype('datetime64[ns]')
data.report_date = data.report_date.astype('datetime64[ns]')
data.decision_time = data.decision_time.astype('datetime64[ns]')
```

```
# pay_type映射为0,1
dict = {
    '正常': 0,
    '重开1': 1
}
data.pay_type = data.pay_type.apply(lambda x : dict[x])
data.pay_type.value_counts()
```

```
0    1415
1      13
```

```
# loss_type映射为 0,1 ,
dict = {
    '人伤': 0,
    '费用': 1
}
data.loss_type = data.loss_type.apply(lambda x : dict[x])
data.loss_type.value_counts()
```

```
0    1400
1      28
```

```
# pay_money改为 float64， 去掉逗号 ，
data.pay_money = data.pay_money.str.replace(',', '')
data.pay_money = data.pay_money.astype('float64')
```

```
# pay_counts改为‘object’
data.pay_counts = data.pay_counts.astype('object')
```

```
# business_source 用字典映射
dict = {
    '1900201专业代理'         : '专业代理',
    '190020202邮政代理'       : '邮政代理',
    '1900103直接业务（网销）': '直接业务（网销）',
    '19003经纪业务'           : '经纪业务',
    '1900203个人代理'         : '个人代理'
}
data.business_source = data.business_source.apply(lambda x : dict[x])
data.business_source.value_counts()
```

```
专业代理          823
邮政代理          264
直接业务（网销）     191
经纪业务          146
个人代理            4
```

```
# terminal_source 用字典映射
dict = {
    '0201PC'                : 'PC',
    '0106移动展业(App)'     : '移动展业(App)',
    '0202APP'               : 'APP',
    '0105微信（WeChat）'    : 'WeChat'
}
data.terminal_source = data.terminal_source.apply(lambda x : dict[x])
data.terminal_source.value_counts()
```

```
PC            1184
移动展业(App)     119
APP             70
WeChat          55
```

```
# insurance_type 用字典映射
dict1 = {
    '个人人身意外伤害保险'          : '个意险',
    '学生幼儿平安人身意外伤害保险'  : '学平险',
    '出行无忧人身意外伤害保险'      : '出行无忧',
    '驾校学员人身意外伤害保险'      : '驾意险',
    '境外出行综合保障保险'          : '境外出行',
    '华安个人医疗保险（A款）'       : '个人医疗A',
    '急性肠胃炎健康保险'            : '肠胃炎',
    '旅游人身意外伤害保险'          : '旅游意外'
}
data.insurance_type = data.insurance_type.apply(lambda x : dict1[x])
data.insurance_type.value_counts()
```

```
个意险       616
学平险       485
出行无忧      264
驾意险        27
境外出行       16
个人医疗A      10
肠胃炎         8
旅游意外        2
```

```
# 去掉名字中的省略号
data.insured_name[data.insured_name.str.contains('\.')] = \
        data.insured_name[data.insured_name.str.contains('\.')].apply(lambda x:
x[0:3])
```

```
data.dtypes
```

```
company                           object
organization                      object
insurance_policy_number           object
policy_holder                     object
id                                object
birthday                  datetime64[ns]
relationship_insured              object
insurance_code                    object
```

```
case_number                          object
claim_number                         object
date_case                    datetime64[ns]
start_date_insurance         datetime64[ns]
report_date                  datetime64[ns]
decision_time                datetime64[ns]
pay_type                             int64
loss_type                            int64
pay_money                          float64
pay_counts                           object
business_source                      object
terminal_source                      object
channel                              object
insurance_type                       object
case_reason                          object
case_province                        object
case_city                            object
case_town                            object
case_road                            object
insured_name                         object
dtype: object
```

2.2 缺失值处理

```
data.isnull().sum()
# 删掉case_reason和case_city的2个缺失值数据
# case_town用处不大，可以不用处理
```

```
company                      0
organization                 0
insurance_policy_number      0
policy_holder                0
id                           0
birthday                     0
relationship_insured         0
insurance_code               0
case_number                  0
claim_number                 0
date_case                    0
start_date_insurance         0
report_date                  0
decision_time                0
pay_type                     0
loss_type                    0
pay_money                    0
pay_counts                   0
business_source              0
terminal_source              0
channel                      0
insurance_type               0
case_reason                  0
case_province                2
case_city                    2
case_town                   46
case_road                    0
insured_name                 0
```

```
dtype: int64
```

```
data[data.case_province.isnull()]
data = data.dropna(axis=0,subset=["case_province"])
```

3. 特征提取

```
# 提取年龄特征，注意birthday为投保人生日，当投保人与被保险人不是同一个人时，设为NaN
age = data.date_case.dt.year - data.birthday.dt.year
data['age'] = age
```

```
# 去除policy_holder和insured_name空格
data.policy_holder = data.policy_holder.str.strip()
data.insured_name = data.insured_name.str.strip()
```

```
# 当投保人与被保险人不是同一个人时，age设为NaN
pd.set_option('display.max_rows', None)
data['age'][data.policy_holder!=data.insured_name] = np.NaN
```

挖掘特征：

```
- 出险日期与保险起期，相差天数  'diff_case_date'
- 出险日期与报案日期，相差天数  'diff_report_date'
- 报案日期与赔款日期，相差天数，'diff_dic_data'
```

```
data['diff_case_date'] = (data.date_case - data.start_date_insurance)/
np.timedelta64(1, 'D')
data['diff_report_date'] = (data.report_date - data.date_case)/
np.timedelta64(1, 'D')
data['diff_dic_data'] = (data.decision_time - data.report_date)/
np.timedelta64(1, 'D')
```

```
# 删掉唯一特征值，以及提取完后无价值特征值，insurance_policy_number， policy_holder，
case_number， claim_number，pay_type(与pay_counts强线性相关)
data.drop(columns=['insurance_policy_number', 'policy_holder', 'case_number',
'claim_number', 'insured_name', 'pay_type'], inplace=True)
```

```
# 身份证后6为提取为id
data.id = data.id.apply(lambda x: x[len(x)-6:])
```

```
# 将处理后的数据保存为CSV，进行后续分析
data.to_csv('process_data.csv', encoding='utf_8_sig', index=0)
```