analyser.py

```
import json
import os
import shutil
import pandas as pd
import utils.analyser_utils as analyser_utils
import requests
import cv2.cv2 as cv2
import datetime

from tkinter import ttk, Button, messagebox, StringVar, Label, Entry, Toplevel
from requests_ntlm import HttpNtlmAuth


class Analyser(object):
    def __init__(self, root, first_analysed_df, save_path, day):
        self.root = root
        self.source_df = first_analysed_df.reset_index(drop=True)
        self.data_frame = self.initialize_df(first_analysed_df)
        self.window = None
        self.temp_window = None
        self.url = 'https://dataorch.axlehire.com/shipments/search'
        self.header = {
            'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36',
            'content-type': 'application/json',
            'cookie': r'fp=1a39e1225ea764ca9f2abf599fafba34; xtoken="dE9DbW1wYkZDI/B28g5MkirtzwljFDty7THWI75r/mVq4do8YKOJBeUtONSQ1d3L1Yb5JCAEZPTk\012FFj7LXpbKjSaV71j1S6I9zjtTLurIi1ddgqe+xsIRU84cjgOSktu\012"'}
        self.index = 0
        self.save_folder_path = save_path
        self.day = day
        self.mission_length = len(self.data_frame['Tracking Code'])

    def initialize_param_dict(self):
        # 周四
        if self.day == '4':
```

```
34                 param_dict = {'Reason for Complaint': StringVar(),
'Details of Complaint': StringVar(),
35                               'Tracking Code': StringVar(), 'Drop
off status': StringVar(),
36                               'Earliest Dropoff Time': StringVar(),
'Latest Dropoff Time': StringVar(),
37                               'Scheduled Delivery Date':
StringVar(), 'Shipment status': StringVar(),
38                               'Inbound Scan Date 减 Scheduled
Delivery Date': StringVar(),
39                               'Inbound Scan Date (Linehaul)':
StringVar(), 'Inbound Scan Time': StringVar(),
40                               'Inbound status': StringVar(), 'Pickup
Date 减 Scheduled Delivery Date': StringVar(),
41                               'Pickup Date': StringVar(), 'Pickup
Time': StringVar(), 'Pickup Status': StringVar(),
42                               'Drop off date 减 Pickup Date':
StringVar(), 'Drop off date': StringVar(),
43                               'Drop off time': StringVar(), 'Drop
off remark': StringVar()}
44                 return param_dict
45             # 周三
46             elif self.day == '3':
47                 param_dict = {'Tracking Code': StringVar(), 'Drop
off status': StringVar(),
48                               'Earliest Dropoff Time': StringVar(),
'Latest Dropoff Time': StringVar(),
49                               'Scheduled Delivery Date':
StringVar(), 'Shipment status': StringVar(),
50                               'Inbound Scan Date 减 Scheduled
Delivery Date': StringVar(),
51                               'Inbound Scan Date (Linehaul)':
StringVar(), 'Inbound Scan Time': StringVar(),
52                               'Inbound status': StringVar(), 'Pickup
Date 减 Scheduled Delivery Date': StringVar(),
53                               'Pickup Date': StringVar(), 'Pickup
Time': StringVar(), 'Pickup Status': StringVar(),
54                               'Drop off date 减 Pickup Date':
StringVar(), 'Drop off date': StringVar(),
55                               'Drop off Time': StringVar(), 'Drop
off remark': StringVar()}
56                 return param_dict
57
58         @staticmethod
```

```
59        def initialize_df(first_df):
60            """
61            取一切需要打标签的内容
62            包含：drop off status == 'Success' 的情况
63            Issue Category 为空的情况 （没填上）
64            """
65            first_df['Inbound Scan Date 减 Scheduled Delivery Date']
= None
66            first_df['Pickup Date 减 Scheduled Delivery Date'] =
None
67            first_df['Pickup Date 减 Drop off date'] = None
68            for index, row in first_df.iterrows():
69                first_df.loc[index, 'Inbound Scan Date 减 Scheduled
Delivery Date'] \
70                    =
analyser_utils.date_subtract(first_df.loc[index, 'Inbound Scan Date
(Linehaul)'],
71
first_df.loc[index, 'Scheduled Delivery Date'])
72                first_df.loc[index, 'Pickup Date 减 Scheduled
Delivery Date'] \
73                    =
analyser_utils.date_subtract(first_df.loc[index, 'Pickup Date'],
74
first_df.loc[index, 'Scheduled Delivery Date'])
75                first_df.loc[index, 'Drop off date 减 Pickup Date']
\
76                    =
analyser_utils.date_subtract(first_df.loc[index, 'Drop off date'],
77
first_df.loc[index, 'Pickup Date'])
78            first_df_up = first_df[first_df['Drop off status'] ==
'SUCCEEDED']
79            first_df_down = first_df[pd.isna(first_df['Issue
Category'])]
80            res_df = pd.concat([first_df_up,
first_df_down]).drop_duplicates().reset_index(drop=True)
81            return res_df
82
83        def run(self):
84            """
85            run 的逻辑
86            再此创建一个新的 tkinter 界面，并提供两个按钮，上一页，
下一页
```

```
            上一页 依然运行 run 函数，只不过 self.index + 1
            """
            # 一堆逻辑 列出当前 index 的 dataframe，周四
            # 周四
            if self.day == '4':
                self.window = Toplevel(master=self.root)
                self.param_dict = self.initialize_param_dict()
                self.temp_window = ttk.Treeview(self.window,
show='headings')
                # 加入各种列
                self.temp_window['columns'] = ('Reason for
Complaint', 'Details of Complaint', 'Tracking Code',
                                               'Drop off status',
'Earliest dropoff time', 'Latest dropoff time',
                                               'Scheduled Date',
'Shipment status', 'Inbound 减 Scheduled', 'Inbound Date',
                                               'Inbound scan time',
'Inbound status', 'Pickup 减 Scheduled', 'Pickup Date',
                                               'Pickup Time',
'Pickup status',
                                               'Drop off 减 Pickup',
'Drop off date', 'Drop off time', 'Drop off remark')

                self.temp_window.column('Reason for Complaint',
width=65)
                self.temp_window.column('Details of Complaint',
width=65)
                self.temp_window.column('Tracking Code', width=100)
                self.temp_window.column('Drop off status',
width=100)
                self.temp_window.column('Earliest dropoff time',
width=50)  #
                self.temp_window.column('Latest dropoff time',
width=50)  #
                self.temp_window.column('Scheduled Date', width=90)
                self.temp_window.column('Shipment status', width=80)
#
                self.temp_window.column('Inbound 减 Scheduled',
width=40)
                self.temp_window.column('Inbound Date', width=80)
                self.temp_window.column('Inbound scan time',
width=50)  #
                self.temp_window.column('Inbound status', width=80)
#
```

```
115                 self.temp_window.column('Pickup 减 Scheduled',
width=40)
116                 self.temp_window.column('Pickup Date', width=90)
117                 self.temp_window.column('Pickup Time', width=65)
118                 self.temp_window.column('Pickup status', width=65)
#
119                 self.temp_window.column('Drop off 减 Pickup',
width=40)
120                 self.temp_window.column('Drop off date', width=65)
121                 self.temp_window.column('Drop off time', width=65)
122                 self.temp_window.column('Drop off remark',
width=120)
123
124                 self.temp_window.heading('Reason for Complaint',
text='Reason for Complaint')
125                 self.temp_window.heading('Details of Complaint',
text='Details of Complaint')
126                 self.temp_window.heading('Tracking Code',
text='Tracking Code')
127                 self.temp_window.heading('Drop off status',
text='Drop off status')
128                 self.temp_window.heading('Earliest dropoff time',
text='Earliest dropoff time')
129                 self.temp_window.heading('Latest dropoff time',
text='Latest dropoff time')
130                 self.temp_window.heading('Scheduled Date',
text='Scheduled Date')
131                 self.temp_window.heading('Shipment status',
text='Shipment status')
132                 self.temp_window.heading('Inbound 减 Scheduled',
text='Inbound 减 Scheduled')
133                 self.temp_window.heading('Inbound Date',
text='Inbound Date')
134                 self.temp_window.heading('Inbound scan time',
text='Inbound scan time')
135                 self.temp_window.heading('Inbound status',
text='Inbound status')
136                 self.temp_window.heading('Pickup 减 Scheduled',
text='Pickup 减 Scheduled')
137                 self.temp_window.heading('Pickup Date', text='Pickup
Date')
138                 self.temp_window.heading('Pickup Time', text='Pickup
Time')
```

```
139             self.temp_window.heading('Pickup status',
text='Pickup status')
140             self.temp_window.heading('Drop off 减 Pickup',
text='Drop off 减 Pickup')
141             self.temp_window.heading('Drop off date', text='Drop
off date')
142             self.temp_window.heading('Drop off time', text='Drop
off time')
143             self.temp_window.heading('Drop off remark',
text='Drop off remark')
144
145             # 初次设置值
146             self.change_data(data_index=0)
147             self.temp_window.pack(pady=20)
148
149             # button_next 的函数为 next_page
150             prev_button = Button(self.window, text='上一页',
command=self.prev_page)
151             prev_button.place(x=100, y=100)
152
153             next_button = Button(self.window, text='下一页',
command=self.next_page)
154             next_button.place(x=300, y=100)
155
156             confirm_button = Button(self.window, text='确定',
command=self.confirm)
157             confirm_button.place(x=900, y=100)
158
159             Button(self.window, text='清除缓存',
command=self.clear_cache).place(x=1200, y=100)
160             Button(self.window, text='显示照片',
command=self.show_pic).place(x=1100, y=100)
161             Button(self.window, text='提交',
command=self.hand_in_result).place(x=1300, y=100)
162             Button(self.window, text='打开字典',
command=self.open_dictionary).place(x=1300, y=200)
163
164             # 显示进度
165             self.process = StringVar()
166             Entry(self.window, width='10',
textvariable=self.process).place(x=100, y=300)
167             self.process.set(str(self.index) + '/' +
str(self.mission_length))
168
```

```
169                 # 绑定按键
170                 self.window.bind('<Down>', self.next_page)
171                 self.window.bind('<Up>', self.prev_page)
172                 self.window.bind('<Return>', self.confirm)
173                 self.window.bind('<s>', self.show_pic)
174 
175                 # 设置一个框，用于填对应的序号
176                 self.answer = StringVar()
177                 Label(self.window, text="此条记录的问题，对应的 JJ
序号:").place(x=500, y=100)
178                 Entry(self.window, width='5',
textvariable=self.answer).place(x=720, y=100)
179 
180                 # 显示 tracking code
181                 self.tracking_code = StringVar()
182                 Label(self.window, text="Tracking
code:").place(x=600, y=200)
183                 Entry(self.window, width='20',
textvariable=self.tracking_code).place(x=700, y=200)
184 
self.tracking_code.set(self.data_frame.loc[self.index, 'Tracking
Code'])
185 
186                 # 显示 顾客的 notes
187                 self.client_comment = StringVar()
188                 Label(self.window, text="note:").place(x=100, y=150)
189                 Entry(self.window, width='100',
textvariable=self.client_comment).place(x=150, y=150)
190                 result_dict = self.get_dict_from_tracking_code(
191                     tracking_code=self.data_frame.loc[self.index,
'Tracking Code']
192                 )
193                 if 'dropoff_note' in
result_dict['results'][0]['shipment'].keys():
194 
self.client_comment.set(result_dict['shipment']['dropoff_note'])
195                 else:
196                     self.client_comment.set('')
197 
198                 # 显示 customer id
199                 self.customer_id = StringVar()
200                 Label(self.window, text="note:").place(x=800, y=150)
201                 Entry(self.window, width='100',
textvariable=self.customer_id).place(x=900, y=150)
```

```
202                 if 'customer' in
result_dict['results'][0]['shipment'].keys():
203
self.customer_id.set(result_dict['shipment']['customer']['phone_numbe
r'])
204                 else:
205                     self.customer_id.set('')
206
207                 # 一堆逻辑 显示出图片和详细地址文字
208                 self.window.mainloop()
209
210             # 周三
211             elif self.day == '3':
212                 self.window = Toplevel(master=self.root)
213                 self.param_dict = self.initialize_param_dict()
214                 self.temp_window = ttk.Treeview(self.window,
show='headings')
215                 # 加入各种列
216                 self.temp_window['columns'] = ('Tracking Code',
217                                                'Drop off status',
218                                                'Scheduled Date',
'Earliest Dropoff Time', 'Latest Dropoff Time',
219                                                'Shipment status',
'Inbound 减 Scheduled', 'Inbound Date',
220                                                'Inbound scan time',
'Inbound status', 'Pickup 减 Scheduled', 'Pickup Date',
221                                                'Pickup Time',
'Pickup status',
222                                                'Drop off 减 Pickup',
'Drop off date', 'Drop off Time', 'Drop off remark')
223
224                 self.temp_window.column('Tracking Code', width=100)
225                 self.temp_window.column('Drop off status',
width=100)
226                 self.temp_window.column('Scheduled Date', width=120)
227                 self.temp_window.column('Earliest Dropoff Time',
width=50)  #
228                 self.temp_window.column('Latest Dropoff Time',
width=50)  #
229                 self.temp_window.column('Shipment status',
width=120)  #
230                 self.temp_window.column('Inbound 减 Scheduled',
width=40)
231                 self.temp_window.column('Inbound Date', width=120)
```

```
232                 self.temp_window.column('Inbound scan time',
width=80)  #
233                 self.temp_window.column('Inbound status', width=80)
#
234                 self.temp_window.column('Pickup 减 Scheduled',
width=40)
235                 self.temp_window.column('Pickup Date', width=50)
236                 self.temp_window.column('Pickup Time', width=50)
237                 self.temp_window.column('Pickup status', width=120)
#
238                 self.temp_window.column('Drop off 减 Pickup',
width=40)
239                 self.temp_window.column('Drop off date', width=100)
240                 self.temp_window.column('Drop off Time', width=50)
241                 self.temp_window.column('Drop off remark',
width=120)
242
243                 self.temp_window.heading('Tracking Code',
text='Tracking Code')
244                 self.temp_window.heading('Drop off status',
text='Drop off status')
245                 self.temp_window.heading('Scheduled Date',
text='Scheduled Date')
246                 self.temp_window.heading('Earliest Dropoff Time',
text='Earliest dropoff Time')
247                 self.temp_window.heading('Latest Dropoff Time',
text='Latest dropoff Time')
248                 self.temp_window.heading('Shipment status',
text='Shipment status')
249                 self.temp_window.heading('Inbound 减 Scheduled',
text='Inbound 减 Scheduled')
250                 self.temp_window.heading('Inbound Date',
text='Inbound Date')
251                 self.temp_window.heading('Inbound scan time',
text='Inbound scan time')
252                 self.temp_window.heading('Inbound status',
text='Inbound status')
253                 self.temp_window.heading('Pickup 减 Scheduled',
text='Pickup 减 Scheduled')
254                 self.temp_window.heading('Pickup Date', text='Pickup
Date')
255                 self.temp_window.heading('Pickup Time', text='Pickup
Time')
```

```
256                 self.temp_window.heading('Pickup status',
text='Pickup status')
257                 self.temp_window.heading('Drop off 减 Pickup',
text='Drop off 减 Pickup')
258                 self.temp_window.heading('Drop off date', text='Drop
off date')
259                 self.temp_window.heading('Drop off Time', text='Drop
off Time')
260                 self.temp_window.heading('Drop off remark',
text='Drop off remark')
261
262                 # 初次设置值
263                 self.change_data(data_index=0)
264                 self.temp_window.pack(pady=20)
265
266                 # button_next 的函数为 next_page
267                 prev_button = Button(self.window, text='上一页',
command=self.prev_page)
268                 prev_button.place(x=100, y=100)
269
270                 next_button = Button(self.window, text='下一页',
command=self.next_page)
271                 next_button.place(x=300, y=100)
272
273                 confirm_button = Button(self.window, text='确定',
command=self.confirm)
274                 confirm_button.place(x=900, y=100)
275                 Button(self.window, text='清除缓存',
command=self.clear_cache).place(x=1200, y=100)
276                 Button(self.window, text='显示照片',
command=self.show_pic).place(x=1100, y=100)
277                 Button(self.window, text='提交',
command=self.hand_in_result).place(x=1300, y=100)
278                 Button(self.window, text='打开字典',
command=self.open_dictionary).place(x=1300, y=200)
279
280                 # 显示进度
281                 self.process = StringVar()
282                 Entry(self.window, width='10',
textvariable=self.process).place(x=100, y=300)
283                 self.process.set(str(self.index) + '/' +
str(self.mission_length))
284
285                 # 绑定按键
```

```
286                 self.window.bind('<Down>', self.next_page)
287                 self.window.bind('<Up>', self.prev_page)
288                 self.window.bind('<Return>', self.confirm)
289                 self.window.bind('<s>', self.show_pic)
290 
291                 # 设置一个框，用于填对应的序号
292                 self.answer = StringVar()
293                 Label(self.window, text="此条记录的问题，对应的 JJ
序号:").place(x=500, y=100)
294                 entry = Entry(self.window, width='5',
textvariable=self.answer).place(x=720, y=100)
295 
296                 # 显示 tracking code
297                 self.tracking_code = StringVar()
298                 Label(self.window, text="Tracking
Code:").place(x=600, y=200)
299                 Entry(self.window, width='20',
textvariable=self.tracking_code).place(x=700, y=200)
300 
self.tracking_code.set(self.data_frame.loc[self.index, 'Tracking
Code'])
301 
302                 # 显示 顾客的 notes
303                 self.client_comment = StringVar()
304                 Label(self.window, text="note:").place(x=100, y=150)
305                 Entry(self.window, width='100',
textvariable=self.client_comment).place(x=150, y=150)
306                 result_dict = self.get_dict_from_tracking_code(
307                     tracking_code=self.data_frame.loc[self.index,
'Tracking Code']
308                 )
309                 if 'dropoff_note' in
result_dict['results'][0]['shipment'].keys():
310 
self.client_comment.set(result_dict['results'][0]['shipment']['dropof
f_note'])
311                 else:
312                     self.client_comment.set('')
313 
314                 # 显示 customer id
315                 self.customer_id = StringVar()
316                 Label(self.window, text="note:").place(x=800, y=150)
317                 Entry(self.window, width='100',
textvariable=self.customer_id).place(x=900, y=150)
```

```
318                 if 'customer' in
result_dict['results'][0]['shipment'].keys():
319
self.customer_id.set(result_dict['shipment']['customer']['phone_numbe
r'])
320                 else:
321                     self.customer_id.set('')
322
323                 # 进度条
324                 self.process.set(str(self.index) + '/' +
str(self.mission_length))
325
326                 # 一堆逻辑 显示出图片和详细地址文字
327                 self.window.mainloop()
328
329     def next_page(self, event=None):
330         self.index = self.index + 1
331
332         if self.index >= len(self.data_frame['Tracking Code']):
333             # 到达最底下了
334             messagebox.showinfo(title='警告', message='没有下一
页了')
335             self.window.focus_force()
336             self.index = self.index - 1
337             self.change_data(self.index)
338
339         self.change_data(self.index)
340
341         # tracing code
342         self.tracking_code.set(self.data_frame.loc[self.index,
'Tracking Code'])
343
344         # dropoff note
345         result_dict = self.get_dict_from_tracking_code(
346             tracking_code=self.data_frame.loc[self.index,
'Tracking Code']
347         )
348         if 'dropoff_note' in
result_dict['results'][0]['shipment'].keys():
349
self.client_comment.set(result_dict['results'][0]['shipment']['dropof
f_note'])
350         else:
351             self.client_comment.set('')
```

```
352
353             # customer_id
354             self.customer_id = StringVar()
355             if 'customer' in
result_dict['results'][0]['shipment'].keys():
356
self.customer_id.set(result_dict['shipment']['customer']['phone_numbe
r'])
357             else:
358                 self.customer_id.set('')
359
360             # 进度条
361             self.process.set(str(self.index) + '/' +
str(self.mission_length))
362
363             self.temp_window.delete(f'item{self.index - 1}')
364
365         def prev_page(self, event=None):
366             self.index = self.index - 1
367             if self.index < 0:
368                 # 到达最开始了
369                 messagebox.showinfo(title='警告', message='没有上一
页了')
370                 self.window.focus_force()
371                 self.index = self.index + 1
372                 self.change_data(self.index)
373
374             self.change_data(self.index)
375
376             # tracing code
377             self.tracking_code.set(self.data_frame.loc[self.index,
'Tracking Code'])
378
379             # dropoff note
380             result_dict = self.get_dict_from_tracking_code(
381                 tracking_code=self.data_frame.loc[self.index,
'Tracking Code']
382             )
383             if 'dropoff_note' in
result_dict['results'][0]['shipment'].keys():
384
self.client_comment.set(result_dict['results'][0]['shipment']['dropof
f_note'])
385             else:
```

```
386                 self.client_comment.set('')
387
388             # customer_id
389             self.customer_id = StringVar()
390             if 'customer' in
result_dict['results'][0]['shipment'].keys():
391
self.customer_id.set(result_dict['shipment']['customer']['phone_numbe
r'])
392             else:
393                 self.customer_id.set('')
394
395             self.temp_window.delete(f'item{self.index + 1}')
396
397         def change_data(self, data_index):
398             # 设置 StringVar
399             if self.day == '4':
400                 self.param_dict['Reason for
Complaint'].set(self.data_frame.loc[data_index, 'Reason for
Complaint'])
401                 self.param_dict['Details of
Complaint'].set(self.data_frame.loc[data_index, 'Details of
Complaint'])
402                 self.param_dict['Tracking
Code'].set(self.data_frame.loc[data_index, 'Tracking Code'])
403                 self.param_dict['Drop off
status'].set(self.data_frame.loc[data_index, 'Drop off status'])
404                 self.param_dict['Earliest Dropoff
Time'].set(self.data_frame.loc[data_index, 'Earliest Dropoff Time'])
405                 self.param_dict['Latest Dropoff
Time'].set(self.data_frame.loc[data_index, 'Latest Dropoff Time'])
406                 self.param_dict['Scheduled Delivery
Date'].set(self.data_frame.loc[data_index, 'Scheduled Delivery
Date'])
407                 self.param_dict['Shipment
status'].set(self.data_frame.loc[data_index, 'Shipment status'])
408                 self.param_dict['Inbound Scan Date 减 Scheduled
Delivery Date'].set(
409                     self.data_frame.loc[data_index, 'Inbound Scan
Date 减 Scheduled Delivery Date'])
410                 self.param_dict['Inbound Scan Date (Linehaul)'].set(
411                     self.data_frame.loc[data_index, 'Inbound Scan
Date (Linehaul)'])
```

```
412                 self.param_dict['Inbound Scan
Time'].set(self.data_frame.loc[data_index, 'Inbound Scan Time'])
413                 self.param_dict['Inbound
status'].set(self.data_frame.loc[data_index, 'Inbound status'])
414                 self.param_dict['Pickup Date 减 Scheduled Delivery
Date'].set(
415                     self.data_frame.loc[data_index, 'Pickup Date 减
Scheduled Delivery Date'])
416                 self.param_dict['Pickup
Date'].set(self.data_frame.loc[data_index, 'Pickup Date'])
417                 self.param_dict['Pickup
Time'].set(self.data_frame.loc[data_index, 'Pickup Time'])
418                 self.param_dict['Pickup
Status'].set(self.data_frame.loc[data_index, 'Pickup Status'])
419                 self.param_dict['Drop off date 减 Pickup Date'].set(
420                     self.data_frame.loc[data_index, 'Drop off date
减 Pickup Date'])
421                 self.param_dict['Drop off
date'].set(self.data_frame.loc[data_index, 'Drop off date'])
422                 self.param_dict['Drop off
time'].set(self.data_frame.loc[data_index, 'Drop off time'])
423                 self.param_dict['Drop off
remark'].set(self.data_frame.loc[data_index, 'Drop off remark'])
424
425                 self.temp_window.insert('', 0, f'item{self.index}',
values=(
426                     self.param_dict['Reason for Complaint'].get(),
427                     self.param_dict['Details of Complaint'].get(),
428                     self.param_dict['Tracking Code'].get(),
429                     self.param_dict['Drop off status'].get(),
430                     self.param_dict['Earliest Dropoff Time'].get(),
431                     self.param_dict['Latest Dropoff Time'].get(),
432                     self.param_dict['Scheduled Delivery
Date'].get(),
433                     self.param_dict['Shipment status'].get(),
434                     self.param_dict['Inbound Scan Date 减 Scheduled
Delivery Date'].get(),
435                     self.param_dict['Inbound Scan Date
(Linehaul)'].get(),
436                     self.param_dict['Inbound Scan Time'].get(),
437                     self.param_dict['Inbound status'].get(),
438                     self.param_dict['Pickup Date 减 Scheduled
Delivery Date'].get(),
439                     self.param_dict['Pickup Date'].get(),
```

```
                    self.param_dict['Pickup Time'].get(),
                    self.param_dict['Pickup Status'].get(),
                    self.param_dict['Drop off date 减 Pickup
Date'].get(),
                    self.param_dict['Drop off date'].get(),
                    self.param_dict['Drop off time'].get(),
                    self.param_dict['Drop off remark'].get()
                ))
                self.temp_window.pack(pady=20)
                print(self.param_dict['Tracking Code'].get())

            elif self.day == '3':
                self.param_dict['Tracking
Code'].set(self.data_frame.loc[data_index, 'Tracking Code'])
                self.param_dict['Drop off
status'].set(self.data_frame.loc[data_index, 'Drop off status'])
                self.param_dict['Scheduled Delivery
Date'].set(self.data_frame.loc[data_index, 'Scheduled Delivery
Date'])
                self.param_dict['Earliest Dropoff
Time'].set(self.data_frame.loc[data_index, 'Earliest Dropoff Time'])
                self.param_dict['Latest Dropoff
Time'].set(self.data_frame.loc[data_index, 'Latest Dropoff Time'])
                self.param_dict['Shipment
status'].set(self.data_frame.loc[data_index, 'Shipment status'])
                self.param_dict['Inbound Scan Date 减 Scheduled
Delivery Date'].set(
                    self.data_frame.loc[data_index, 'Inbound Scan
Date 减 Scheduled Delivery Date'])
                self.param_dict['Inbound Scan Date (Linehaul)'].set(
                    self.data_frame.loc[data_index, 'Inbound Scan
Date (Linehaul)'])
                self.param_dict['Inbound Scan
Time'].set(self.data_frame.loc[data_index, 'Inbound Scan Time'])
                self.param_dict['Inbound
status'].set(self.data_frame.loc[data_index, 'Inbound status'])
                self.param_dict['Pickup Date 减 Scheduled Delivery
Date'].set(
                    self.data_frame.loc[data_index, 'Pickup Date 减
Scheduled Delivery Date'])
                self.param_dict['Pickup
Date'].set(self.data_frame.loc[data_index, 'Pickup Date'])
                self.param_dict['Pickup
Time'].set(self.data_frame.loc[data_index, 'Pickup Time'])
```

```
467                 self.param_dict['Pickup
Status'].set(self.data_frame.loc[data_index, 'Pickup Status'])
468                 self.param_dict['Drop off date 减 Pickup Date'].set(
469                     self.data_frame.loc[data_index, 'Drop off date
减 Pickup Date'])
470                 self.param_dict['Drop off
date'].set(self.data_frame.loc[data_index, 'Drop off date'])
471                 self.param_dict['Drop off
Time'].set(self.data_frame.loc[data_index, 'Drop off Time'])
472                 self.param_dict['Drop off
remark'].set(self.data_frame.loc[data_index, 'Drop off remark'])
473
474                 self.temp_window.insert('', 0, f'item{self.index}',
values=(
475                     self.param_dict['Tracking Code'].get(),
476                     self.param_dict['Drop off status'].get(),
477                     self.param_dict['Scheduled Delivery
Date'].get(),
478                     self.param_dict['Earliest Dropoff Time'].get(),
479                     self.param_dict['Latest Dropoff Time'].get(),
480                     self.param_dict['Shipment status'].get(),
481                     self.param_dict['Inbound Scan Date 减 Scheduled
Delivery Date'].get(),
482                     self.param_dict['Inbound Scan Date
(Linehaul)'].get(),
483                     self.param_dict['Inbound Scan Time'].get(),
484                     self.param_dict['Inbound status'].get(),
485                     self.param_dict['Pickup Date 减 Scheduled
Delivery Date'].get(),
486                     self.param_dict['Pickup Date'].get(),
487                     self.param_dict['Pickup Time'].get(),
488                     self.param_dict['Pickup Status'].get(),
489                     self.param_dict['Drop off date 减 Pickup
Date'].get(),
490                     self.param_dict['Drop off date'].get(),
491                     self.param_dict['Drop off Time'].get(),
492                     self.param_dict['Drop off remark'].get()
493                 ))
494                 self.temp_window.pack(pady=20)
495                 print(self.param_dict['Tracking Code'].get())
496
497     @staticmethod
498     def get_dict_from_tracking_code(tracking_code):
499         url = 'https://dataorch.axlehire.com/shipments/search'
```

```
500             header = {
501                 'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69
Safari/537.36',
502                 'content-type': 'application/json',
503                 'cookie': r'fp=1a39e1225ea764ca9f2abf599fafba34;
xtoken="dE9DbW1wYkZDI/B28g5MkirtzwljFDty7THWI75r/mVq4do8YKOJBeUtONSQ1
d3L1Yb5JCAEZPTk\012FFj7LXpbKjSaV71j1S6I9zjtTLurIi1ddgqe+xsIRU84cjgOSk
tu\012"'}
504             # 生成 post 的 json_data
505             data_dict = {'size': 15, 'q': tracking_code,
506                          'filters': {}, 'sorts': ['-
dropoff_earliest_ts']}
507             json_data = json.dumps(data_dict)
508
509             session = requests.Session()
510             user = 'yanxia.ji'
511             password = 'Axl12345'
512             response = session.post(url=url, headers=header,
data=json_data, auth=HttpNtlmAuth(user, password))
513
514             result_dict = json.loads(response.text)
515             return result_dict
516
517         def show_pic(self, event=None):
518             # 生成 dict_data
519             result_dict = self.get_dict_from_tracking_code(
520                 tracking_code=self.data_frame.loc[self.index,
'Tracking Code']
521             )
522             # data_dict = {'size': 15, 'q':
self.data_frame.loc[self.index, 'Tracking Code'],
523             #              'filters': {}, 'sorts': ['-
dropoff_earliest_ts']}
524             # json_data = json.dumps(data_dict)
525
526             session = requests.Session()
527             # user = 'yanxia.ji'
528             # password = 'Axl12345'
529             # response = session.post(url=self.url,
headers=self.header, data=json_data, auth=HttpNtlmAuth(user,
password))
530             #
531             # result_dict = json.loads(response.text)
```

```
532             # print(result_dict)
533
534             # 如果存在照片，就显示
535             if result_dict['results'][0]['pod']['images'] != []:
536                 # 如果是单张照片
537                 if len(result_dict['results'][0]['pod']['images'])
== 1:
538                     img_url =
result_dict['results'][0]['pod']['images'][0]['url']
539                     img_url_response = session.get(img_url)
540
541                     # 写入文件到 cache
542                     with open(f'tools/img_cache/{img_url[-
10:]}.png', 'wb') as fp:
543                         fp.write(img_url_response.content)
544
545                     if 'street2' in
result_dict['results'][0]['shipment']['dropoff_address'].keys():
546                         address_street2 =
result_dict['results'][0]['shipment']['dropoff_address']['street2'] +
' '
547                     else:
548                         address_street2 = "" + ' '
549                     address_street =
result_dict['results'][0]['shipment']['dropoff_address']['street'] +
' '
550                     address_city =
result_dict['results'][0]['shipment']['dropoff_address']['city'] + '
'
551                     address_state =
result_dict['results'][0]['shipment']['dropoff_address']['state'] + '
'
552                     address_zipcode =
result_dict['results'][0]['shipment']['dropoff_address']['zipcode'] +
' '
553                     address = address_street2 + address_street +
address_city + address_state + address_zipcode
554
555                     img = cv2.imread(f'tools/img_cache/{img_url[-
10:]}.png')
556                     img = process_image(img)
557                     cv2.imshow(f"address: {address}", img)
558                     cv2.waitKey()
559                 # 多张照片
```

```
560             if len(result_dict['results'][0]['pod']['images']) >
1:
561                 imgs = []
562                 for img_url in
result_dict['results'][0]['pod']['images']:
563                     img_url_response =
session.get(img_url['url'])
564
565                     # 写入文件到 cache
566                     with
open(f'tools/img_cache/{img_url["url"][-10:]}.png', 'wb') as fp:
567                         fp.write(img_url_response.content)
568
imgs.append(f'tools/img_cache/{img_url["url"][-10:]}.png')
569
570                 if 'street2' in
result_dict['results'][0]['shipment']['dropoff_address'].keys():
571                     address_street2 =
result_dict['results'][0]['shipment']['dropoff_address']['street2'] +
' '
572                 else:
573                     address_street2 = "" + ' '
574
575                 address_street =
result_dict['results'][0]['shipment']['dropoff_address']['street'] +
' '
576                 address_city =
result_dict['results'][0]['shipment']['dropoff_address']['city'] + '
'
577                 address_state =
result_dict['results'][0]['shipment']['dropoff_address']['state'] + '
'
578                 address_zipcode =
result_dict['results'][0]['shipment']['dropoff_address']['zipcode'] +
' '
579                 address = address_street2 + address_street +
address_city + address_state + address_zipcode
580
581                 # 依次显示照片
582                 for img in imgs:
583                     img = cv2.imread(img)
584                     img = process_image(img)
585                     cv2.imshow(f"address: {address} have more
than one pic", img)
```

```
                    cv2.waitKey()
        else:
            if 'street2' in
result_dict['results'][0]['shipment']['dropoff_address'].keys():
                address_street2 =
result_dict['results'][0]['shipment']['dropoff_address']['street2'] +
' '
            else:
                address_street2 = " "
            address_street =
result_dict['results'][0]['shipment']['dropoff_address']['street'] +
' '
            address_city =
result_dict['results'][0]['shipment']['dropoff_address']['city'] + '
'
            address_state =
result_dict['results'][0]['shipment']['dropoff_address']['state'] + '
'
            address_zipcode =
result_dict['results'][0]['shipment']['dropoff_address']['zipcode'] +
' '
            address = address_street2 + address_street +
address_city + address_state + address_zipcode

            messagebox.showinfo('没有照片', message=f'地址为:
{address}')
            self.window.focus_force()

    def clear_cache(self):
        if not os.path.exists('tools/img_cache'):
            os.mkdir('tools/img_cache')
        del_list = os.listdir('tools/img_cache')
        if len(del_list) == 0:
            messagebox.showinfo(title='清除失败', message='无缓
存')
            return
        file_size_sum = 0
        for f in del_list:
            file_path = os.path.join('tools/img_cache', f)
            if os.path.isfile(file_path):
                file_size_sum += self.get_filesize(file_path)
                os.remove(file_path)
            elif os.path.isdir(file_path):
                shutil.rmtree(file_path)
```

```
616             messagebox.showinfo(title='清除成功', message=f'清除缓存共 {round(file_size_sum, 1)}mb')
617
618         @staticmethod
619         def get_filesize(file_path):
620             file_size = os.path.getsize(file_path)
621             file_size = file_size / float(1024 * 1024)
622             return round(file_size, 2)
623
624         def confirm(self, event=None):
625             print(self.answer)
626             print(self.answer.get())
627             answer_index = self.answer.get()
628             print('answer_index', int(answer_index))
629             analyser_utils.copy_reason(
630                 data_frame_row=self.data_frame.iloc[self.index: self.index + 1, :],
631                 index=int(answer_index)
632             )
633             messagebox.showinfo(title='确定', message='您的输入已写入')
634             self.window.focus_force()
635
636         def hand_in_result(self):
637             # 生成 csv
638             date_time = datetime.datetime.now().strftime('%Y-%m-%d %H-%M-%S')
639             path = str(self.save_folder_path) + '/最终版' + date_time + '.csv'
640
641             res_df = self.write_in()
642
643             # 周三的需要改动列名
644             if self.day == '3':
645                 res_df.rename(columns={'AH Assessment': 'HF Reason Code'}, inplace=True)
646             # 这里 res_df 中的五列将带 x 的写回去，并 drop 掉
647             res_df.to_csv(path, index=False)
648
649             messagebox.showinfo(title='成功', message=f'已生成 {path}')
650
651         def write_in(self):
652             def get_index(tracking_code, source_df):
```

```
653                 return source_df[source_df['Tracking Code'] ==
tracking_code].index
654
655             data_frame = self.data_frame.copy()
656             source_df = self.source_df.copy()
657             # 把 data_frame 根据相同的 tracking code 将五列写入
source_df 返回 res_df
658             for index, row in data_frame.iterrows():
659                 source_df.loc[get_index(data_frame.loc[index,
'Tracking Code'], source_df),
660                               'Issue Category'] =
data_frame.loc[index, 'Issue Category']
661                 source_df.loc[get_index(data_frame.loc[index,
'Tracking Code'], source_df),
662                               'Delivery Comments'] =
data_frame.loc[index, 'Delivery Comments']
663                 source_df.loc[get_index(data_frame.loc[index,
'Tracking Code'], source_df),
664                               'AH Assessment'] =
data_frame.loc[index, 'AH Assessment']
665                 source_df.loc[get_index(data_frame.loc[index,
'Tracking Code'], source_df),
666                               'POD Quality'] = data_frame.loc[index,
'POD Quality']
667                 source_df.loc[get_index(data_frame.loc[index,
'Tracking Code'], source_df),
668                               'POD Valid?'] = data_frame.loc[index,
'POD Valid?']
669             return source_df
670
671         @staticmethod
672         def open_dictionary():
673             os.system(os.getcwd() + '/tools/files/dictionary.xlsx')
674
675
676     def process_image(img):
677         min_side = 768
678         size = img.shape
679         h, w = size[0], size[1]
680         # 长边缩放为min_side
681         scale = max(w, h) / float(min_side)
682         new_w, new_h = int(w / scale), int(h / scale)
683         resize_img = cv2.resize(img, (new_w, new_h))
684         # 填充至min_side * min_side
```

```
685         if new_w % 2 != 0 and new_h % 2 == 0:
686             top, bottom, left, right = (min_side - new_h) / 2,
(min_side - new_h) / 2, (min_side - new_w) / 2 + 1, (
687                         min_side - new_w) / 2
688         elif new_h % 2 != 0 and new_w % 2 == 0:
689             top, bottom, left, right = (min_side - new_h) / 2 + 1,
(min_side - new_h) / 2, (min_side - new_w) / 2, (
690                         min_side - new_w) / 2
691         elif new_h % 2 == 0 and new_w % 2 == 0:
692             top, bottom, left, right = (min_side - new_h) / 2,
(min_side - new_h) / 2, (min_side - new_w) / 2, (
693                         min_side - new_w) / 2
694         else:
695             top, bottom, left, right = (min_side - new_h) / 2 + 1,
(min_side - new_h) / 2, (min_side - new_w) / 2 + 1, (
696                         min_side - new_w) / 2
697         pad_img = cv2.copyMakeBorder(resize_img, int(top),
int(bottom), int(left), int(right), cv2.BORDER_CONSTANT,
698                                      value=[0, 0, 0])  # 从图像边界
向上,下,左,右扩的像素数目
699         return pad_img
700
701
702  class Thursday(object):
703         def __init__(self, init_df, policy):
704             self.init_df = init_df
705             self.policy = policy
706
707         def analyse(self):
708             res_data = self.init_df.copy()
709             result = pd.DataFrame(columns=res_data.columns)
710             for index, row in self.init_df.iterrows():
711                 # 修改 Scheduled Delivery Date 成为 %Y-%m-%d
712                 temp =
analyser_utils.change_Scheduled_Delivery_Date(res_data.iloc[index:
index + 1, :])
713                 # 填入 week ✓
714                 temp = analyser_utils.get_week_num(temp)
715                 # 修改时区
716                 # temp =
analyser_utils.data_frame_row_time_change(temp)
717                 res_data.iloc[index: index + 1, :] =
analyser_utils.get_status(temp, day='4', policy=self.policy)
718                 result = pd.concat([result, temp])
```

```
719             result['Updated Reason Code'] = result['AH Assessment']
720             return result
721
722
723     class Wednesday(object):
724         def __init__(self, init_df, policy):
725             self.init_df = init_df
726             self.policy = policy
727
728         def analyse(self):
729             res_data = self.init_df.copy()
730             res_data.rename(columns={'HF Reason Code': 'AH
Assessment', 'POD Qaulity': 'POD Quality'}, inplace=True)
731             result = pd.DataFrame(columns=res_data.columns)
732             for index, row in self.init_df.iterrows():
733                 # 修改 Scheduled Delivery Date 成为 %Y-%m-%d
734                 temp =
analyser_utils.change_Scheduled_Delivery_Date(res_data.iloc[index:
index + 1, :])
735                 # 填入 week ✓
736                 temp = analyser_utils.get_week_num(temp)
737                 # 修改时区
738                 # temp =
analyser_utils.data_frame_row_time_change(temp)
739                 # 分析 status
740                 res_data.iloc[index: index + 1, :] =
analyser_utils.get_status(temp, day='3', policy=self.policy)
741                 result = pd.concat([result, temp])
742             result['Updated Reason Code'] = result['AH Assessment']
743             return result
744
```

analyser_utils.py

```
import os
import datetime
import pandas as pd


DICT_DF = pd.read_excel(os.getcwd() +
'/utils/files/dictionary.xlsx')


def is_mouth_day_year(date):
    try:
        datetime.datetime.strptime(date, "%m/%d/%Y")
        return True
    except:
        return False


def get_week_num(data_frame_row):
    date_str = data_frame_row['Scheduled Delivery
Date'].values[0]
    if pd.isna(date_str):

        return data_frame_row
    # 如果 scheduled date 是月日年
    if is_mouth_day_year(date_str):
        res_date = datetime.datetime.strptime(date_str,
'%m/%d/%Y')
        data_frame_row['Week#'] = [res_date.isocalendar()[1]]
        return data_frame_row
    else:
        res_date = datetime.datetime.strptime(date_str,
'%Y-%m-%d')
        # res_date = datetime.datetime.strptime(date_str,
'%Y-%m-%d %H:%M:%S')
        data_frame_row['Week#'] = [res_date.isocalendar()[1]]
        return data_frame_row


def nan_to_none(x):
    if str(x) == 'nan' or pd.isna(x):
        return ''
    return x
```

```
39
40
41    def copy_reason(data_frame_row, index):
42        # print('values', data_frame_row['POD Valid?'].values[0])
43        # print('values', data_frame_row['Issue
Category'].values[0])
44        copy_df = DICT_DF[DICT_DF.index == index]
45        # 如果全是空的，直接复制
46        if pd.isna(data_frame_row['POD Valid?'].values[0]) and
pd.isna(data_frame_row['POD Quality'].values[0]) and \
47            pd.isna(data_frame_row['Issue Category'].values[0]) and
pd.isna(data_frame_row['Delivery Comments'].values[0]) and \
48            pd.isna(data_frame_row['AH Assessment'].values[0]):
49            # print('全是空的')
50            data_frame_row['POD Valid?'] = [nan_to_none(copy_df['POD
Valid?'].values[0])]
51            data_frame_row['POD Quality'] =
[nan_to_none(copy_df['POD Quality'].values[0])]
52            data_frame_row['Issue Category'] = copy_df['Issue
Category'].values
53            data_frame_row['Delivery Comments'] = copy_df['Delivery
Comments'].values
54            data_frame_row['AH Assessment'] = copy_df['AH
Assessment'].values
55            # print('写进去了')
56            return data_frame_row
57        # 如果不是空的，加一个 / 再将内容附着上
58        else:
59            # print('不是空的')
60            data_frame_row['POD Valid?'] =
[nan_to_none(str(copy_df['POD Valid?'].values[0]))]
61            data_frame_row['POD Quality'] =
[nan_to_none(str(copy_df['POD Quality'].values[0]))]
62            data_frame_row['Issue Category'] =
[str(data_frame_row['Issue Category'].values[0]) + '/' +
str(copy_df['Issue Category'].values[0])]
63            data_frame_row['Delivery Comments'] =
[str(data_frame_row['Delivery Comments'].values[0]) + '/' +
str(copy_df['Delivery Comments'].values[0])]
64            data_frame_row['AH Assessment'] =
[str(data_frame_row['AH Assessment'].values[0]) + '/' +
str(copy_df['AH Assessment'].values[0])]
65            # print('附着进去了')
66            return data_frame_row
```

```
67
68
69    def get_pickup_and_delivery_status(data_frame_row, day, policy):
70        pickup_diff = date_subtract(data_frame_row['Pickup
Date'].values[0],
71                                    data_frame_row['Scheduled
Delivery Date'].values[0])
72
73        # 如果 pick 当天送达
74        if pickup_diff == 0:
75            # 如果 pick 当天晚于 12
76            if time_upper_than(data_frame_row['Pickup
Time'].values[0], '12:00', 0):
77                # print('Pickup Time', data_frame_row['Pickup
Time'].values[0])
78                data_frame_row['Pickup Comments'] = ['Pickup after
12pm']
79
80                delivery_diff = date_subtract(data_frame_row['Drop
off date'].values[0],
81                                              data_frame_row['Pickup
Date'].values[0])
82                # 判断 delivery 当天
83                if delivery_diff == 0:
84                    if time_upper_than(data_frame_row['Drop off
time'].values[0],
85                                       data_frame_row['Latest
Dropoff Time'].values[0], policy):
86                        data_frame_row = copy_reason(data_frame_row,
118)
87                        return data_frame_row
88                # 配送晚于 2 天
89                else:
90                    data_frame_row = copy_reason(data_frame_row,
119)
91
92                    if delivery_diff == 1:
93                        return data_frame_row
94                    else:
95                        if delivery_diff < 2:
96                            return data_frame_row
97                        else:
98                            # data_frame_row['Delivery Comments'] =
[
```

```
99                          #     f'pickup ok but delivery late for
{delivery_diff} days']
100                         data_frame_row =
write_in_delivery_comments(data_frame_row,
101
f'pickup ok but delivery late for {delivery_diff} days')
102                         return data_frame_row
103         # 如果 pick 早于 12 点
104         else:
105             delivery_diff = date_subtract(data_frame_row['Drop
off date'].values[0],
106                                           data_frame_row['Pickup
Date'].values[0])
107             # 判断 delivery 当天
108             if delivery_diff == 0:
109                 if time_upper_than(data_frame_row['Drop off
time'].values[0],
110                                    data_frame_row['Latest
Dropoff Time'].values[0], policy):
111                     data_frame_row = copy_reason(data_frame_row,
118)
112                     return data_frame_row
113             # 配送晚于 2 天
114             else:
115                 data_frame_row = copy_reason(data_frame_row,
119)
116
117                 if delivery_diff == 1:
118                     return data_frame_row
119                 else:
120                     if delivery_diff < 2:
121                         return data_frame_row
122                     else:
123                         # data_frame_row['Delivery Comments'] =
[
124                         #     f'pickup ok but delivery late for
{delivery_diff} days']
125                         data_frame_row =
write_in_delivery_comments(data_frame_row,
126
f'pickup ok but delivery late for {delivery_diff} days')
127                         return data_frame_row
128         return data_frame_row
129
```

```
130         # 如果 pick 晚了 n 天
131         if pickup_diff > 1:
132             # 对于周三
133             if day == '3':
134                 data_frame_row = copy_reason(data_frame_row, 41)
135                 if pickup_diff == 1:
136                     # data_frame_row['Delivery Comments'] = [
137                     #     f'Inbound ontime but outbound late for 1
day']
138                     data_frame_row =
write_in_delivery_comments(data_frame_row,
139
f'Inbound ontime but outbound late for 1 day')
140                     data_frame_row['Pickup Comments'] = [
141                         f'Inbound ontime but outbound late for 1
day']
142
143                     delivery_diff =
date_subtract(data_frame_row['Drop off date'].values[0],
144
data_frame_row['Pickup Date'].values[0])
145                     # pickup 晚了一天, delivery 当天
146                     if delivery_diff == 0:
147                         # 当天晚了
148                         if time_upper_than(data_frame_row['Drop off
time'].values[0],
149                                            data_frame_row['Latest
Dropoff Time'].values[0], policy):
150                             data_frame_row =
copy_reason(data_frame_row, 118)
151
152                             data_frame_row =
write_in_delivery_comments(data_frame_row,
153
f'pickup late for {pickup_diff} day and  delivery late for same day')
154
155                             return data_frame_row
156                     # pickup 一天,  delivery 多天
157                     else:
158                         data_frame_row = copy_reason(data_frame_row,
119)
159                         if delivery_diff == 1:
160                             data_frame_row =
write_in_delivery_comments(data_frame_row,
```

```
393                 if 'cant be reach'.lower() in data_frame_row['Drop
off remark'].values[0].lower():
394                     data_frame_row = copy_reason(data_frame_row, 13)
395                     return data_frame_row
396                 # ok
397                 if 'closed'.lower() in data_frame_row['Drop off
remark'].values[0].lower():
398                     data_frame_row = copy_reason(data_frame_row, 14)
399                     return data_frame_row
400                 # ok
401                 if 'requested redelivery'.lower() in
data_frame_row['Drop off remark'].values[0].lower():
402                     data_frame_row = copy_reason(data_frame_row, 23)
403                     return data_frame_row
404                 # ok
405                 if 'redelivery requested'.lower() in
data_frame_row['Drop off remark'].values[0].lower():
406                     data_frame_row = copy_reason(data_frame_row, 23)
407                     return data_frame_row
408                 # ok
409                 if 'Cancel'.lower() in data_frame_row['Drop off
remark'].values[0].lower():
410                     data_frame_row = copy_reason(data_frame_row, 20)
411                     return data_frame_row
412                 # ok
413                 if 'refused'.lower() in data_frame_row['Drop off
remark'].values[0].lower():
414                     data_frame_row = copy_reason(data_frame_row, 19)
415
416         # 如果是 SUCCEEDED 状态
417         if data_frame_row['Drop off status'].values[0] ==
'SUCCEEDED':
418             # 先查看一些明显的问题
419             if 'no access'.lower() in str(data_frame_row['Drop off
remark'].values[0]).lower():
420                 data_frame_row = copy_reason(data_frame_row, 14)
421             elif 'no answer'.lower() in str(data_frame_row['Drop off
remark'].values[0]).lower():
422                 data_frame_row = copy_reason(data_frame_row, 14)
423             elif 'no code'.lower() in str(data_frame_row['Drop off
remark'].values[0]).lower():
424                 data_frame_row = copy_reason(data_frame_row, 14)
425
426             # 再看时间差，附着到之前的结果
```

```
427             inbound_diff = date_subtract(data_frame_row['Inbound
Scan Date (Linehaul)'].values[0],
428                                          data_frame_row['Scheduled
Delivery Date'].values[0])
429             # 如果  inbound_diff 等于 0
430             if inbound_diff == 0:
431                 # 如果 inbound 当天晚于 12 点
432                 if time_upper_than(data_frame_row['Inbound Scan
Time'].values[0], '12:00', 0):
433                     data_frame_row['Inbound Comments'] = ['Inbound
late']
434                     data_frame_row = copy_reason(data_frame_row, 24)
435                     return data_frame_row
436                 # 如果 inbound 当天早于 12 点
437                 else:
438                     get_pickup_and_delivery_status(data_frame_row,
day, policy)
439
440             # 如果 inbound_diff 大于一天
441             elif inbound_diff > 0:
442                 data_frame_row['Inbound Comments'] = ['Inbound
late']
443                 data_frame_row = copy_reason(data_frame_row, 24)
444                 # # 看 pick 减 inbound 的日子
445                 # pickup_diff = date_subtract(data_frame_row['Pickup
Date'].values[0],
446                 #
data_frame_row['Scheduled Delivery Date'].values[0]) - inbound_diff
447
448                 # inbound late 了，不看别的了
449                 # if pickup_diff > 0:
450                 #     if day == '3':
451                 #         data_frame_row =
copy_reason(data_frame_row, 41)
452                 #         if pickup_diff == 1:
453                 #             # data_frame_row['Delivery Comments']
= [
454                 #             #     f'Inbound ontime but outbound
late for 1 day']
455                 #             data_frame_row =
write_in_delivery_comments(data_frame_row, f'Inbound late and
outbound late for 1 day')
456                 #             data_frame_row['Pickup Comments'] = [
```

```
457             #                 f'Inbound late and outbound late
for 1 day']
458             #             return data_frame_row
459             #         else:
460             #             # data_frame_row['Delivery Comments']
= [
461             #             #     f'Inbound ontime but outbound
late for {pickup_diff} days']
462             #             data_frame_row =
write_in_delivery_comments(data_frame_row, f'Inbound late and
outbound late for {pickup_diff} days')
463             #             data_frame_row['Pickup Comments'] = [
464             #                 f'Inbound late and outbound late
for {pickup_diff} days']
465             #             return data_frame_row
466             #     if day == '4':
467             #         data_frame_row =
copy_reason(data_frame_row, 0)
468             #         if pickup_diff == 1:
469             #             # data_frame_row['Delivery Comments']
= [
470             #             #     f'Inbound ontime but outbound
late for 1 day']
471             #             data_frame_row =
write_in_delivery_comments(data_frame_row, f'Inbound late and
outbound late for 1 day')
472             #             data_frame_row['Pickup Comments'] = [
473             #                 f'Inbound late and outbound late
for 1 day']
474             #             return data_frame_row
475             #         else:
476             #             # data_frame_row['Delivery Comments']
= [
477             #             #     f'Inbound ontime but outbound
late for {pickup_diff} days']
478             #             data_frame_row =
write_in_delivery_comments(data_frame_row, f'Inbound late and
outbound late for {pickup_diff} days')
479             #             data_frame_row['Pickup Comments'] = [
480             #                 f'Inbound late and outbound late
for {pickup_diff} days']
481             #             return data_frame_row
482             #     return data_frame_row
483             return data_frame_row
```

```
484
485             # 当 Inbound 没 late
486             else:
487                 get_pickup_and_delivery_status(data_frame_row, day,
policy)
488
489         return data_frame_row
490
491
492     def date_subtract(compared_date, schedule_date):
493         if pd.isna(compared_date) or str(compared_date) == 'nan':
494             return -100
495         if pd.isna(schedule_date) or str(schedule_date) == 'nan':
496             return -100
497         else:
498             try:
499                 print(compared_date, schedule_date)
500                 compared_date =
datetime.datetime.strptime(compared_date, '%Y-%m-%d')
501                 # 如果 scheduled date 是月日年
502                 if is_mouth_day_year(schedule_date):
503                     schedule_date =
datetime.datetime.strptime(schedule_date, '%m/%d/%Y')
504                 else:
505                     # print('卧槽尼玛', schedule_date)
506                     schedule_date =
datetime.datetime.strptime(schedule_date, '%Y-%m-%d')
507                 return (compared_date - schedule_date).days
508             except ValueError:
509                 return -100
510
511
512     def time_upper_than(time_str, upper, policy):
513         upper_time = datetime.datetime.strptime(upper, '%H:%M')
514         upper_time += datetime.timedelta(minutes=int(policy))
515         time_str = datetime.datetime.strptime(time_str, '%H:%M')
516         # print(upper_time, '-', time_str, '=', end=' ')
517         # print(int(upper_time.strftime('%H%M')) -
int(time_str.strftime('%H%M')))
518         if (int(upper_time.strftime('%H%M')) -
int(time_str.strftime('%H%M'))) > 0:
519             return False
520         else:
521             return True
```

```
522
523
524 def write_in_delivery_comments(data_frame_row, string):
525     if pd.isna(data_frame_row['Delivery Comments'].values[0]):
526         data_frame_row['Delivery Comments'] = [string]
527         return data_frame_row
528     # 如果已经有了，就不管了
529     elif string in data_frame_row['Delivery
Comments'].values[0]:
530         return data_frame_row
531     else:
532         # data_frame_row['Delivery Comments'] =
[data_frame_row['Delivery Comments'].values[0] + '/' + string]
533         data_frame_row['Delivery Comments'] = [string]
534         return data_frame_row
535
536
537 def data_frame_row_time_change(data_frame_row):
538     region = data_frame_row['Region Code'].values[0]
539
540     try:
541         if pd.isna(region):
542             return data_frame_row
543
544         # 判断 Region Code 属于那个地区
545         elif region == 'CHI' or region == 'DFW' or region ==
'HOU':
546             # early 时间 latest 时间
547             early_time_str = str(data_frame_row['Earliest
Dropoff Time'].values[0])
548             new_time = time_subtract(early_time_str, hours=2,
days=0)
549             new_time_str = new_time.strftime('%H:%M')
550             data_frame_row['Earliest Dropoff Time'] =
[new_time_str]
551
552             latest_time_str = str(data_frame_row['Latest Dropoff
Time'].values[0])
553             new_time = time_subtract(latest_time_str, hours=2,
days=0)
554             new_time_str = new_time.strftime('%H:%M')
555             data_frame_row['Latest Dropoff Time'] =
[new_time_str]
556             # 针对 inbound
```

```
557             # 如果 时间有空的，跳过
558             if pd.isna(data_frame_row['Inbound Scan
Time'].values[0]):
559                 new_time = None
560             else:
561                 inbound_time_str = str(data_frame_row['Inbound
Scan Time'].values[0])
562                 new_time = time_subtract(inbound_time_str,
hours=2, days=0)
563                 new_time_str = new_time.strftime('%H:%M')
564                 data_frame_row['Inbound Scan Time'] =
[new_time_str]
565
566             # 针对 pickup time
567             if pd.isna(data_frame_row['Pickup Time'].values[0]):
568                 pass
569             else:
570                 pickup_time_str = str(data_frame_row['Pickup
Time'].values[0])
571                 new_pickup_time = time_subtract(pickup_time_str,
hours=2, days=0)
572                 new_pickup_time_str =
new_pickup_time.strftime('%H:%M')
573                 data_frame_row['Pickup Time'] =
[new_pickup_time_str]
574
575             # 针对 drop off time
576             if pd.isna(data_frame_row['Drop off
time'].values[0]):
577                 pass
578             else:
579                 drop_time_str = str(data_frame_row['Drop off
time'].values[0])
580                 new_drop_time = time_subtract(drop_time_str,
hours=2, days=0)
581                 new_drop_time_str =
new_drop_time.strftime('%H:%M')
582                 data_frame_row['Drop off time'] =
[new_drop_time_str]
583
584             # 如果前进了一天
585             if new_time is None:
586                 return data_frame_row
587             else:
```

```
588                     if str(new_time.date()) == '1899-12-31':
589                         date_str = str(data_frame_row['Inbound Scan
Date (Linehaul)'].values[0])
590
591                         time_object =
datetime.datetime.strptime(date_str, '%Y-%m-%d')
592                         new_date = time_object -
datetime.timedelta(days=1)
593                         new_date_str = new_date.strftime('%Y-%m-%d')
594
595                         data_frame_row['Inbound Scan Date
(Linehaul)'] = [new_date_str]
596                         return data_frame_row
597                     else:
598                         return data_frame_row
599
600             elif region == 'JFK' or region == 'PHL' or region ==
'EWR':
601                 # early 时间 latest 时间
602                 early_time_str = str(data_frame_row['Earliest
Dropoff Time'].values[0])
603                 new_time = time_subtract(early_time_str, hours=3,
days=0)
604                 new_time_str = new_time.strftime('%H:%M')
605                 data_frame_row['Earliest Dropoff Time'] =
[new_time_str]
606
607                 latest_time_str = str(data_frame_row['Latest Dropoff
Time'].values[0])
608                 new_time = time_subtract(latest_time_str, hours=3,
days=0)
609                 new_time_str = new_time.strftime('%H:%M')
610                 data_frame_row['Latest Dropoff Time'] =
[new_time_str]
611                 # 针对 inbound
612                 # 如果 时间有空的，跳过
613                 if pd.isna(data_frame_row['Inbound Scan
Time'].values[0]):
614                     new_time = None
615                 else:
616                     inbound_time_str = str(data_frame_row['Inbound
Scan Time'].values[0])
617                     new_time = time_subtract(inbound_time_str,
hours=3, days=0)
```

```
618                     new_time_str = new_time.strftime('%H:%M')
619                     data_frame_row['Inbound Scan Time'] =
[new_time_str]
620
621                 # 针对 pickup time
622                 if pd.isna(data_frame_row['Pickup Time'].values[0]):
623                     pass
624                 else:
625                     pickup_time_str = str(data_frame_row['Pickup
Time'].values[0])
626                     new_pickup_time = time_subtract(pickup_time_str,
hours=3, days=0)
627                     new_pickup_time_str =
new_pickup_time.strftime('%H:%M')
628                     data_frame_row['Pickup Time'] =
[new_pickup_time_str]
629
630                 # 针对 drop off time
631                 if pd.isna(data_frame_row['Drop off
time'].values[0]):
632                     pass
633                 else:
634                     drop_time_str = str(data_frame_row['Drop off
time'].values[0])
635                     new_drop_time = time_subtract(drop_time_str,
hours=3, days=0)
636                     new_drop_time_str =
new_drop_time.strftime('%H:%M')
637                     data_frame_row['Drop off time'] =
[new_drop_time_str]
638
639                 # 如果前进了一天
640                 if new_time is None:
641                     return data_frame_row
642                 else:
643                     if str(new_time.date()) == '1899-12-31':
644                         date_str = str(data_frame_row['Inbound Scan
Date (Linehaul)'].values[0])
645
646                         time_object =
datetime.datetime.strptime(date_str, '%Y-%m-%d')
647                         new_date = time_object -
datetime.timedelta(days=1)
648                         new_date_str = new_date.strftime('%Y-%m-%d')
```

```
649
650                     data_frame_row['Inbound Scan Date
(Linehaul)'] = [new_date_str]
651                     return data_frame_row
652                 else:
653                     return data_frame_row
654
655         elif region == 'PHX':
656             # early 时间 latest 时间
657             early_time_str = str(data_frame_row['Earliest
Dropoff Time'].values[0])
658             new_time = time_subtract(early_time_str, hours=1,
days=0)
659             new_time_str = new_time.strftime('%H:%M')
660             data_frame_row['Earliest Dropoff Time'] =
[new_time_str]
661
662             latest_time_str = str(data_frame_row['Latest Dropoff
Time'].values[0])
663             new_time = time_subtract(latest_time_str, hours=1,
days=0)
664             new_time_str = new_time.strftime('%H:%M')
665             data_frame_row['Latest Dropoff Time'] =
[new_time_str]
666             # 针对 inbound
667             # 如果 时间有空的，跳过
668             if pd.isna(data_frame_row['Inbound Scan
Time'].values[0]):
669                 new_time = None
670             else:
671                 inbound_time_str = str(data_frame_row['Inbound
Scan Time'].values[0])
672                 new_time = time_subtract(inbound_time_str,
hours=1, days=0)
673                 new_time_str = new_time.strftime('%H:%M')
674                 data_frame_row['Inbound Scan Time'] =
[new_time_str]
675
676             # 针对 pickup time
677             if pd.isna(data_frame_row['Pickup Time'].values[0]):
678                 pass
679             else:
680                 pickup_time_str = str(data_frame_row['Pickup
Time'].values[0])
```

```
                    new_pickup_time = time_subtract(pickup_time_str,
hours=1, days=0)
                    new_pickup_time_str =
new_pickup_time.strftime('%H:%M')
                    data_frame_row['Pickup Time'] =
[new_pickup_time_str]

                # 针对 drop off time
                if pd.isna(data_frame_row['Drop off
time'].values[0]):
                    pass
                else:
                    drop_time_str = str(data_frame_row['Drop off
time'].values[0])
                    new_drop_time = time_subtract(drop_time_str,
hours=1, days=0)
                    new_drop_time_str =
new_drop_time.strftime('%H:%M')
                    data_frame_row['Drop off time'] =
[new_drop_time_str]

                # 如果前进了一天
                if new_time is None:
                    return data_frame_row
                else:
                    if str(new_time.date()) == '1899-12-31':
                        date_str = str(data_frame_row['Inbound Scan
Date (Linehaul)'].values[0])

                        time_object =
datetime.datetime.strptime(date_str, '%Y-%m-%d')
                        new_date = time_object -
datetime.timedelta(days=1)
                        new_date_str = new_date.strftime('%Y-%m-%d')

                        data_frame_row['Inbound Scan Date
(Linehaul)'] = [new_date_str]
                        return data_frame_row
                    else:
                        return data_frame_row

            else:
                return data_frame_row
        except ValueError:
```

```
713             return data_frame_row
714
715
716     def time_subtract(time_str, hours, days):
717         time_object = datetime.datetime.strptime(time_str, '%H:%M')
718         new_time = time_object - datetime.timedelta(hours=hours,
days=days)
719         return new_time
720
721
722     def change_Scheduled_Delivery_Date(data_frame_row):
723         # if pd.isna(data_frame_row['Scheduled Delivery Date'][0]):
724         #     return data_frame_row
725         # else:
726         s_date_str = data_frame_row['Scheduled Delivery
Date'].values[0]
727         if format_1(s_date_str):
728             s_str = datetime.datetime.strptime(s_date_str,
'%Y/%m/%d')
729             s_str = s_str.strftime('%Y-%m-%d')
730             data_frame_row['Scheduled Delivery Date'] = [s_str]
731             return data_frame_row
732         elif format_2(s_date_str):
733             s_str = datetime.datetime.strptime(s_date_str,
'%m/%d/%Y')
734             s_str = s_str.strftime('%Y-%m-%d')
735             data_frame_row['Scheduled Delivery Date'] = [s_str]
736             return data_frame_row
737         elif format_3(s_date_str):
738             return data_frame_row
739         else:
740             return data_frame_row
741
742
743     def format_1(date):
744         try:
745             datetime.datetime.strptime(date, "%Y/%m/%d")
746             return True
747         except:
748             return False
749
750
751     def format_2(date):
752         try:
```

```
753             datetime.datetime.strptime(date, "%m/%d/%Y")
754             return True
755         except:
756             return False
757 
758 
759 def format_3(date):
760     try:
761         datetime.datetime.strptime(date, "%Y-%m-%d")
762         return True
763     except:
764         return False
765 
766 
767 if __name__ == '__main__':
768     d = '2021/2/2'
769     g = datetime.datetime.strptime(d, '%Y/%m/%d')
770     print(g)
```

concat_csv.py

```python
import pandas as pd
import os

from tkinter import Toplevel, StringVar, Label, Button, Entry, messagebox
from tkinter.filedialog import askdirectory


class Concat(object):
    def __init__(self, root):
        self.root = root

    def get_folder_path(self):
        folder_path = askdirectory()
        self.folder_path.set(folder_path)

    def concat_from_folder(self):
        dir_list = os.listdir(self.folder_path.get())
        res_df = pd.DataFrame()
        for file in dir_list:
            file_path = self.folder_path.get() + '/' + file
            temp_df = pd.read_csv(file_path)
            res_df = pd.concat([res_df, temp_df])
        res_df.to_csv(self.folder_path.get() + '/all.csv', index=False)
        path = self.folder_path.get() + '/all.csv'
        messagebox.showinfo(title='成功', message=f'输出路径为:{path}')
        return res_df

    def run(self):
        self.window = Toplevel(master=self.root)
        self.window.geometry('1000x120')
        self.folder_path = StringVar()

        # label ending
        Label(self.window, text="要合并的文件夹:").place(x=100, y=50)
        Entry(self.window, textvariable=self.folder_path, width='60').place(x=220, y=50)
        Button(self.window, text="选择文件夹", command=self.get_folder_path, width='10').place(x=680, y=50)

```

```
38            # button
39            Button(self.window, text='生成', width='10',
command=self.concat_from_folder).place(x=780, y=50)
40            print(self.window.focus)
41            self.window.mainloop()
42
```

downloader.py

```
import os
import datetime
import time
import requests
import json
import pandas as pd
from requests_ntlm import HttpNtlmAuth
from tkinter import Toplevel, Label, Entry, Button, StringVar,
messagebox


class DownLoader(object):
    def __init__(self, root=None):
        self.root = root
        self.window = Toplevel(master=self.root)
        self.url =
'https://dataorch.beta.axlehire.com/reports/all/request'
        self.header = {
            'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) '
                          'Chrome/95.0.4638.69 Safari/537.36',
            'content-type': 'application/json',
            'cookie': r'fp=1a39e1225ea764ca9f2abf599fafba34;
xtoken="dE9DbW1wYkZDI/B28g5MkirtzwljFDty7THWI75r/mVq4do8Y'

r'KOJBeUtONSQ1d3L1Yb5JCAEZPTk\012FFj7LXpbKjSaV71j1S6I9zjtTLurIi1ddgqe
+xsIRU84cjgOSktu\012"'
        }
        self.user_name = 'yanxia.ji'
        self.password = 'Axl12345'

    def download_from_url(self, url, file_name):
        print(url)
        # url =
'https://dataorch.beta.axlehire.com/reports/uploaded/8d38816a-98f4-
4461-9d4b-1f79cd360e34/download'
        self.make_dir('all_report_history')
        session = requests.Session()
        time.sleep(5)
        response = session.get(url=url, headers=self.header)
        # response = session.get(url=url, headers=self.header)
        if response.status_code == 200:
            json_data = json.loads(response.text)
```

```
36                 print('json_data', json_data)
37                 response = session.get(url=url+'/download',
headers=self.header)
38                 if 'url' in json_data.keys():
39                     with open(file_name, 'wb') as fp:
40                         fp.write(response.content)
41                 else:
42                     self.download_from_url(url, file_name)
43
44             else:
45                 self.download_from_url(url, file_name)
46
47         def get_csv_from_date(self, client_id, date, file_name):
48             session = requests.Session()
49             if client_id.find(', ') == -1:
50                 client_id_string = [str(client_id)]
51             else:
52                 client_id_string = client_id.split(', ')
53
54             # date
55             date = datetime.datetime.strptime(date,
'%Y/%m/%d').strftime('%Y-%m-%d')
56
57             json_data = {
58                 'clients': client_id_string,
59                 'date': date,
60             }
61
62             response = session.post(url=self.url,
headers=self.header, json=json_data,
63
auth=HttpNtlmAuth(self.user_name, self.password))
64             json_response = json.loads(response.content)
65             url =
'https://dataorch.beta.axlehire.com/reports/uploaded/'
66             url += json_response['id']
67             self.download_from_url(url, file_name)
68
69         def get_date_list(self, from_date, to_date):
70             from_date = datetime.datetime.strptime(from_date,
'%Y/%m/%d')
71             to_date = datetime.datetime.strptime(to_date,
'%Y/%m/%d')
72             diff = (to_date - from_date).days
```

```
73            if diff <= 0:
74                messagebox.showwarning(title='警告', message='日期范
围有误')
75            else:
76                date_list = [from_date.strftime('%Y/%m/%d')]
77                for i in range(1, diff+1):
78                    date = from_date + datetime.timedelta(days=i)
79                    date = date.strftime('%Y/%m/%d')
80                    date_list.append(date)
81                return date_list
82
83        @staticmethod
84        def make_dir(path):
85            if not os.path.exists(path):
86                os.mkdir(path)
87
88        def run(self):
89            # 初始化界面
90            self.window.geometry('700x240')
91            self.client = StringVar()
92            self.date = StringVar()
93            Label(self.window, text='输入 client 号（如有多个，请
用，（中文逗号）分隔）:').place(x=50, y=20)
94            Entry(self.window, textvariable=self.client).place(x=50,
y=60)
95            Label(self.window, text='输入日期（形如 2021/11/07 如有
多个日期请用,分隔,如果为时间段,请输入形如 2021/11/07-2021/11/09）:')\
96                .place(x=50, y=100)
97            Entry(self.window, textvariable=self.date).place(x=50,
y=140)
98            Button(self.window, text='生成 all_report csv',
command=self.confirm).place(x=50, y=190)
99            self.window.mainloop()
100
101       @staticmethod
102       def is_date(date):
103           try:
104               datetime.datetime.strptime(date, "%Y/%m/%d")
105               return True
106           except:
107               return False
108
109       def confirm(self):
110           if self.check_client():
```

```
111                 # date 为 range
112                 now = datetime.datetime.now().strftime('%m 月%d 日-%H
点%M 分%S 秒')
113                 if self.date.get().find('-') != -1:
114                     date_from_to_list = self.date.get().split('-')
115                     date_list =
self.get_date_list(from_date=date_from_to_list[0],
to_date=date_from_to_list[1])
116                     # 创建文件夹
117                     folder_name = self.date.get().replace('-', 'to')
118                     folder_name = folder_name.replace('/', '-')
119                     folder_name += '&client=' + self.client.get() +
'_' + now
120
self.make_dir(f'all_report_history/{folder_name}')
121                     for date in date_list:
122                         date_name = date.replace('/', '-')
123                         self.get_csv_from_date(
124                             client_id=self.client.get(),
125                             date=date,
126
file_name=f'all_report_history/{folder_name}/client={self.client.get(
)}&date='f'{date_name}&{now}.csv'
127                         )
128
self.concat_from_folder(f'all_report_history/{folder_name}')
129                 # 单个 date
130                 elif self.is_date(self.date.get()):
131                     date =
datetime.datetime.strptime(self.date.get(),
'%Y/%m/%d').strftime('%Y-%m-%d')
132                     self.get_csv_from_date(
133                         client_id=self.client.get(),
134                         date=self.date.get(),
135
file_name=f'all_report_history/client={self.client.get()}&date={date}
&{now}.csv'
136                     )
137                 else:
138                     messagebox.showwarning(title='警告', message='日
期格式有误')
139             else:
140                 messagebox.showwarning(title='警告', message='client
格式有误')
```

```
141
142     def check_client(self):
143         # client 是单个
144         if self.client.get().find(', ') == -1:
145             if not self.client.get().isnumeric():
146                 return False
147             # 是数字
148             else:
149                 if int(self.client.get()) <= 11 or
int(self.client.get()) == 471 or int(self.client.get()) == 621 \
150                         or (int(self.client.get()) >= 15 and
int(self.client.get()) <= 214):
151                     return True
152                 return False
153         # client 是多个
154         else:
155             client_list = self.client.get().split(', ')
156             for client in client_list:
157                 if client.find(', ') == -1:
158                     if not client.isnumeric():
159                         return False
160                     # 是数字
161                     else:
162                         if int(client) <= 214 or int(client) ==
471 or int(client) == 621:
163                             return True
164                         return False
165
166     @staticmethod
167     def get_dict_from_tracking_code(tracking_code):
168         url = 'https://dataorch.axlehire.com/shipments/search'
169         header = {
170             'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69
Safari/537.36',
171             'content-type': 'application/json',
172             'cookie': r'fp=1a39e1225ea764ca9f2abf599fafba34;
xtoken="dE9DbW1wYkZDI/B28g5MkirtzwljFDty7THWI75r/mVq4do8YKOJBeUtONSQ1
d3L1Yb5JCAEZPTk\012FFj7LXpbKjSaV71j1S6I9zjtTLurIi1ddgqe+xsIRU84cjgOSk
tu\012"'}
173         # 生成 post 的 json_data
174         data_dict = {'size': 15, 'q': tracking_code,
175                      'filters': {}, 'sorts': ['-
dropoff_earliest_ts']}
```

```
176             json_data = json.dumps(data_dict)
177
178             session = requests.Session()
179             user = 'yanxia.ji'
180             password = 'Axl12345'
181             response = session.post(url=url, headers=header,
data=json_data, auth=HttpNtlmAuth(user, password))
182
183             result_dict = json.loads(response.text)
184             return result_dict
185
186         @staticmethod
187         def concat_from_folder(folder_path):
188             dir_list = os.listdir(folder_path)
189             res_df = pd.DataFrame()
190             for file in dir_list:
191                 file_path = folder_path + '/' + file
192                 temp_df = pd.read_csv(file_path)
193                 res_df = pd.concat([res_df, temp_df])
194             res_df.to_csv(folder_path + '/all.csv', index=False)
195             path = folder_path + '/all.csv'
196             messagebox.showinfo(title='成功', message=f'输出路径为:
{path}')
197             return res_df
198
199
200     if __name__ == '__main__':
201         downloader = DownLoader()
202         # downloader.get_csv_from_date(client_id='159', date='2021-
10-19')2021-10-19
203
204         # downloader.get_csv_from_date(
205         #     client_id=['49'],
206         #     date='2021-11-17',
207         #     file_name='ff.csv'
208         # )
209         #
downloader.download_from_url(url='https://dataorch.beta.axlehire.com/
reports/uploaded/f705aa7f-6b32-4868-9892-5fb9ab6e138d',
file_name='ff')
210         # st = '2021/11/17'
211         # s = str(int(datetime.datetime.strptime(st,
'%Y/%m/%d').timestamp()))
212         # print(s)
```

```
213         downloader.run()
214
```

generator.py

```
"""
    填完数字之后，传回来加数字的 csv，点击生成，出结果
"""
import warnings
from tkinter import Toplevel

import pandas as pd


class Generator(object):
    def __init__(self, root=None):
        self.root = root
        self.window = Toplevel(master=self.root)
        self.reason_code = pd.read_csv('utils/files/JJ - Reason code.csv')

    def options(self):
        pass

    def get_final(self, csv_file):
        # 1. 读入传入的 csv
        res_df = pd.read_csv(csv_file)
        res_df.rename(columns={'HF Reason Code': 'AH Assessment'}, inplace=True)
        # 2. 遍历 res_df
        for idx, row in res_df.iterrows():
            res_df.iloc[idx: idx + 1, :] = self.parse_rows(res_df.iloc[idx: idx + 1, :])
            # print(self.parse_rows(res_df.iloc[idx: idx + 1, :]))
            print(f'\rwrite {idx} rows', end='')
        return res_df

    def parse_rows(self, data_frame_rows):
        # 解析每一行
        if pd.isna(data_frame_rows['Answer Number'].values[0]):
            return data_frame_rows

        number = int(data_frame_rows['Answer Number'].values[0]) \
            if type(data_frame_rows['Answer Number'].values[0]) == 'float' else data_frame_rows['Answer Number'].values[0]

```

```
38            # 不是数字，一定是 14 15 apt 这种形式
39            if not str(number).isnumeric():
40                answer_list = str(number).split(' ')
41                # 三种形式 pic shows building #,the correct is #
42                if 'apt' in answer_list:
43                    data_frame_rows =
self.copy_rows(data_frame_rows, int(109))
44                    apt_answer = str(data_frame_rows['POD
Quality'].values[0]). \
45                        replace('shows apt #', f'shows apt
#{answer_list[0]}'). \
46                        replace('the correct is apt#', f'the correct
is apt#{answer_list[1]}')
47                    data_frame_rows['POD Quality'] = [apt_answer]
48                    return data_frame_rows
49                elif 'st' in answer_list:
50                    data_frame_rows =
self.copy_rows(data_frame_rows, int(108))
51                    s_answer = str(data_frame_rows['POD
Quality'].values[0]). \
52                        replace('shows street #', f'shows street
#{answer_list[0]}'). \
53                        replace('the correct is #', f'the correct is
#{answer_list[1]}')
54                    data_frame_rows['POD Quality'] = [s_answer]
55                    return data_frame_rows
56                elif 'b' in answer_list:
57                    data_frame_rows =
self.copy_rows(data_frame_rows, int(107))
58                    b_answer = str(data_frame_rows['POD
Quality'].values[0]). \
59                        replace('shows building #', f'shows building
#{answer_list[0]}'). \
60                        replace('the correct is #', f'the correct is
#{answer_list[1]}')
61                    data_frame_rows['POD Quality'] = [b_answer]
62                    return data_frame_rows
63                else:
64                    return data_frame_rows
65            else:
66                data_frame_rows = self.copy_rows(data_frame_rows,
int(number))
67                return data_frame_rows
68
```

```
69        def copy_rows(self, data_frame_row, index):
70            pd.set_option("display.max_columns", 50)
71
72            # print(data_frame_row, 'index', index)
73            def nan_to_none(x):
74                if str(x) == 'nan' or pd.isna(x):
75                    return ''
76                return x
77
78            if pd.isna(data_frame_row['POD Valid?'].values[0]) and
pd.isna(data_frame_row['POD Quality'].values[0]) and \
79                    pd.isna(data_frame_row['Issue
Category'].values[0]) and pd.isna(
80                data_frame_row['Delivery Comments'].values[0]) and \
81                    pd.isna(data_frame_row['AH
Assessment'].values[0]):
82                data_frame_row['POD Valid?'] =
[nan_to_none(self.reason_code.loc[index, 'POD'])]
83                data_frame_row['POD Quality'] =
[nan_to_none(self.reason_code.loc[index, 'POD Qaulity'])]
84                data_frame_row['Issue Category'] =
[self.reason_code.loc[index, 'Issue Category']]
85                data_frame_row['Delivery Comments'] =
[self.reason_code.loc[index, 'Delivery Comments']]
86                data_frame_row['AH Assessment'] =
[self.reason_code.loc[index, 'AH Assignment']]
87
88                return data_frame_row
89
90            # 如果不是空的，加一个 / 再将内容附着上
91            else:
92                # print('不是空的')
93                data_frame_row['POD Valid?'] =
[nan_to_none(self.reason_code.loc[index, 'POD'])]
94                data_frame_row['POD Quality'] =
[nan_to_none(self.reason_code.loc[index, 'POD Qaulity'])]
95                if index == 122 or index == 123:
96                    return data_frame_row
97                # 如果本来就有，比如已经是 Delivery 了，你再加个
Delivery 就不对了
98                if self.reason_code.loc[index, 'Issue Category'] in
str(data_frame_row['Issue Category'].values[0]):
99                    pass
100               else:
```

```
                data_frame_row['Issue Category'] = [
                    str(data_frame_row['Issue
Category'].values[0]) + '/' + self.reason_code.loc[
                        index, 'Issue Category']]
            data_frame_row['Delivery Comments'] = [
                str(data_frame_row['Delivery
Comments'].values[0]) + '/' + self.reason_code.loc[
                    index, 'Delivery Comments']]
            data_frame_row['AH Assessment'] = [
                str(data_frame_row['AH Assessment'].values[0]) +
'/' + self.reason_code.loc[index, 'AH Assignment']]
            # print('附着进去了')
            return data_frame_row


def run(files, dis_files):
    generator = Generator()
    pd.set_option('display.max_columns', 50)
    # print(generator.reason_code.iloc[122:123, :])
    # print(generator.reason_code.iloc[123:124, :])
    df = generator.get_final(
        csv_file=files
    )
    # print(df)
    df.to_csv(dis_files, index=False)


```

preprocessing_data.py

```
import re
import pandas as pd


def preprocessing_data(ending_df, boss2me_df, all_report_df,
day):
    if day == '4':
        # 0. 获取文件
        big_sheet = ending_df
        boss2me = boss2me_df
        report = all_report_df

        # 3. initialize res_data
        columns_list = list(big_sheet.columns)
        columns_list.append('Earliest Dropoff Date')
        columns_list.append('Latest Dropoff Date')
        res_data = pd.DataFrame(columns=big_sheet.columns,
dtype='object')

        # 1. 将 boss "tracking code" 改为和 report "Tracking
Code" 一致
        # boss 的 tracking code 有多种可能 "tracking #" or
"Tracking Number"
        boss2me = change_title_name(boss2me,
re.search(r'\'[Tt]racking(#| Number| code| Code|_code|_Code)\'',

str(boss2me.columns)).group(0)[1:-1], "Tracking Code")

        # 2. 合并 boss 和 report 合并为 same
        same = pd.merge(boss2me, report, how='left',
on='Tracking Code')

        # 4. 将 res_data 的一些标题改为 same 的
        # ending 与 same 的不同除 Region Code --> Region, REgion
Code --> Region
        # ending_wednesday 的 Drop off Time ending_thursday 是
Drop off time --> Dropoff Time
        res_data = change_title_name(res_data,
re.search(r'\'[Tt]racking(#| Number| code| Code)\'',

str(res_data.columns)).group(0)[1:-1], "Tracking Code")
```

```
            res_data = change_title_name(res_data,
re.search(r'\'((Region Code)|(REgion Code)|(region Code)|(rEgion
Code)|(Region code)|(REgion code))\'',

str(res_data.columns)).group(0)[1:-1], "Region")
            res_data = change_title_name(res_data, 'Assignment ID',
'Assignment Id')
            res_data = change_title_name(res_data,
re.search(r'\'(Issue)|(Reason for Complaint)\'',
str(res_data.columns)).group(0)[:-1], 'Reason for Complaint')
            res_data = change_title_name(res_data, 'Inbound Scan
Date (Linehaul)', 'Inbound Scan Date')
            res_data = change_title_name(res_data, 'Pickup remark',
'Pickup Remark')
            res_data = change_title_name(res_data, 'Drop off date',
'Dropoff Date')
            res_data = change_title_name(res_data,
re.search(r'\'Drop off [Tt]ime\'',

str(res_data.columns)).group(0)[1:-1], "Dropoff Time")
            res_data = change_title_name(res_data, 'Drop off
status', 'Dropoff Status')
            res_data = change_title_name(res_data, 'Drop off
remark', 'Dropoff Remark')
            res_data = change_title_name(res_data, 'Requested
Amount', 'Requested Credit Amount')

            # 解决 Reason for complaint 问题
            if 'Issue' in same.columns:
                res_data['Reason for Complaint'] = same['Issue']

            # 5. 遍历 same 的标题，将 same 的数据写入 same 和
res_data 共有的标题下
            # print('\nres', res_data.columns)
            for title in same.columns:
                if title in res_data.columns:
                    res_data[title] = same[title]
                    # print(title + ' success to write in')

            # 6. 将 res_data 的标题重置为 ending 的标题
            res_data.columns = big_sheet.columns

            return res_data

```

```
60        elif day == '3':
61            print(ending_df.columns)
62            print(all_report_df.columns)
63            # 0. 获取文件
64            big_sheet = ending_df
65            boss2me = boss2me_df
66            report = all_report_df
67
68            # 3. initialize res_data
69            columns_list = list(big_sheet.columns)
70            columns_list.append('delivery_date')
71            columns_list.append('Earliest Dropoff Time')
72            columns_list.append('Latest Dropoff Time')
73            columns_list.append('Earliest Dropoff Date')
74            columns_list.append('Latest Dropoff Date')
75            res_data = pd.DataFrame(columns=columns_list,
dtype='object')
76
77            # 1. 将boss "tracking code" 改为和 report "Tracking
Code" 一致
78            # boss 的 tracking code 有多种可能 "tracking #" or
"Tracking Number"
79            boss2me = change_title_name(boss2me,
re.search(r'\'[Tt]racking(#| Number| code| Code|_code|_Code|)\'',
80
str(boss2me.columns)).group(0)[1:-1], "Tracking Code")
81
82            # 2. 合并 boss 和 report 合并为 same
83            same = pd.merge(boss2me, report, how='left',
on='Tracking Code')
84
85            # 4. 将 res_data 的一些标题改为 same 的
86            # ending 与 same 的不同除 Region Code --> Region, REgion
Code --> Region
87            # ending_wednesday 的 Drop off Time ending_thursday是
Drop off time --> Dropoff Time
88            res_data = change_title_name(res_data,
re.search(r'\'[Tt]racking(#| Number| code| Code)\'',
89
str(res_data.columns)).group(0)[1:-1], "Tracking Code")
90            res_data = change_title_name(res_data, re.search(
91                r'\'((Region Code)|(REgion Code)|(region
Code)|(rEgion Code)|(Region code)|(REgion code))\'',
92                str(res_data.columns)).group(0)[1:-1], "Region")
```

```
            res_data = change_title_name(res_data, 'Assignment ID',
'Assignment Id')
            res_data = change_title_name(res_data,

re.search(r'\'(Issue)|(Reason for Complaint)\'',
str(res_data.columns)).group(0)[
                                         :-1], 'Reason for
Complaint')
            res_data = change_title_name(res_data, 'Inbound Scan
Date (Linehaul)', 'Inbound Scan Date')
            res_data = change_title_name(res_data, 'Pickup remark',
'Pickup Remark')
            res_data = change_title_name(res_data, 'Drop off date',
'Dropoff Date')
            res_data = change_title_name(res_data,
re.search(r'\'Drop off [Tt]ime\'',

str(res_data.columns)).group(0)[1:-1], "Dropoff Time")
            res_data = change_title_name(res_data, 'Drop off
status', 'Dropoff Status')
            res_data = change_title_name(res_data, 'Drop off
remark', 'Dropoff Remark')
            res_data = change_title_name(res_data, 'Requested
Amount', 'Requested Credit Amount')

            # 解决 Reason for complaint 问题
            if 'Issue' in same.columns:
                res_data['Reason for Complaint'] = same['Issue']

            # 5. 遍历 same 的标题，将 same 的数据写入 same 和
res_data 共有的标题下
            # print('\nres', res_data.columns)
            for title in same.columns:
                if title in res_data.columns:
                    res_data[title] = same[title]
                    # print(title + ' success to write in')

            # 6. 将 res_data 的标题重置为 ending 的标题
            columns_list = list(big_sheet.columns)
            columns_list.append('delivery_date')
            columns_list.append('Earliest Dropoff Time')
            columns_list.append('Latest Dropoff Time')
            columns_list.append('Earliest Dropoff Date')
            columns_list.append('Latest Dropoff Date')
```

```
        res_data.columns = columns_list

        return res_data


def change_title_name(pd, pd_title, pd_title_change):
    # print(pd_title, 'change to', pd_title_change)
    df = pd.rename(columns={pd_title: pd_title_change})
    return df

```