analyser.py

```
import json
import os
import shutil
import uuid

import pandas as pd
import utils.analyser_utils as analyser_utils
import requests
import cv2.cv2 as cv2
import datetime

from const import USER_NAME, USER_PSW
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/B28g5Mkirtzwl jFDty7THWI75r/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):
        # 普通分析
        if self.day == '4':
            param_dict = {'Reason for Complaint': StringVar(), 'Details of Complaint': StringVar(),
                          'Tracking Code': StringVar(), 'Drop off status': StringVar(),
                          'Earliest Dropoff Time': StringVar(), 'Latest Dropoff Time': StringVar(),
                          'Scheduled Delivery Date': StringVar(), 'Shipment status': StringVar(),
```

```
                            'Inbound Scan Date 减 Scheduled Delivery Date': StringVar(),
                            'Inbound Scan Date (Linehaul)': StringVar(), 'Inbound Scan Time': StringVar(),
                            'Inbound status': StringVar(), 'Pickup Date 减 Scheduled Delivery Date':
StringVar(),
                            'Pickup Date': StringVar(), 'Pickup Time': StringVar(), 'Pickup Status':
StringVar(),
                            'Drop off date 减 Pickup Date': StringVar(), 'Drop off date': StringVar(),
                            'Drop off time': StringVar(), 'Drop off remark': StringVar()}
                return param_dict
            # 智能分析
            elif self.day == '3':
                param_dict = {'Tracking Code': StringVar(), 'Drop off status': StringVar(),
                            'Earliest Dropoff Time': StringVar(), 'Latest Dropoff Time': StringVar(),
                            'Scheduled Delivery Date': StringVar(), 'Shipment status': StringVar(),
                            'Inbound Scan Date 减 Scheduled Delivery Date': StringVar(),
                            'Inbound Scan Date (Linehaul)': StringVar(), 'Inbound Scan Time': StringVar(),
                            'Inbound status': StringVar(), 'Pickup Date 减 Scheduled Delivery Date':
StringVar(),
                            'Pickup Date': StringVar(), 'Pickup Time': StringVar(), 'Pickup Status':
StringVar(),
                            'Drop off date 减 Pickup Date': StringVar(), 'Drop off date': StringVar(),
                            'Drop off Time': StringVar(), 'Drop off remark': StringVar()}
                return param_dict

        @staticmethod
        def initialize_df(first_df):
            """
            取一切需要打标签的内容
            包含: drop off status == 'Success' 的情况
            Issue Category 为空的情况 （没填上）
            """
            first_df['Inbound Scan Date 减 Scheduled Delivery Date'] = None
            first_df['Pickup Date 减 Scheduled Delivery Date'] = None
            first_df['Pickup Date 减 Drop off date'] = None
            for index, row in first_df.iterrows():
                first_df.loc[index, 'Inbound Scan Date 减 Scheduled Delivery Date'] \
                    = analyser_utils.date_subtract(first_df.loc[index, 'Inbound Scan Date (Linehaul)'],
                                                   first_df.loc[index, 'Scheduled Delivery Date'])
                first_df.loc[index, 'Pickup Date 减 Scheduled Delivery Date'] \
                    = analyser_utils.date_subtract(first_df.loc[index, 'Pickup Date'],
                                                   first_df.loc[index, 'Scheduled Delivery Date'])
                first_df.loc[index, 'Drop off date 减 Pickup Date'] \
                    = analyser_utils.date_subtract(first_df.loc[index, 'Drop off date'],
                                                   first_df.loc[index, 'Pickup Date'])
```

```
81            first_df_up = first_df[first_df['Drop off status'] == 'SUCCEEDED']
82            first_df_down = first_df[pd.isna(first_df['Issue Category'])]
83            res_df = pd.concat([first_df_up, first_df_down]).drop_duplicates().reset_index(drop=True)
84            return res_df
85
86        def run(self):
87            """
88            run 的逻辑
89            再此创建一个新的 tkinter 界面，并提供两个按钮，上一页，下一页
90            上一页 依然运行 run 函数，只不过 self.index + 1
91            """
92            # 一堆逻辑 列出当前 index 的 dataframe，普通分析
93            # 普通分析
94            if self.day == '4':
95                self.window = Toplevel(master=self.root)
96                self.param_dict = self.initialize_param_dict()
97                self.temp_window = ttk.Treeview(self.window, show='headings')
98                # 加入各种列
99                self.temp_window['columns'] = ('Reason for Complaint', 'Details of Complaint', 'Tracking
Code',
100                                               'Drop off status', 'Earliest dropoff time', 'Latest dropoff
time',
101                                               'Scheduled Date', 'Shipment status', 'Inbound 减 Scheduled',
'Inbound Date',
102                                               'Inbound scan time', 'Inbound status', 'Pickup 减 Scheduled',
'Pickup Date',
103                                               'Pickup Time', 'Pickup status',
104                                               'Drop off 减 Pickup', 'Drop off date', 'Drop off time', 'Drop
off remark')
105
106                self.temp_window.column('Reason for Complaint', width=65)
107                self.temp_window.column('Details of Complaint', width=65)
108                self.temp_window.column('Tracking Code', width=100)
109                self.temp_window.column('Drop off status', width=100)
110                self.temp_window.column('Earliest dropoff time', width=50)  #
111                self.temp_window.column('Latest dropoff time', width=50)  #
112                self.temp_window.column('Scheduled Date', width=90)
113                self.temp_window.column('Shipment status', width=80)  #
114                self.temp_window.column('Inbound 减 Scheduled', width=40)
115                self.temp_window.column('Inbound Date', width=80)
116                self.temp_window.column('Inbound scan time', width=50)  #
117                self.temp_window.column('Inbound status', width=80)  #
118                self.temp_window.column('Pickup 减 Scheduled', width=40)
119                self.temp_window.column('Pickup Date', width=90)
```

```
120             self.temp_window.column('Pickup Time', width=65)
121             self.temp_window.column('Pickup status', width=65)  #
122             self.temp_window.column('Drop off 减 Pickup', width=40)
123             self.temp_window.column('Drop off date', width=65)
124             self.temp_window.column('Drop off time', width=65)
125             self.temp_window.column('Drop off remark', width=120)
126
127             self.temp_window.heading('Reason for Complaint', text='Reason for Complaint')
128             self.temp_window.heading('Details of Complaint', text='Details of Complaint')
129             self.temp_window.heading('Tracking Code', text='Tracking Code')
130             self.temp_window.heading('Drop off status', text='Drop off status')
131             self.temp_window.heading('Earliest dropoff time', text='Earliest dropoff time')
132             self.temp_window.heading('Latest dropoff time', text='Latest dropoff time')
133             self.temp_window.heading('Scheduled Date', text='Scheduled Date')
134             self.temp_window.heading('Shipment status', text='Shipment status')
135             self.temp_window.heading('Inbound 减 Scheduled', text='Inbound 减 Scheduled')
136             self.temp_window.heading('Inbound Date', text='Inbound Date')
137             self.temp_window.heading('Inbound scan time', text='Inbound scan time')
138             self.temp_window.heading('Inbound status', text='Inbound status')
139             self.temp_window.heading('Pickup 减 Scheduled', text='Pickup 减 Scheduled')
140             self.temp_window.heading('Pickup Date', text='Pickup Date')
141             self.temp_window.heading('Pickup Time', text='Pickup Time')
142             self.temp_window.heading('Pickup status', text='Pickup status')
143             self.temp_window.heading('Drop off 减 Pickup', text='Drop off 减 Pickup')
144             self.temp_window.heading('Drop off date', text='Drop off date')
145             self.temp_window.heading('Drop off time', text='Drop off time')
146             self.temp_window.heading('Drop off remark', text='Drop off remark')
147
148             # 初次设置值
149             self.change_data(data_index=0)
150             self.temp_window.pack(pady=20)
151
152             # button_next 的函数为 next_page
153             prev_button = Button(self.window, text='上一页', command=self.prev_page)
154             prev_button.place(x=100, y=100)
155
156             next_button = Button(self.window, text='下一页', command=self.next_page)
157             next_button.place(x=300, y=100)
158
159             confirm_button = Button(self.window, text='确定', command=self.confirm)
160             confirm_button.place(x=900, y=100)
161
162             Button(self.window, text='清除缓存', command=self.clear_cache).place(x=1200, y=100)
163             Button(self.window, text='显示照片', command=self.show_pic).place(x=1100, y=100)
```

```
164             Button(self.window, text='提交', command=self.hand_in_result).place(x=1300, y=100)
165             Button(self.window, text='打开字典', command=self.open_dictionary).place(x=1300, y=200)
166
167             # 显示进度
168             self.process = StringVar()
169             Entry(self.window, width='10', textvariable=self.process).place(x=100, y=300)
170             self.process.set(str(self.index) + '/' + str(self.mission_length))
171
172             # 绑定按键
173             self.window.bind('<Down>', self.next_page)
174             self.window.bind('<Up>', self.prev_page)
175             self.window.bind('<Return>', self.confirm)
176             self.window.bind('<s>', self.show_pic)
177
178             # 设置一个框，用于填对应的序号
179             self.answer = StringVar()
180             Label(self.window, text="此条记录的问题，对应的 JJ 序号:").place(x=500, y=100)
181             Entry(self.window, width='5', textvariable=self.answer).place(x=720, y=100)
182
183             # 显示 tracking code
184             self.tracking_code = StringVar()
185             Label(self.window, text="Tracking code:").place(x=600, y=200)
186             Entry(self.window, width='20', textvariable=self.tracking_code).place(x=700, y=200)
187             self.tracking_code.set(self.data_frame.loc[self.index, 'Tracking Code'])
188
189             # 显示 顾客的 notes
190             self.client_comment = StringVar()
191             Label(self.window, text="note:").place(x=100, y=150)
192             Entry(self.window, width='100', textvariable=self.client_comment).place(x=150, y=150)
193             result_dict = self.get_dict_from_tracking_code(
194                 tracking_code=self.data_frame.loc[self.index, 'Tracking Code']
195             )
196             if 'dropoff_note' in result_dict['results'][0]['shipment'].keys():
197                 self.client_comment.set(result_dict['shipment']['dropoff_note'])
198             else:
199                 self.client_comment.set('')
200
201             # 显示 customer id
202             self.customer_id = StringVar()
203             Label(self.window, text="note:").place(x=800, y=150)
204             Entry(self.window, width='100', textvariable=self.customer_id).place(x=900, y=150)
205             if 'customer' in result_dict['results'][0]['shipment'].keys():
206                 self.customer_id.set(result_dict['shipment']['customer']['phone_number'])
207             else:
```

```
208                 self.customer_id.set('')
209
210             # 一堆逻辑 显示出图片和详细地址文字
211             self.window.mainloop()
212
213         # 智能分析
214         elif self.day == '3':
215             self.window = Toplevel(master=self.root)
216             self.param_dict = self.initialize_param_dict()
217             self.temp_window = ttk.Treeview(self.window, show='headings')
218             # 加入各种列
219             self.temp_window['columns'] = ('Tracking Code',
220                                            'Drop off status',
221                                            'Scheduled Date', 'Earliest Dropoff Time', 'Latest Dropoff
Time',
222                                            'Shipment status', 'Inbound 减 Scheduled', 'Inbound Date',
223                                            'Inbound scan time', 'Inbound status', 'Pickup 减 Scheduled',
'Pickup Date',
224                                            'Pickup Time', 'Pickup status',
225                                            'Drop off 减 Pickup', 'Drop off date', 'Drop off Time', 'Drop
off remark')
226
227             self.temp_window.column('Tracking Code', width=100)
228             self.temp_window.column('Drop off status', width=100)
229             self.temp_window.column('Scheduled Date', width=120)
230             self.temp_window.column('Earliest Dropoff Time', width=50)  #
231             self.temp_window.column('Latest Dropoff Time', width=50)  #
232             self.temp_window.column('Shipment status', width=120)  #
233             self.temp_window.column('Inbound 减 Scheduled', width=40)
234             self.temp_window.column('Inbound Date', width=120)
235             self.temp_window.column('Inbound scan time', width=80)  #
236             self.temp_window.column('Inbound status', width=80)  #
237             self.temp_window.column('Pickup 减 Scheduled', width=40)
238             self.temp_window.column('Pickup Date', width=50)
239             self.temp_window.column('Pickup Time', width=50)
240             self.temp_window.column('Pickup status', width=120)  #
241             self.temp_window.column('Drop off 减 Pickup', width=40)
242             self.temp_window.column('Drop off date', width=100)
243             self.temp_window.column('Drop off Time', width=50)
244             self.temp_window.column('Drop off remark', width=120)
245
246             self.temp_window.heading('Tracking Code', text='Tracking Code')
247             self.temp_window.heading('Drop off status', text='Drop off status')
248             self.temp_window.heading('Scheduled Date', text='Scheduled Date')
```

```
249             self.temp_window.heading('Earliest Dropoff Time', text='Earliest dropoff Time')
250             self.temp_window.heading('Latest Dropoff Time', text='Latest dropoff Time')
251             self.temp_window.heading('Shipment status', text='Shipment status')
252             self.temp_window.heading('Inbound 减 Scheduled', text='Inbound 减 Scheduled')
253             self.temp_window.heading('Inbound Date', text='Inbound Date')
254             self.temp_window.heading('Inbound scan time', text='Inbound scan time')
255             self.temp_window.heading('Inbound status', text='Inbound status')
256             self.temp_window.heading('Pickup 减 Scheduled', text='Pickup 减 Scheduled')
257             self.temp_window.heading('Pickup Date', text='Pickup Date')
258             self.temp_window.heading('Pickup Time', text='Pickup Time')
259             self.temp_window.heading('Pickup status', text='Pickup status')
260             self.temp_window.heading('Drop off 减 Pickup', text='Drop off 减 Pickup')
261             self.temp_window.heading('Drop off date', text='Drop off date')
262             self.temp_window.heading('Drop off Time', text='Drop off Time')
263             self.temp_window.heading('Drop off remark', text='Drop off remark')
264
265             # 初次设置值
266             self.change_data(data_index=0)
267             self.temp_window.pack(pady=20)
268
269             # button_next 的函数为 next_page
270             prev_button = Button(self.window, text='上一页', command=self.prev_page)
271             prev_button.place(x=100, y=100)
272
273             next_button = Button(self.window, text='下一页', command=self.next_page)
274             next_button.place(x=300, y=100)
275
276             confirm_button = Button(self.window, text='确定', command=self.confirm)
277             confirm_button.place(x=900, y=100)
278             Button(self.window, text='清除缓存', command=self.clear_cache).place(x=1200, y=100)
279             Button(self.window, text='显示照片', command=self.show_pic).place(x=1100, y=100)
280             Button(self.window, text='提交', command=self.hand_in_result).place(x=1300, y=100)
281             Button(self.window, text='打开字典', command=self.open_dictionary).place(x=1300, y=200)
282
283             # 显示进度
284             self.process = StringVar()
285             Entry(self.window, width='10', textvariable=self.process).place(x=100, y=300)
286             self.process.set(str(self.index) + '/' + str(self.mission_length))
287
288             # 绑定按键
289             self.window.bind('<Down>', self.next_page)
290             self.window.bind('<Up>', self.prev_page)
291             self.window.bind('<Return>', self.confirm)
292             self.window.bind('<s>', self.show_pic)
```

```
293
294             # 设置一个框，用于填对应的序号
295             self.answer = StringVar()
296             Label(self.window, text="此条记录的问题，对应的 JJ 序号:").place(x=500, y=100)
297             entry = Entry(self.window, width='5', textvariable=self.answer).place(x=720, y=100)
298
299             # 显示 tracking code
300             self.tracking_code = StringVar()
301             Label(self.window, text="Tracking Code:").place(x=600, y=200)
302             Entry(self.window, width='20', textvariable=self.tracking_code).place(x=700, y=200)
303             self.tracking_code.set(self.data_frame.loc[self.index, 'Tracking Code'])
304
305             # 显示 顾客的 notes
306             self.client_comment = StringVar()
307             Label(self.window, text="note:").place(x=100, y=150)
308             Entry(self.window, width='100', textvariable=self.client_comment).place(x=150, y=150)
309             result_dict = self.get_dict_from_tracking_code(
310                 tracking_code=self.data_frame.loc[self.index, 'Tracking Code']
311             )
312             if 'dropoff_note' in result_dict['results'][0]['shipment'].keys():
313                 self.client_comment.set(result_dict['results'][0]['shipment']['dropoff_note'])
314             else:
315                 self.client_comment.set('')
316
317             # 显示 customer id
318             self.customer_id = StringVar()
319             Label(self.window, text="note:").place(x=800, y=150)
320             Entry(self.window, width='100', textvariable=self.customer_id).place(x=900, y=150)
321             if 'customer' in result_dict['results'][0]['shipment'].keys():
322                 self.customer_id.set(result_dict['shipment']['customer']['phone_number'])
323             else:
324                 self.customer_id.set('')
325
326             # 进度条
327             self.process.set(str(self.index) + '/' + str(self.mission_length))
328
329             # 一堆逻辑 显示出图片和详细地址文字
330             self.window.mainloop()
331
332     def next_page(self, event=None):
333         self.index = self.index + 1
334
335         if self.index >= len(self.data_frame['Tracking Code']):
336             # 到达最底下了
```

```
337             messagebox.showinfo(title='警告', message='没有下一页了')
338             self.window.focus_force()
339             self.index = self.index - 1
340             self.change_data(self.index)
341 
342         self.change_data(self.index)
343 
344         # tracing code
345         self.tracking_code.set(self.data_frame.loc[self.index, 'Tracking Code'])
346 
347         # dropoff note
348         result_dict = self.get_dict_from_tracking_code(
349             tracking_code=self.data_frame.loc[self.index, 'Tracking Code']
350         )
351         if 'dropoff_note' in result_dict['results'][0]['shipment'].keys():
352             self.client_comment.set(result_dict['results'][0]['shipment']['dropoff_note'])
353         else:
354             self.client_comment.set('')
355 
356         # customer_id
357         self.customer_id = StringVar()
358         if 'customer' in result_dict['results'][0]['shipment'].keys():
359             self.customer_id.set(result_dict['shipment']['customer']['phone_number'])
360         else:
361             self.customer_id.set('')
362 
363         # 进度条
364         self.process.set(str(self.index) + '/' + str(self.mission_length))
365 
366         self.temp_window.delete(f'item{self.index - 1}')
367 
368     def prev_page(self, event=None):
369         self.index = self.index - 1
370         if self.index < 0:
371             # 到达最开始了
372             messagebox.showinfo(title='警告', message='没有上一页了')
373             self.window.focus_force()
374             self.index = self.index + 1
375             self.change_data(self.index)
376 
377         self.change_data(self.index)
378 
379         # tracing code
380         self.tracking_code.set(self.data_frame.loc[self.index, 'Tracking Code'])
```

```
381
382             # dropoff note
383             result_dict = self.get_dict_from_tracking_code(
384                 tracking_code=self.data_frame.loc[self.index, 'Tracking Code']
385             )
386             if 'dropoff_note' in result_dict['results'][0]['shipment'].keys():
387                 self.client_comment.set(result_dict['results'][0]['shipment']['dropoff_note'])
388             else:
389                 self.client_comment.set('')
390
391             # customer_id
392             self.customer_id = StringVar()
393             if 'customer' in result_dict['results'][0]['shipment'].keys():
394                 self.customer_id.set(result_dict['shipment']['customer']['phone_number'])
395             else:
396                 self.customer_id.set('')
397
398             self.temp_window.delete(f'item{self.index + 1}')
399
400         def change_data(self, data_index):
401             # 设置 StringVar
402             if self.day == '4':
403                 self.param_dict['Reason for Complaint'].set(self.data_frame.loc[data_index, 'Reason for
Complaint'])
404                 self.param_dict['Details of Complaint'].set(self.data_frame.loc[data_index, 'Details of
Complaint'])
405                 self.param_dict['Tracking Code'].set(self.data_frame.loc[data_index, 'Tracking Code'])
406                 self.param_dict['Drop off status'].set(self.data_frame.loc[data_index, 'Drop off status'])
407                 self.param_dict['Earliest Dropoff Time'].set(self.data_frame.loc[data_index, 'Earliest
Dropoff Time'])
408                 self.param_dict['Latest Dropoff Time'].set(self.data_frame.loc[data_index, 'Latest Dropoff
Time'])
409                 self.param_dict['Scheduled Delivery Date'].set(self.data_frame.loc[data_index, 'Scheduled
Delivery Date'])
410                 self.param_dict['Shipment status'].set(self.data_frame.loc[data_index, 'Shipment status'])
411                 self.param_dict['Inbound Scan Date 减 Scheduled Delivery Date'].set(
412                     self.data_frame.loc[data_index, 'Inbound Scan Date 减 Scheduled Delivery Date'])
413                 self.param_dict['Inbound Scan Date (Linehaul)'].set(
414                     self.data_frame.loc[data_index, 'Inbound Scan Date (Linehaul)'])
415                 self.param_dict['Inbound Scan Time'].set(self.data_frame.loc[data_index, 'Inbound Scan
Time'])
416                 self.param_dict['Inbound status'].set(self.data_frame.loc[data_index, 'Inbound status'])
417                 self.param_dict['Pickup Date 减 Scheduled Delivery Date'].set(
418                     self.data_frame.loc[data_index, 'Pickup Date 减 Scheduled Delivery Date'])
```

```
419             self.param_dict['Pickup Date'].set(self.data_frame.loc[data_index, 'Pickup Date'])
420             self.param_dict['Pickup Time'].set(self.data_frame.loc[data_index, 'Pickup Time'])
421             self.param_dict['Pickup Status'].set(self.data_frame.loc[data_index, 'Pickup Status'])
422             self.param_dict['Drop off date 减 Pickup Date'].set(
423                 self.data_frame.loc[data_index, 'Drop off date 减 Pickup Date'])
424             self.param_dict['Drop off date'].set(self.data_frame.loc[data_index, 'Drop off date'])
425             self.param_dict['Drop off time'].set(self.data_frame.loc[data_index, 'Drop off time'])
426             self.param_dict['Drop off remark'].set(self.data_frame.loc[data_index, 'Drop off remark'])
427
428             self.temp_window.insert('', 0, f'item{self.index}', values=(
429                 self.param_dict['Reason for Complaint'].get(),
430                 self.param_dict['Details of Complaint'].get(),
431                 self.param_dict['Tracking Code'].get(),
432                 self.param_dict['Drop off status'].get(),
433                 self.param_dict['Earliest Dropoff Time'].get(),
434                 self.param_dict['Latest Dropoff Time'].get(),
435                 self.param_dict['Scheduled Delivery Date'].get(),
436                 self.param_dict['Shipment status'].get(),
437                 self.param_dict['Inbound Scan Date 减 Scheduled Delivery Date'].get(),
438                 self.param_dict['Inbound Scan Date (Linehaul)'].get(),
439                 self.param_dict['Inbound Scan Time'].get(),
440                 self.param_dict['Inbound status'].get(),
441                 self.param_dict['Pickup Date 减 Scheduled Delivery Date'].get(),
442                 self.param_dict['Pickup Date'].get(),
443                 self.param_dict['Pickup Time'].get(),
444                 self.param_dict['Pickup Status'].get(),
445                 self.param_dict['Drop off date 减 Pickup Date'].get(),
446                 self.param_dict['Drop off date'].get(),
447                 self.param_dict['Drop off time'].get(),
448                 self.param_dict['Drop off remark'].get()
449             ))
450             self.temp_window.pack(pady=20)
451             print(self.param_dict['Tracking Code'].get())
452
453         elif self.day == '3':
454             self.param_dict['Tracking Code'].set(self.data_frame.loc[data_index, 'Tracking Code'])
455             self.param_dict['Drop off status'].set(self.data_frame.loc[data_index, 'Drop off status'])
456             self.param_dict['Scheduled Delivery Date'].set(self.data_frame.loc[data_index, 'Scheduled
Delivery Date'])
457             self.param_dict['Earliest Dropoff Time'].set(self.data_frame.loc[data_index, 'Earliest
Dropoff Time'])
458             self.param_dict['Latest Dropoff Time'].set(self.data_frame.loc[data_index, 'Latest Dropoff
Time'])
459             self.param_dict['Shipment status'].set(self.data_frame.loc[data_index, 'Shipment status'])
```

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

```
503             header = {
504                 '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',
505                 'content-type': 'application/json',
506                 'cookie': r'fp=1a39e1225ea764ca9f2abf599fafba34;
xtoken="dE9DbW1wYkZDI/B28g5MkirtzwljFDty7THWI75r/mVq4do8YKOJBeUtONSQ1d3L1Yb5JCAEZPTk\012FFj7LXpbKjSaV71j1S6I9z
jtTLurIi1ddgqe+xsIRU84cjgOSktu\012"'}
507             # 生成 post 的 json_data
508             data_dict = {'size': 15, 'q': tracking_code,
509                          'filters': {}, 'sorts': ['-dropoff_earliest_ts']}
510             json_data = json.dumps(data_dict)
511
512             session = requests.Session()
513             user = USER_NAME
514             password = USER_PSW
515             response = session.post(url=url, headers=header, data=json_data, auth=HttpNtlmAuth(user,
password))
516
517             result_dict = json.loads(response.text)
518             return result_dict
519
520         def show_pic(self, event=None):
521             # 生成 dict_data
522             result_dict = self.get_dict_from_tracking_code(
523                 tracking_code=self.data_frame.loc[self.index, 'Tracking Code']
524             )
525
526             session = requests.Session()
527
528             # 如果存在照片，就显示
529             if result_dict['results'][0]['pod']['images'] != []:
530                 # 如果是单张照片
531                 if len(result_dict['results'][0]['pod']['images']) == 1:
532                     img_url = result_dict['results'][0]['pod']['images'][0]['url']
533                     img_url_response = session.get(img_url)
534
535                     # 写入文件到 cache
536                     with open(f'tools/img_cache/{img_url[-10:]}.png', 'wb') as fp:
537                         fp.write(img_url_response.content)
538
539                     if 'street2' in result_dict['results'][0]['shipment']['dropoff_address'].keys():
540                         address_street2 = result_dict['results'][0]['shipment']['dropoff_address']['street2']
+ ' '
541                     else:
```

```
542                     address_street2 = "" + ' '
543                 address_street = result_dict['results'][0]['shipment']['dropoff_address']['street'] + ' '
544                 address_city = result_dict['results'][0]['shipment']['dropoff_address']['city'] + ' '
545                 address_state = result_dict['results'][0]['shipment']['dropoff_address']['state'] + ' '
546                 address_zipcode = result_dict['results'][0]['shipment']['dropoff_address']['zipcode'] + '
,
547                 address = address_street2 + address_street + address_city + address_state +
address_zipcode
548
549                 img = cv2.imread(f'tools/img_cache/{img_url[-10:]}.png')
550                 img = process_image(img)
551                 cv2.imshow(f"address: {address}", img)
552                 cv2.waitKey()
553             # 多张照片
554             if len(result_dict['results'][0]['pod']['images']) > 1:
555                 imgs = []
556                 for img_url in result_dict['results'][0]['pod']['images']:
557                     img_url_response = session.get(img_url['url'])
558
559                     # 写入文件到 cache
560                     with open(f'tools/img_cache/{img_url["url"][-10:]}.png', 'wb') as fp:
561                         fp.write(img_url_response.content)
562                         imgs.append(f'tools/img_cache/{img_url["url"][-10:]}.png')
563
564                 if 'street2' in result_dict['results'][0]['shipment']['dropoff_address'].keys():
565                     address_street2 = result_dict['results'][0]['shipment']['dropoff_address']['street2']
+ ' '
566                 else:
567                     address_street2 = "" + ' '
568
569                 address_street = result_dict['results'][0]['shipment']['dropoff_address']['street'] + ' '
570                 address_city = result_dict['results'][0]['shipment']['dropoff_address']['city'] + ' '
571                 address_state = result_dict['results'][0]['shipment']['dropoff_address']['state'] + ' '
572                 address_zipcode = result_dict['results'][0]['shipment']['dropoff_address']['zipcode'] + '
,
573                 address = address_street2 + address_street + address_city + address_state +
address_zipcode
574
575                 # 依次显示照片
576                 for img in imgs:
577                     img = cv2.imread(img)
578                     img = process_image(img)
579                     cv2.imshow(f"address: {address} have more than one pic", img)
580                     cv2.waitKey()
```

```
581         else:
582             if 'street2' in result_dict['results'][0]['shipment']['dropoff_address'].keys():
583                 address_street2 = result_dict['results'][0]['shipment']['dropoff_address']['street2'] + '
,
584             else:
585                 address_street2 = " "
586             address_street = result_dict['results'][0]['shipment']['dropoff_address']['street'] + ' '
587             address_city = result_dict['results'][0]['shipment']['dropoff_address']['city'] + ' '
588             address_state = result_dict['results'][0]['shipment']['dropoff_address']['state'] + ' '
589             address_zipcode = result_dict['results'][0]['shipment']['dropoff_address']['zipcode'] + ' '
590             address = address_street2 + address_street + address_city + address_state + address_zipcode
591
592             messagebox.showinfo('没有照片', message=f'地址为: {address}')
593             self.window.focus_force()
594
595     def clear_cache(self):
596         if not os.path.exists('tools/img_cache'):
597             os.mkdir('tools/img_cache')
598         del_list = os.listdir('tools/img_cache')
599         if len(del_list) == 0:
600             messagebox.showinfo(title='清除失败', message='无缓存')
601             return
602         file_size_sum = 0
603         for f in del_list:
604             file_path = os.path.join('tools/img_cache', f)
605             if os.path.isfile(file_path):
606                 file_size_sum += self.get_filesize(file_path)
607                 os.remove(file_path)
608             elif os.path.isdir(file_path):
609                 shutil.rmtree(file_path)
610         messagebox.showinfo(title='清除成功', message=f'清除缓存共 {round(file_size_sum, 1)}mb')
611
612     @staticmethod
613     def get_filesize(file_path):
614         file_size = os.path.getsize(file_path)
615         file_size = file_size / float(1024 * 1024)
616         return round(file_size, 2)
617
618     def confirm(self, event=None):
619         answer_index = self.answer.get()
620         analyser_utils.copy_reason(
621             data_frame_row=self.data_frame.iloc[self.index: self.index + 1, :],
622             index=int(answer_index)
623         )
```

```
624             messagebox.showinfo(title='确定', message='您的输入已写入')
625             self.window.focus_force()
626 
627         def hand_in_result(self):
628             # 生成 csv
629             date_time = datetime.datetime.now().strftime('%Y-%m-%d %H-%M-%S')
630             path = str(self.save_folder_path) + '/最终版' + date_time + '.csv'
631 
632             res_df = self.write_in()
633 
634             # 智能分析的需要改动列名
635             if self.day == '3':
636                 res_df.rename(columns={'AH Assessment': 'HF Reason Code'}, inplace=True)
637             # 这里 res_df 中的五列将带 x 的写回去，并 drop 掉
638             res_df.to_csv(path, index=False)
639 
640             messagebox.showinfo(title='成功', message=f'已生成 {path}')
641 
642         def write_in(self):
643             def get_index(tracking_code, source_df):
644                 return source_df[source_df['Tracking Code'] == tracking_code].index
645 
646             data_frame = self.data_frame.copy()
647             source_df = self.source_df.copy()
648             # 把 data_frame 根据相同的 tracking code 将五列写入 source_df 返回 res_df
649             for index, row in data_frame.iterrows():
650                 source_df.loc[get_index(data_frame.loc[index, 'Tracking Code'], source_df),
651                               'Issue Category'] = data_frame.loc[index, 'Issue Category']
652                 source_df.loc[get_index(data_frame.loc[index, 'Tracking Code'], source_df),
653                               'Delivery Comments'] = data_frame.loc[index, 'Delivery Comments']
654                 source_df.loc[get_index(data_frame.loc[index, 'Tracking Code'], source_df),
655                               'AH Assessment'] = data_frame.loc[index, 'AH Assessment']
656                 source_df.loc[get_index(data_frame.loc[index, 'Tracking Code'], source_df),
657                               'POD Quality'] = data_frame.loc[index, 'POD Quality']
658                 source_df.loc[get_index(data_frame.loc[index, 'Tracking Code'], source_df),
659                               'POD Valid?'] = data_frame.loc[index, 'POD Valid?']
660             return source_df
661 
662         @staticmethod
663         def open_dictionary():
664             os.system(os.getcwd() + '/tools/files/dictionary.xlsx')
665 
666 
667 def process_image(img):
```

```
668     """
669     此处使用改进的 REAL-ESRGAN 算法，增强两倍画质，用于查看高清大图
670     :param img:
671     :return:
672     """
673     min_side = 768
674     size = img.shape
675     h, w = size[0], size[1]
676     # 长边缩放为 min_side
677     scale = max(w, h) / float(min_side)
678     new_w, new_h = int(w / scale), int(h / scale)
679     resize_img = cv2.resize(img, (new_w, new_h))
680     # 填充至 min_side * min_side
681     if new_w % 2 != 0 and new_h % 2 == 0:
682         top, bottom, left, right = (min_side - new_h) / 2, (min_side - new_h) / 2, (min_side - new_w) / 2
+ 1, (
683                     min_side - new_w) / 2
684     elif new_h % 2 != 0 and new_w % 2 == 0:
685         top, bottom, left, right = (min_side - new_h) / 2 + 1, (min_side - new_h) / 2, (min_side - new_w)
/ 2, (
686                     min_side - new_w) / 2
687     elif new_h % 2 == 0 and new_w % 2 == 0:
688         top, bottom, left, right = (min_side - new_h) / 2, (min_side - new_h) / 2, (min_side - new_w) /
2, (
689                     min_side - new_w) / 2
690     else:
691         top, bottom, left, right = (min_side - new_h) / 2 + 1, (min_side - new_h) / 2, (min_side - new_w)
/ 2 + 1, (
692                     min_side - new_w) / 2
693     pad_img = cv2.copyMakeBorder(resize_img, int(top), int(bottom), int(left), int(right),
cv2.BORDER_CONSTANT,
694                                  value=[0, 0, 0])  # 从图像边界向上,下,左,右扩的像素数目
695     u_id = uuid.uuid4()
696
697     try:
698         cv2.imwrite(f'cache_imgs/{u_id}.png', pad_img)
699     except:
700         raise "图片写入发生错误"
701
702     out_path = os.path.join('reuslt', str(u_id) + '_out.png')
703     run_cmd(
704         cmd=f'python inference_realesrgan.py -n RealESRGAN_x4plus -i cache_imgs/'
705     )
706
```

```
    img = cv2.imread(out_path)
    min_side = 768
    size = img.shape
    h, w = size[0], size[1]
    # 长边缩放为 min_side
    scale = max(w, h) / float(min_side)
    new_w, new_h = int(w / scale), int(h / scale)
    resize_img = cv2.resize(img, (new_w, new_h))
    # 填充至 min_side * min_side
    if new_w % 2 != 0 and new_h % 2 == 0:
        top, bottom, left, right = (min_side - new_h) / 2, (min_side - new_h) / 2, (min_side - new_w) / 2
+ 1, (
                min_side - new_w) / 2
    elif new_h % 2 != 0 and new_w % 2 == 0:
        top, bottom, left, right = (min_side - new_h) / 2 + 1, (min_side - new_h) / 2, (min_side - new_w)
/ 2, (
                min_side - new_w) / 2
    elif new_h % 2 == 0 and new_w % 2 == 0:
        top, bottom, left, right = (min_side - new_h) / 2, (min_side - new_h) / 2, (min_side - new_w) /
2, (
                min_side - new_w) / 2
    else:
        top, bottom, left, right = (min_side - new_h) / 2 + 1, (min_side - new_h) / 2, (min_side - new_w)
/ 2 + 1, (
                min_side - new_w) / 2
    upscale_img = cv2.copyMakeBorder(resize_img, int(top), int(bottom), int(left), int(right),
cv2.BORDER_CONSTANT,
                                     value=[0, 0, 0])
    return upscale_img


def run_cmd(cmd):
    """
    运行命令并返回返回值
    :param cmd: 命令
    :return: 命令输出
    """
    var = os.popen(cmd)
    result = var.read()
    var.close()
    return result


class Thursday(object):
```

```
746     def __init__(self, init_df, policy):
747         self.init_df = init_df
748         self.policy = policy
749 
750     def analyse(self):
751         res_data = self.init_df.copy()
752         result = pd.DataFrame(columns=res_data.columns)
753         for index, row in self.init_df.iterrows():
754             # 修改 Scheduled Delivery Date 成为 %Y-%m-%d
755             temp = analyser_utils.change_Scheduled_Delivery_Date(res_data.iloc[index: index + 1, :])
756             # 填入 week ✓
757             temp = analyser_utils.get_week_num(temp)
758             # 修改时区
759             # temp = analyser_utils.data_frame_row_time_change(temp)
760             res_data.iloc[index: index + 1, :] = analyser_utils.get_status(temp, day='4',
policy=self.policy)
761             result = pd.concat([result, temp])
762         result['Updated Reason Code'] = result['AH Assessment']
763         return result
764 
765 
766 class Wednesday(object):
767     def __init__(self, init_df, policy):
768         self.init_df = init_df
769         self.policy = policy
770 
771     def analyse(self):
772         res_data = self.init_df.copy()
773         res_data.rename(columns={'HF Reason Code': 'AH Assessment', 'POD Qaulity': 'POD Quality'},
inplace=True)
774         result = pd.DataFrame(columns=res_data.columns)
775         for index, row in self.init_df.iterrows():
776             # 修改 Scheduled Delivery Date 成为 %Y-%m-%d
777             temp = analyser_utils.change_Scheduled_Delivery_Date(res_data.iloc[index: index + 1, :])
778             # 填入 week ✓
779             temp = analyser_utils.get_week_num(temp)
780             # 分析 status
781             res_data.iloc[index: index + 1, :] = analyser_utils.get_status(temp, day='3',
policy=self.policy)
782             result = pd.concat([result, temp])
783         result['Updated Reason Code'] = result['AH Assessment']
784         return result
785 
```

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


def copy_reason(data_frame_row, index):
    copy_df = DICT_DF[DICT_DF.index == index]
    # 如果全是空的，直接复制
```

```
        if pd.isna(data_frame_row['POD Valid?'].values[0]) and pd.isna(data_frame_row['POD Quality'].values[0]) and \
            pd.isna(data_frame_row['Issue Category'].values[0]) and pd.isna(data_frame_row['Delivery Comments'].values[0]) and \
            pd.isna(data_frame_row['AH Assessment'].values[0]):
            data_frame_row['POD Valid?'] = [nan_to_none(copy_df['POD Valid?'].values[0])]
            data_frame_row['POD Quality'] = [nan_to_none(copy_df['POD Quality'].values[0])]
            data_frame_row['Issue Category'] = copy_df['Issue Category'].values
            data_frame_row['Delivery Comments'] = copy_df['Delivery Comments'].values
            data_frame_row['AH Assessment'] = copy_df['AH Assessment'].values

            return data_frame_row
        # 如果不是空的，加一个 / 再将内容附着上
        else:
            data_frame_row['POD Valid?'] = [nan_to_none(str(copy_df['POD Valid?'].values[0]))]
            data_frame_row['POD Quality'] = [nan_to_none(str(copy_df['POD Quality'].values[0]))]
            data_frame_row['Issue Category'] = [str(data_frame_row['Issue Category'].values[0]) + '/' + str(copy_df['Issue Category'].values[0])]
            data_frame_row['Delivery Comments'] = [str(data_frame_row['Delivery Comments'].values[0]) + '/' + str(copy_df['Delivery Comments'].values[0])]
            data_frame_row['AH Assessment'] = [str(data_frame_row['AH Assessment'].values[0]) + '/' + str(copy_df['AH Assessment'].values[0])]
            return data_frame_row


    def get_pickup_and_delivery_status(data_frame_row, day, policy):
        pickup_diff = date_subtract(data_frame_row['Pickup Date'].values[0],
                                    data_frame_row['Scheduled Delivery Date'].values[0])

        # 如果 pick 当天送达
        if pickup_diff == 0:
            # 如果 pick 当天晚于 12
            if time_upper_than(data_frame_row['Pickup Time'].values[0], '12:00', 0):
                data_frame_row['Pickup Comments'] = ['Pickup after 12pm']

                delivery_diff = date_subtract(data_frame_row['Drop off date'].values[0],
                                              data_frame_row['Pickup Date'].values[0])
                # 判断 delivery 当天
                if delivery_diff == 0:
                    if time_upper_than(data_frame_row['Drop off time'].values[0],
                                       data_frame_row['Latest Dropoff Time'].values[0], policy):
                        data_frame_row = copy_reason(data_frame_row, 118)
                        return data_frame_row
                # 配送晚于 2 天
```

```
            else:
                data_frame_row = copy_reason(data_frame_row, 119)

                if delivery_diff == 1:
                    return data_frame_row
                else:
                    if delivery_diff < 2:
                        return data_frame_row
                    else:

                        data_frame_row = write_in_delivery_comments(data_frame_row,
                                                                    f'pickup ok but delivery late for
{delivery_diff} days')
                        return data_frame_row
        # 如果 pick 早于 12 点
        else:
            delivery_diff = date_subtract(data_frame_row['Drop off date'].values[0],
                                          data_frame_row['Pickup Date'].values[0])
            # 判断 delivery 当天
            if delivery_diff == 0:
                if time_upper_than(data_frame_row['Drop off time'].values[0],
                                   data_frame_row['Latest Dropoff Time'].values[0], policy):
                    data_frame_row = copy_reason(data_frame_row, 118)
                    return data_frame_row
            # 配送晚于 2 天
            else:
                data_frame_row = copy_reason(data_frame_row, 119)

                if delivery_diff == 1:
                    return data_frame_row
                else:
                    if delivery_diff < 2:
                        return data_frame_row
                    else:

                        data_frame_row = write_in_delivery_comments(data_frame_row,
                                                                    f'pickup ok but delivery late for
{delivery_diff} days')
                        return data_frame_row
        return data_frame_row

    # 如果 pick 晚了 n 天
    if pickup_diff > 1:
        # 对于智能分析
```

```
125         if day == '3':
126             data_frame_row = copy_reason(data_frame_row, 41)
127             if pickup_diff == 1:
128 
129                 data_frame_row = write_in_delivery_comments(data_frame_row,
130                                                             f'Inbound ontime but outbound late for 1
day')
131                 data_frame_row['Pickup Comments'] = [
132                     f'Inbound ontime but outbound late for 1 day']
133 
134                 delivery_diff = date_subtract(data_frame_row['Drop off date'].values[0],
135                                               data_frame_row['Pickup Date'].values[0])
136                 # pickup 晚了一天, delivery 当天
137                 if delivery_diff == 0:
138                     # 当天晚了
139                     if time_upper_than(data_frame_row['Drop off time'].values[0],
140                                        data_frame_row['Latest Dropoff Time'].values[0], policy):
141                         data_frame_row = copy_reason(data_frame_row, 118)
142 
143                         data_frame_row = write_in_delivery_comments(data_frame_row,
144                                                                     f'pickup late for {pickup_diff} day
and  delivery late for same day')
145 
146                         return data_frame_row
147                 # pickup 一天,  delivery 多天
148                 else:
149                     data_frame_row = copy_reason(data_frame_row, 119)
150                     if delivery_diff == 1:
151                         data_frame_row = write_in_delivery_comments(data_frame_row,
152                                                                     f'pickup late for 1 day and delivery
late for 1 day')
153                         return data_frame_row
154                     else:
155                         data_frame_row = write_in_delivery_comments(data_frame_row,
156                                                                     f'pickup late for 1 day and delivery
late for {delivery_diff} days')
157                         return data_frame_row
158                 return data_frame_row
159 
160             # pick up 晚于 1 天, 看 delivery
161             elif pickup_diff > 1:
162 
163                 data_frame_row = write_in_delivery_comments(data_frame_row,
```

```
164                                                 f'Inbound ontime but outbound late for
{pickup_diff} days')
165                 data_frame_row['Pickup Comments'] = [
166                     f'Inbound ontime but outbound late for {pickup_diff} days']
167                 return data_frame_row
168             return data_frame_row
169
170         # 普通分析
171         if day == '4':
172             data_frame_row = copy_reason(data_frame_row, 0)
173             if pickup_diff == 1:
174
175                 data_frame_row = write_in_delivery_comments(data_frame_row,
176                                                             f'Inbound ontime but outbound late for 1
day')
177                 data_frame_row['Pickup Comments'] = [
178                     f'Inbound ontime but outbound late for 1 day']
179
180                 delivery_diff = date_subtract(data_frame_row['Drop off date'].values[0],
181                                               data_frame_row['Pickup Date'].values[0])
182                 # pickup 晚了一天, delivery 当天
183                 if delivery_diff == 0:
184                     # 当天晚了
185                     if time_upper_than(data_frame_row['Drop off time'].values[0],
186                                        data_frame_row['Latest Dropoff Time'].values[0], policy):
187                         data_frame_row = copy_reason(data_frame_row, 118)
188
189                         data_frame_row = write_in_delivery_comments(data_frame_row,
190                                                                     f'pickup late for {pickup_diff} day
and  delivery late for same day')
191
192                         return data_frame_row
193
194                 # pickup 一天,  delivery 多天
195                 else:
196                     data_frame_row = copy_reason(data_frame_row, 119)
197                     if delivery_diff == 1:
198                         data_frame_row = write_in_delivery_comments(data_frame_row,
199                                                                     f'pickup late for 1 day and delivery
late for 1 day')
200                         return data_frame_row
201                     else:
202                         data_frame_row = write_in_delivery_comments(data_frame_row,
```

```
                                                                f'pickup late for 1 day and delivery late for {delivery_diff} days')
                        return data_frame_row
                return data_frame_row

            # pick up 晚于 1 天，看 delivery
            elif pickup_diff > 1:

                data_frame_row = write_in_delivery_comments(data_frame_row,
                                                            f'Inbound ontime but outbound late for {pickup_diff} days')
                data_frame_row['Pickup Comments'] = [
                    f'Inbound ontime but outbound late for {pickup_diff} days']
                return data_frame_row
            return data_frame_row

    # pickup 没晚
    else:
        delivery_diff = date_subtract(data_frame_row['Drop off date'].values[0],
                                      data_frame_row['Pickup Date'].values[0])
        if delivery_diff == 0:
            if time_upper_than(data_frame_row['Drop off time'].values[0],
                               data_frame_row['Latest Dropoff Time'].values[0], policy):
                data_frame_row = copy_reason(data_frame_row, 118)
                return data_frame_row
        else:
            data_frame_row = copy_reason(data_frame_row, 119)
            if delivery_diff == 1:

                return data_frame_row
            else:
                if delivery_diff < 2:
                    return data_frame_row

                else:
                    data_frame_row = write_in_delivery_comments(data_frame_row,
                                                                f'pickup ok but delivery late for {delivery_diff} days')
                    return data_frame_row
            return data_frame_row


def get_status(data_frame_row, day, policy):
    # 判断 shipment status 完了
```

```
    if 'GEOCODED'.lower() in str(data_frame_row['Shipment status'].values[0]).lower():
        # 这里还要继续分析，但是比较复杂，不过多写了
        data_frame_row = copy_reason(data_frame_row, 29)
        return data_frame_row
    if 'CANCELLED_BEFORE_PICKUP'.lower() in str(data_frame_row['Shipment status'].values[0]).lower():
        data_frame_row = copy_reason(data_frame_row, 20)
        return data_frame_row
    if 'GEOCODE_FAILED'.lower() in str(data_frame_row['Shipment status'].values[0]).lower():
        data_frame_row = copy_reason(data_frame_row, 22)
        return data_frame_row

    # 判断 Inbound status Missing
    if 'MISSING'.lower() in str(data_frame_row['Inbound status'].values[0]).lower():
        data_frame_row = copy_reason(data_frame_row, 25)
        return data_frame_row
    if 'DAMAGED'.lower() in str(data_frame_row['Inbound status'].values[0]).lower():
        data_frame_row = copy_reason(data_frame_row, 26)
        return data_frame_row

    # 开始逐一检查 Drop off status
    if data_frame_row['Drop off status'].values[0] == 'DISCARDED':
        if "Damaged".lower() in str(data_frame_row['Drop off remark'].values[0]).lower():
            if day == '4':
                data_frame_row = copy_reason(data_frame_row, 5)
                data_frame_row['Pickup Comments'] = ['Inbound ok but pickup damaged']
                return data_frame_row
            if day == '3':
                data_frame_row = copy_reason(data_frame_row, 34)
                data_frame_row['Pickup Comments'] = ['Inbound ok but pickup damaged']
                return data_frame_row
        if 'RECEIVED_DAMAGED'.lower() in str(data_frame_row['Drop off remark'].values[0]).lower():
            data_frame_row = copy_reason(data_frame_row, 26)
            return data_frame_row
        if 'discard'.lower() in str(data_frame_row['Drop off remark'].values[0]).lower():
            if day == '3':
                data_frame_row = copy_reason(data_frame_row, 37)
                data_frame_row['Pickup Comments'] = ['Inbound ok but pickup failed']
                return data_frame_row
            if day == '4':
                data_frame_row = copy_reason(data_frame_row, 3)
                data_frame_row['Pickup Comments'] = ['Inbound ok but pickup failed']
                return data_frame_row
        if "Missing".lower() in str(data_frame_row['Drop off remark'].values[0]).lower():
            if day == '3':
```

```
                data_frame_row = copy_reason(data_frame_row, 35)
                data_frame_row['Pickup Comments'] = ['Inbound ok but pickup failed']
                return data_frame_row
            if day == '4':
                data_frame_row = copy_reason(data_frame_row, 4)
                data_frame_row['Pickup Comments'] = ['Inbound ok but pickup failed']
                return data_frame_row
        if pd.isna(data_frame_row['Drop off remark'].values[0]):
            data_frame_row = copy_reason(data_frame_row, 52)
            return data_frame_row

    if data_frame_row['Drop off status'].values[0] is None:
        if 'missing by inbound' in str(data_frame_row['Drop off remark'].values[0]).lower():
            data_frame_row = copy_reason(data_frame_row, 25)
            return data_frame_row
        if 'missing by outbound' in str(data_frame_row['Drop off remark'].values[0]).lower():
            if day == '3':
                data_frame_row = copy_reason(data_frame_row, 35)
                return data_frame_row
            if day == '4':
                data_frame_row = copy_reason(data_frame_row, 4)
                return data_frame_row

    if data_frame_row['Drop off status'].values[0] == 'EN_ROUTE':
        if data_frame_row['Pickup Status'].values[0] == 'SUCCEEDED':
            data_frame_row = copy_reason(data_frame_row, 52)
            return data_frame_row

    if data_frame_row['Drop off status'].values[0] == 'PENDING':
        if data_frame_row['Pickup Status'].values[0] == 'SUCCEEDED':
            data_frame_row = copy_reason(data_frame_row, 52)
            return data_frame_row
        if data_frame_row['Pickup Status'].values[0] == 'FAILED' or \
                data_frame_row['Pickup Status'].values[0] == 'PENDING':
            if day == '4':
                data_frame_row = copy_reason(data_frame_row, 3)
                return data_frame_row
            elif day == '3':
                data_frame_row = copy_reason(data_frame_row, 31)
                return data_frame_row

    if data_frame_row['Drop off status'].values[0] == 'FAILED':
        # ok
        if pd.isna(data_frame_row['Drop off remark'].values[0]):
```

```
            data_frame_row = copy_reason(data_frame_row, 52)
            return data_frame_row
        # 如果 remark 不是空
        if isinstance(data_frame_row['Drop off remark'].values[0], str):
            # ok
            if 'out of cold chain'.lower() in data_frame_row['Drop off remark'].values[0].lower():
                data_frame_row = copy_reason(data_frame_row, 51)
                return data_frame_row
            # ok
            if 'missing'.lower() in data_frame_row['Drop off remark'].values[0].lower():
                data_frame_row = copy_reason(data_frame_row, 52)
                return data_frame_row
            # ok
            if 'damaged'.lower() in data_frame_row['Drop off remark'].values[0].lower():
                data_frame_row = copy_reason(data_frame_row, 46)
                return data_frame_row
            # ok
            # if 'wrong'.lower() in data_frame_row['Drop off remark'].values[0].lower():
            #     data_frame_row = copy_reason(data_frame_row, 14)
            #     return data_frame_row
            # ok
            if 'no access'.lower() in data_frame_row['Drop off remark'].values[0].lower():
                data_frame_row = copy_reason(data_frame_row, 14)
                return data_frame_row
            # ok
            if 'access code'.lower() in data_frame_row['Drop off remark'].values[0].lower():
                data_frame_row = copy_reason(data_frame_row, 14)
                return data_frame_row
            # ok
            if 'no answer'.lower() in data_frame_row['Drop off remark'].values[0].lower():
                data_frame_row = copy_reason(data_frame_row, 14)
                return data_frame_row
            # ok
            if 'can\'t be reach'.lower() in data_frame_row['Drop off remark'].values[0].lower():
                data_frame_row = copy_reason(data_frame_row, 13)
                return data_frame_row
            # ok
            if 'cant be reach'.lower() in data_frame_row['Drop off remark'].values[0].lower():
                data_frame_row = copy_reason(data_frame_row, 13)
                return data_frame_row
            # ok
            if 'closed'.lower() in data_frame_row['Drop off remark'].values[0].lower():
                data_frame_row = copy_reason(data_frame_row, 14)
                return data_frame_row
```

```
            # ok
            if 'requested redelivery'.lower() in data_frame_row['Drop off remark'].values[0].lower():
                data_frame_row = copy_reason(data_frame_row, 23)
                return data_frame_row
            # ok
            if 'redelivery requested'.lower() in data_frame_row['Drop off remark'].values[0].lower():
                data_frame_row = copy_reason(data_frame_row, 23)
                return data_frame_row
            # ok
            if 'Cancel'.lower() in data_frame_row['Drop off remark'].values[0].lower():
                data_frame_row = copy_reason(data_frame_row, 20)
                return data_frame_row
            # ok
            if 'refused'.lower() in data_frame_row['Drop off remark'].values[0].lower():
                data_frame_row = copy_reason(data_frame_row, 19)

    # 如果是 SUCCEEDED 状态
    if data_frame_row['Drop off status'].values[0] == 'SUCCEEDED':
        # 先查看一些明显的问题
        if 'no access'.lower() in str(data_frame_row['Drop off remark'].values[0]).lower():
            data_frame_row = copy_reason(data_frame_row, 14)
        elif 'no answer'.lower() in str(data_frame_row['Drop off remark'].values[0]).lower():
            data_frame_row = copy_reason(data_frame_row, 14)
        elif 'no code'.lower() in str(data_frame_row['Drop off remark'].values[0]).lower():
            data_frame_row = copy_reason(data_frame_row, 14)

        # 再看时间差，附着到之前的结果
        inbound_diff = date_subtract(data_frame_row['Inbound Scan Date (Linehaul)'].values[0],
                                     data_frame_row['Scheduled Delivery Date'].values[0])
        # 如果 inbound_diff 等于 0
        if inbound_diff == 0:
            # 如果 inbound 当天晚于 12 点
            if time_upper_than(data_frame_row['Inbound Scan Time'].values[0], '12:00', 0):
                data_frame_row['Inbound Comments'] = ['Inbound late']
                data_frame_row = copy_reason(data_frame_row, 24)
                return data_frame_row
            # 如果 inbound 当天早于 12 点
            else:
                get_pickup_and_delivery_status(data_frame_row, day, policy)

        # 如果 inbound_diff 大于一天
        elif inbound_diff > 0:
            data_frame_row['Inbound Comments'] = ['Inbound late']
            data_frame_row = copy_reason(data_frame_row, 24)
```

```
420
421             return data_frame_row
422
423         # 当 Inbound 没 late
424         else:
425             get_pickup_and_delivery_status(data_frame_row, day, policy)
426
427     return data_frame_row
428
429
430 def date_subtract(compared_date, schedule_date):
431     if pd.isna(compared_date) or str(compared_date) == 'nan':
432         return -100
433     if pd.isna(schedule_date) or str(schedule_date) == 'nan':
434         return -100
435     else:
436         try:
437             compared_date = datetime.datetime.strptime(compared_date, '%Y-%m-%d')
438             # 如果 scheduled date 是月日年
439             if is_mouth_day_year(schedule_date):
440                 schedule_date = datetime.datetime.strptime(schedule_date, '%m/%d/%Y')
441             else:
442                 schedule_date = datetime.datetime.strptime(schedule_date, '%Y-%m-%d')
443             return (compared_date - schedule_date).days
444         except ValueError:
445             return -100
446
447
448 def time_upper_than(time_str, upper, policy):
449     upper_time = datetime.datetime.strptime(upper, '%H:%M')
450     upper_time += datetime.timedelta(minutes=int(policy))
451     time_str = datetime.datetime.strptime(time_str, '%H:%M')
452     if (int(upper_time.strftime('%H%M')) - int(time_str.strftime('%H%M'))) > 0:
453         return False
454     else:
455         return True
456
457
458 def write_in_delivery_comments(data_frame_row, string):
459     if pd.isna(data_frame_row['Delivery Comments'].values[0]):
460         data_frame_row['Delivery Comments'] = [string]
461         return data_frame_row
462     # 如果已经有了，就不管了
463     elif string in data_frame_row['Delivery Comments'].values[0]:
```

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

```
551                 new_time = time_subtract(inbound_time_str, hours=3, days=0)
552                 new_time_str = new_time.strftime('%H:%M')
553                 data_frame_row['Inbound Scan Time'] = [new_time_str]
554 
555             # 针对 pickup time
556             if pd.isna(data_frame_row['Pickup Time'].values[0]):
557                 pass
558             else:
559                 pickup_time_str = str(data_frame_row['Pickup Time'].values[0])
560                 new_pickup_time = time_subtract(pickup_time_str, hours=3, days=0)
561                 new_pickup_time_str = new_pickup_time.strftime('%H:%M')
562                 data_frame_row['Pickup Time'] = [new_pickup_time_str]
563 
564             # 针对 drop off time
565             if pd.isna(data_frame_row['Drop off time'].values[0]):
566                 pass
567             else:
568                 drop_time_str = str(data_frame_row['Drop off time'].values[0])
569                 new_drop_time = time_subtract(drop_time_str, hours=3, days=0)
570                 new_drop_time_str = new_drop_time.strftime('%H:%M')
571                 data_frame_row['Drop off time'] = [new_drop_time_str]
572 
573             # 如果前进了一天
574             if new_time is None:
575                 return data_frame_row
576             else:
577                 if str(new_time.date()) == '1899-12-31':
578                     date_str = str(data_frame_row['Inbound Scan Date (Linehaul)'].values[0])
579 
580                     time_object = datetime.datetime.strptime(date_str, '%Y-%m-%d')
581                     new_date = time_object - datetime.timedelta(days=1)
582                     new_date_str = new_date.strftime('%Y-%m-%d')
583 
584                     data_frame_row['Inbound Scan Date (Linehaul)'] = [new_date_str]
585                     return data_frame_row
586                 else:
587                     return data_frame_row
588 
589         elif region == 'PHX':
590             # early 时间 latest 时间
591             early_time_str = str(data_frame_row['Earliest Dropoff Time'].values[0])
592             new_time = time_subtract(early_time_str, hours=1, days=0)
593             new_time_str = new_time.strftime('%H:%M')
594             data_frame_row['Earliest Dropoff Time'] = [new_time_str]
```

```
595
596             latest_time_str = str(data_frame_row['Latest Dropoff Time'].values[0])
597             new_time = time_subtract(latest_time_str, hours=1, days=0)
598             new_time_str = new_time.strftime('%H:%M')
599             data_frame_row['Latest Dropoff Time'] = [new_time_str]
600             # 针对 inbound
601             # 如果 时间有空的，跳过
602             if pd.isna(data_frame_row['Inbound Scan Time'].values[0]):
603                 new_time = None
604             else:
605                 inbound_time_str = str(data_frame_row['Inbound Scan Time'].values[0])
606                 new_time = time_subtract(inbound_time_str, hours=1, days=0)
607                 new_time_str = new_time.strftime('%H:%M')
608                 data_frame_row['Inbound Scan Time'] = [new_time_str]
609
610             # 针对 pickup time
611             if pd.isna(data_frame_row['Pickup Time'].values[0]):
612                 pass
613             else:
614                 pickup_time_str = str(data_frame_row['Pickup Time'].values[0])
615                 new_pickup_time = time_subtract(pickup_time_str, hours=1, days=0)
616                 new_pickup_time_str = new_pickup_time.strftime('%H:%M')
617                 data_frame_row['Pickup Time'] = [new_pickup_time_str]
618
619             # 针对 drop off time
620             if pd.isna(data_frame_row['Drop off time'].values[0]):
621                 pass
622             else:
623                 drop_time_str = str(data_frame_row['Drop off time'].values[0])
624                 new_drop_time = time_subtract(drop_time_str, hours=1, days=0)
625                 new_drop_time_str = new_drop_time.strftime('%H:%M')
626                 data_frame_row['Drop off time'] = [new_drop_time_str]
627
628             # 如果前进了一天
629             if new_time is None:
630                 return data_frame_row
631             else:
632                 if str(new_time.date()) == '1899-12-31':
633                     date_str = str(data_frame_row['Inbound Scan Date (Linehaul)'].values[0])
634
635                     time_object = datetime.datetime.strptime(date_str, '%Y-%m-%d')
636                     new_date = time_object - datetime.timedelta(days=1)
637                     new_date_str = new_date.strftime('%Y-%m-%d')
638
```

```
                    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:
        return data_frame_row


def time_subtract(time_str, hours, days):
    time_object = datetime.datetime.strptime(time_str, '%H:%M')
    new_time = time_object - datetime.timedelta(hours=hours, days=days)
    return new_time


def change_Scheduled_Delivery_Date(data_frame_row):
    s_date_str = data_frame_row['Scheduled Delivery Date'].values[0]
    if format_1(s_date_str):
        s_str = datetime.datetime.strptime(s_date_str, '%Y/%m/%d')
        s_str = s_str.strftime('%Y-%m-%d')
        data_frame_row['Scheduled Delivery Date'] = [s_str]
        return data_frame_row
    elif format_2(s_date_str):
        s_str = datetime.datetime.strptime(s_date_str, '%m/%d/%Y')
        s_str = s_str.strftime('%Y-%m-%d')
        data_frame_row['Scheduled Delivery Date'] = [s_str]
        return data_frame_row
    elif format_3(s_date_str):
        return data_frame_row
    else:
        return data_frame_row


def format_1(date):
    try:
        datetime.datetime.strptime(date, "%Y/%m/%d")
        return True
    except:
        return False


def format_2(date):
```

```
683     try:
684         datetime.datetime.strptime(date, "%m/%d/%Y")
685         return True
686     except:
687         return False
688 
689 
690 def format_3(date):
691     try:
692         datetime.datetime.strptime(date, "%Y-%m-%d")
693         return True
694     except:
695         return False
696 
```

concat_csv.py

```
1   import pandas as pd
2   import os
3   
4   from tkinter import Toplevel, StringVar, Label, Button, Entry, messagebox
5   from tkinter.filedialog import askdirectory
6   
7   
8   class Concat(object):
9       def __init__(self, root):
10          self.root = root
11  
12      def get_folder_path(self):
13          folder_path = askdirectory()
14          self.folder_path.set(folder_path)
15  
16      def concat_from_folder(self):
17          dir_list = os.listdir(self.folder_path.get())
18          res_df = pd.DataFrame()
19          for file in dir_list:
20              file_path = self.folder_path.get() + '/' + file
21              temp_df = pd.read_csv(file_path)
22              res_df = pd.concat([res_df, temp_df])
23          res_df.to_csv(self.folder_path.get() + '/all.csv', index=False)
24          path = self.folder_path.get() + '/all.csv'
25          messagebox.showinfo(title='成功', message=f'输出路径为: {path}')
26          return res_df
27  
28      def run(self):
29          self.window = Toplevel(master=self.root)
```

```
30            self.window.geometry('1000x120')
31            self.folder_path = StringVar()
32
33            # label ending
34            Label(self.window, text="要合并的文件夹:").place(x=100, y=50)
35            Entry(self.window, textvariable=self.folder_path, width='60').place(x=220, y=50)
36            Button(self.window, text="选择文件夹", command=self.get_folder_path, width='10').place(x=680,
y=50)
37
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
```

const.py

```
1    USER_NAME = "your_axlehire_user_name"
2    USER_PSW = "your_axlehire_user_password"
3
```

discriminator_arch.py

```
1    from basicsr.utils.registry import ARCH_REGISTRY
2    from torch import nn as nn
3    from torch.nn import functional as F
4    from torch.nn.utils import spectral_norm
5
6
7    @ARCH_REGISTRY.register()
8    class UNetDiscriminatorSN(nn.Module):
9        """Defines a U-Net discriminator with spectral normalization (SN)
10
11       It is used in Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data.
12
13       Arg:
14           num_in_ch (int): Channel number of inputs. Default: 3.
15           num_feat (int): Channel number of base intermediate features. Default: 64.
16           skip_connection (bool): Whether to use skip connections between U-Net. Default: True.
17       """
18
19       def __init__(self, num_in_ch, num_feat=64, skip_connection=True):
20           super(UNetDiscriminatorSN, self).__init__()
21           self.skip_connection = skip_connection
22           norm = spectral_norm
23           # the first convolution
24           self.conv0 = nn.Conv2d(num_in_ch, num_feat, kernel_size=3, stride=1, padding=1)
25           # downsample
```

```
        self.conv1 = norm(nn.Conv2d(num_feat, num_feat * 2, 4, 2, 1, bias=False))
        self.conv2 = norm(nn.Conv2d(num_feat * 2, num_feat * 4, 4, 2, 1, bias=False))
        self.conv3 = norm(nn.Conv2d(num_feat * 4, num_feat * 8, 4, 2, 1, bias=False))
        # upsample
        self.conv4 = norm(nn.Conv2d(num_feat * 8, num_feat * 4, 3, 1, 1, bias=False))
        self.conv5 = norm(nn.Conv2d(num_feat * 4, num_feat * 2, 3, 1, 1, bias=False))
        self.conv6 = norm(nn.Conv2d(num_feat * 2, num_feat, 3, 1, 1, bias=False))
        # extra convolutions
        self.conv7 = norm(nn.Conv2d(num_feat, num_feat, 3, 1, 1, bias=False))
        self.conv8 = norm(nn.Conv2d(num_feat, num_feat, 3, 1, 1, bias=False))
        self.conv9 = nn.Conv2d(num_feat, 1, 3, 1, 1)

    def forward(self, x):
        # downsample
        x0 = F.leaky_relu(self.conv0(x), negative_slope=0.2, inplace=True)
        x1 = F.leaky_relu(self.conv1(x0), negative_slope=0.2, inplace=True)
        x2 = F.leaky_relu(self.conv2(x1), negative_slope=0.2, inplace=True)
        x3 = F.leaky_relu(self.conv3(x2), negative_slope=0.2, inplace=True)

        # upsample
        x3 = F.interpolate(x3, scale_factor=2, mode='bilinear', align_corners=False)
        x4 = F.leaky_relu(self.conv4(x3), negative_slope=0.2, inplace=True)

        if self.skip_connection:
            x4 = x4 + x2
        x4 = F.interpolate(x4, scale_factor=2, mode='bilinear', align_corners=False)
        x5 = F.leaky_relu(self.conv5(x4), negative_slope=0.2, inplace=True)

        if self.skip_connection:
            x5 = x5 + x1
        x5 = F.interpolate(x5, scale_factor=2, mode='bilinear', align_corners=False)
        x6 = F.leaky_relu(self.conv6(x5), negative_slope=0.2, inplace=True)

        if self.skip_connection:
            x6 = x6 + x0

        # extra convolutions
        out = F.leaky_relu(self.conv7(x6), negative_slope=0.2, inplace=True)
        out = F.leaky_relu(self.conv8(out), negative_slope=0.2, inplace=True)
        out = self.conv9(out)

        return out

```

downloader.py

```
import os
import datetime
import time
import requests
import json
import pandas as pd

from const import USER_PSW, USER_NAME
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 = USER_NAME
        self.password = USER_PSW

    def download_from_url(self, url, file_name):
        self.make_dir('all_report_history')
        session = requests.Session()
        time.sleep(5)
        response = session.get(url=url, headers=self.header)
        if response.status_code == 200:
            json_data = json.loads(response.text)
            response = session.get(url=url+'/download', headers=self.header)
            if 'url' in json_data.keys():
                with open(file_name, 'wb') as fp:
                    fp.write(response.content)
            else:
                self.download_from_url(url, file_name)

```

```
42          else:
43              self.download_from_url(url, file_name)
44
45      def get_csv_from_date(self, client_id, date, file_name):
46          session = requests.Session()
47          if client_id.find(', ') == -1:
48              client_id_string = [str(client_id)]
49          else:
50              client_id_string = client_id.split(', ')
51
52          # date
53          date = datetime.datetime.strptime(date, '%Y/%m/%d').strftime('%Y-%m-%d')
54
55          json_data = {
56              'clients': client_id_string,
57              'date': date,
58          }
59
60          response = session.post(url=self.url, headers=self.header, json=json_data,
61                                  auth=HttpNtlmAuth(self.user_name, self.password))
62          json_response = json.loads(response.content)
63          url = 'https://dataorch.beta.axlehire.com/reports/uploaded/'
64          url += json_response['id']
65          self.download_from_url(url, file_name)
66
67      def get_date_list(self, from_date, to_date):
68          from_date = datetime.datetime.strptime(from_date, '%Y/%m/%d')
69          to_date = datetime.datetime.strptime(to_date, '%Y/%m/%d')
70          diff = (to_date - from_date).days
71          if diff <= 0:
72              messagebox.showwarning(title='警告', message='日期范围有误')
73          else:
74              date_list = [from_date.strftime('%Y/%m/%d')]
75              for i in range(1, diff+1):
76                  date = from_date + datetime.timedelta(days=i)
77                  date = date.strftime('%Y/%m/%d')
78                  date_list.append(date)
79              return date_list
80
81      @staticmethod
82      def make_dir(path):
83          if not os.path.exists(path):
84              os.mkdir(path)
85
```

```
86      def run(self):
87          # 初始化界面
88          self.window.geometry('700x240')
89          self.client = StringVar()
90          self.date = StringVar()
91          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)
92          Entry(self.window, textvariable=self.client).place(x=50, y=60)
93          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）:')\
94              .place(x=50, y=100)
95          Entry(self.window, textvariable=self.date).place(x=50, y=140)
96          Button(self.window, text='生成 all_report csv', command=self.confirm).place(x=50, y=190)
97          self.window.mainloop()
98
99      @staticmethod
100     def is_date(date):
101         try:
102             datetime.datetime.strptime(date, "%Y/%m/%d")
103             return True
104         except:
105             return False
106
107     def confirm(self):
108         if self.check_client():
109             # date 为 range
110             now = datetime.datetime.now().strftime('%m 月%d 日-%H 点%M 分%S 秒')
111             if self.date.get().find('-') != -1:
112                 date_from_to_list = self.date.get().split('-')
113                 date_list = self.get_date_list(from_date=date_from_to_list[0],
to_date=date_from_to_list[1])
114                 # 创建文件夹
115                 folder_name = self.date.get().replace('-', 'to')
116                 folder_name = folder_name.replace('/', '-')
117                 folder_name += '&client=' + self.client.get() + '_' + now
118                 self.make_dir(f'all_report_history/{folder_name}')
119                 for date in date_list:
120                     date_name = date.replace('/', '-')
121                     self.get_csv_from_date(
122                         client_id=self.client.get(),
123                         date=date,
124
file_name=f'all_report_history/{folder_name}/client={self.client.get()}&date='f'{date_name}&{now}.csv'
125                         )
126                 self.concat_from_folder(f'all_report_history/{folder_name}')
```

```
127             # 单个 date
128             elif self.is_date(self.date.get()):
129                 date = datetime.datetime.strptime(self.date.get(), '%Y/%m/%d').strftime('%Y-%m-%d')
130                 self.get_csv_from_date(
131                     client_id=self.client.get(),
132                     date=self.date.get(),
133                     file_name=f'all_report_history/client={self.client.get()}&date={date}&{now}.csv'
134                 )
135             else:
136                 messagebox.showwarning(title='警告', message='日期格式有误')
137         else:
138             messagebox.showwarning(title='警告', message='client 格式有误')
139 
140     def check_client(self):
141         # client 是单个
142         if self.client.get().find(', ') == -1:
143             if not self.client.get().isnumeric():
144                 return False
145             # 是数字
146             else:
147                 if int(self.client.get()) <= 11 or int(self.client.get()) == 471 or
int(self.client.get()) == 621 \
148                         or (int(self.client.get()) >= 15 and int(self.client.get()) <= 214):
149                     return True
150                 return False
151         # client 是多个
152         else:
153             client_list = self.client.get().split(', ')
154             for client in client_list:
155                 if client.find(', ') == -1:
156                     if not client.isnumeric():
157                         return False
158                     # 是数字
159                     else:
160                         if int(client) <= 214 or int(client) == 471 or int(client) == 621:
161                             return True
162                         return False
163 
164     @staticmethod
165     def get_dict_from_tracking_code(tracking_code):
166         url = 'https://dataorch.axlehire.com/shipments/search'
167         header = {
168             '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',
```

```
169             'content-type': 'application/json',
170             'cookie': r'fp=1a39e1225ea764ca9f2abf599fafba34;
xtoken="dE9DbW1wYkZDI/B28g5MkirtzwljFDty7THWI75r/mVq4do8YKOJBeUtONSQ1d3L1Yb5JCAEZPTk\012FFj7LXpbKjSaV71j1S6I9z
jtTLurIi1ddgqe+xsIRU84cjgOSktu\012"'}
171         # 生成 post 的 json_data
172         data_dict = {'size': 15, 'q': tracking_code,
173                      'filters': {}, 'sorts': ['-dropoff_earliest_ts']}
174         json_data = json.dumps(data_dict)
175
176         session = requests.Session()
177         user = USER_NAME
178         password = USER_PSW
179         response = session.post(url=url, headers=header, data=json_data, auth=HttpNtlmAuth(user,
password))
180
181         result_dict = json.loads(response.text)
182         return result_dict
183
184     @staticmethod
185     def concat_from_folder(folder_path):
186         dir_list = os.listdir(folder_path)
187         res_df = pd.DataFrame()
188         for file in dir_list:
189             file_path = folder_path + '/' + file
190             temp_df = pd.read_csv(file_path)
191             res_df = pd.concat([res_df, temp_df])
192         res_df.to_csv(folder_path + '/all.csv', index=False)
193         path = folder_path + '/all.csv'
194         messagebox.showinfo(title='成功', message=f'输出路径为: {path}')
195         return res_df
196
```

generator.py

```
1   """
2       填完数字之后，传回来加数字的 csv，点击生成，出结果
3   """
4   import warnings
5   import pandas as pd
6
7   from tkinter import Toplevel
8
9
10  warnings.filterwarnings('ignore')
11
12
```

```
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(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]

        # 不是数字, 一定是 14 15 apt 这种形式
        if not str(number).isnumeric():
            answer_list = str(number).split(' ')
            # 三种形式 pic shows building #,the correct is #
            if 'apt' in answer_list:
                data_frame_rows = self.copy_rows(data_frame_rows, int(109))
                apt_answer = str(data_frame_rows['POD Quality'].values[0]). \
                    replace('shows apt #', f'shows apt #{answer_list[0]}'). \
                    replace('the correct is apt#', f'the correct is apt#{answer_list[1]}')
                data_frame_rows['POD Quality'] = [apt_answer]
                return data_frame_rows
            elif 'st' in answer_list:
                data_frame_rows = self.copy_rows(data_frame_rows, int(108))
                s_answer = str(data_frame_rows['POD Quality'].values[0]). \
                    replace('shows street #', f'shows street #{answer_list[0]}'). \
                    replace('the correct is #', f'the correct is #{answer_list[1]}')
```

```
56                  data_frame_rows['POD Quality'] = [s_answer]
57                  return data_frame_rows
58              elif 'b' in answer_list:
59                  data_frame_rows = self.copy_rows(data_frame_rows, int(107))
60                  b_answer = str(data_frame_rows['POD Quality'].values[0]). \
61                      replace('shows building #', f'shows building #{answer_list[0]}'). \
62                      replace('the correct is #', f'the correct is #{answer_list[1]}')
63                  data_frame_rows['POD Quality'] = [b_answer]
64                  return data_frame_rows
65              else:
66                  return data_frame_rows
67          else:
68              data_frame_rows = self.copy_rows(data_frame_rows, int(number))
69              return data_frame_rows
70
71      def copy_rows(self, data_frame_row, index):
72          pd.set_option("display.max_columns", 50)
73
74          def nan_to_none(x):
75              if str(x) == 'nan' or pd.isna(x):
76                  return ''
77              return x
78
79          if pd.isna(data_frame_row['POD Valid?'].values[0]) and pd.isna(data_frame_row['POD
Quality'].values[0]) and \
80                  pd.isna(data_frame_row['Issue Category'].values[0]) and pd.isna(
81              data_frame_row['Delivery Comments'].values[0]) and \
82                  pd.isna(data_frame_row['AH Assessment'].values[0]):
83              data_frame_row['POD Valid?'] = [nan_to_none(self.reason_code.loc[index, 'POD'])]
84              data_frame_row['POD Quality'] = [nan_to_none(self.reason_code.loc[index, 'POD Qaulity'])]
85              data_frame_row['Issue Category'] = [self.reason_code.loc[index, 'Issue Category']]
86              data_frame_row['Delivery Comments'] = [self.reason_code.loc[index, 'Delivery Comments']]
87              data_frame_row['AH Assessment'] = [self.reason_code.loc[index, 'AH Assignment']]
88
89              return data_frame_row
90
91          # 如果不是空的，加一个 / 再将内容附着上
92          else:
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:
101                   data_frame_row['Issue Category'] = [
102                       str(data_frame_row['Issue Category'].values[0]) + '/' + self.reason_code.loc[
103                           index, 'Issue Category']]
104               data_frame_row['Delivery Comments'] = [
105                   str(data_frame_row['Delivery Comments'].values[0]) + '/' + self.reason_code.loc[
106                       index, 'Delivery Comments']]
107               data_frame_row['AH Assessment'] = [
108                   str(data_frame_row['AH Assessment'].values[0]) + '/' + self.reason_code.loc[index, 'AH
Assignment']]
109               return data_frame_row
110
```

inference_realesrgan.py

```
1   import argparse
2   import cv2
3   import glob
4   import os
5   from basicsr.archs.rrdbnet_arch import RRDBNet
6
7   from realesrgan import RealESRGANer
8   from realesrgan.archs.srvgg_arch import SRVGGNetCompact
9
10
11  def main():
12      """Inference demo for Real-ESRGAN.
13      """
14      parser = argparse.ArgumentParser()
15      parser.add_argument('-i', '--input', type=str, default='inputs', help='Input image or folder')
16      parser.add_argument(
17          '-n',
18          '--model_name',
19          type=str,
20          default='RealESRGAN_x4plus',
21          help=('Model names: RealESRGAN_x4plus | RealESRNet_x4plus | RealESRGAN_x4plus_anime_6B |
RealESRGAN_x2plus | '
22                'realesr-animevideov3'))
23      parser.add_argument('-o', '--output', type=str, default='results', help='Output folder')
24      parser.add_argument('-s', '--outscale', type=float, default=4, help='The final upsampling scale of
the image')
25      parser.add_argument('--suffix', type=str, default='out', help='Suffix of the restored image')
```

```
26        parser.add_argument('-t', '--tile', type=int, default=0, help='Tile size, 0 for no tile during testing')
27
28        # add denoise model.
29        parser.add_argument('--denoise', action='store_true', help='Image pre-denoise with GRFDN')
30
31        parser.add_argument('--tile_pad', type=int, default=10, help='Tile padding')
32        parser.add_argument('--pre_pad', type=int, default=0, help='Pre padding size at each border')
33        parser.add_argument('--face_enhance', action='store_true', help='Use GFPGAN to enhance face')
34        parser.add_argument(
35            '--fp32', action='store_true', help='Use fp32 precision during inference. Default: fp16 (half precision).')
36        parser.add_argument(
37            '--alpha_upsampler',
38            type=str,
39            default='realesrgan',
40            help='The upsampler for the alpha channels. Options: realesrgan | bicubic')
41        parser.add_argument(
42            '--ext',
43            type=str,
44            default='auto',
45            help='Image extension. Options: auto | jpg | png, auto means using the same extension as inputs')
46        parser.add_argument(
47            '-g', '--gpu-id', type=int, default=None, help='gpu device to use (default=None) can be 0,1,2 for multi-gpu')
48
49        args = parser.parse_args()
50
51        # determine models according to model names
52        args.model_name = args.model_name.split('.')[0]
53        if args.model_name in ['RealESRGAN_x4plus', 'RealESRNet_x4plus']:  # x4 RRDBNet model
54            model = RRDBNet(num_in_ch=3, num_out_ch=3, num_feat=64, num_block=23, num_grow_ch=32, scale=4)
55            netscale = 4
56        elif args.model_name in ['RealESRGAN_x4plus_anime_6B']:  # x4 RRDBNet model with 6 blocks
57            model = RRDBNet(num_in_ch=3, num_out_ch=3, num_feat=64, num_block=6, num_grow_ch=32, scale=4)
58            netscale = 4
59        elif args.model_name in ['RealESRGAN_x2plus']:  # x2 RRDBNet model
60            model = RRDBNet(num_in_ch=3, num_out_ch=3, num_feat=64, num_block=23, num_grow_ch=32, scale=2)
61            netscale = 2
62        elif args.model_name in ['realesr-animevideov3']:  # x4 VGG-style model (XS size)
63            model = SRVGGNetCompact(num_in_ch=3, num_out_ch=3, num_feat=64, num_conv=16, upscale=4, act_type='prelu')
64            netscale = 4
65
```

```
66      # determine model paths
67      model_path = os.path.join('experiments/pretrained_models', args.model_name + '.pth')
68      if not os.path.isfile(model_path):
69          model_path = os.path.join('realesrgan/weights', args.model_name + '.pth')
70      if not os.path.isfile(model_path):
71          raise ValueError(f'Model {args.model_name} does not exist.')
72
73      # restorer
74      upsampler = RealESRGANer(
75          scale=netscale,
76          model_path=model_path,
77          model=model,
78          tile=args.tile,
79          tile_pad=args.tile_pad,
80          pre_pad=args.pre_pad,
81          half=not args.fp32,
82          gpu_id=args.gpu_id)
83
84      if args.face_enhance:  # Use GFPGAN for face enhancement
85          from gfpgan import GFPGANer
86          face_enhancer = GFPGANer(
87              model_path='https://github.com/TencentARC/GFPGAN/releases/download/v1.3.0/GFPGANv1.3.pth',
88              upscale=args.outscale,
89              arch='clean',
90              channel_multiplier=2,
91              bg_upsampler=upsampler)
92      os.makedirs(args.output, exist_ok=True)
93
94      if os.path.isfile(args.input):
95          paths = [args.input]
96      else:
97          paths = sorted(glob.glob(os.path.join(args.input, '*')))
98
99      for idx, path in enumerate(paths):
100         imgname, extension = os.path.splitext(os.path.basename(path))
101         print('Testing', idx, imgname)
102
103         img = cv2.imread(path, cv2.IMREAD_UNCHANGED)
104         if len(img.shape) == 3 and img.shape[2] == 4:
105             img_mode = 'RGBA'
106         else:
107             img_mode = None
108
109         # adding GRFDNET algorithm, denoise img first
```

```
110         if args.denoise:
111             img = GRFDN(img)
112 
113         try:
114             if args.face_enhance:
115                 _, _, output = face_enhancer.enhance(img, has_aligned=False, only_center_face=False,
paste_back=True)
116             else:
117                 output, _ = upsampler.enhance(img, outscale=args.outscale)
118         except RuntimeError as error:
119             print('Error', error)
120             print('If you encounter CUDA out of memory, try to set --tile with a smaller number.')
121         else:
122             if args.ext == 'auto':
123                 extension = extension[1:]
124             else:
125                 extension = args.ext
126             if img_mode == 'RGBA':  # RGBA images should be saved in png format
127                 extension = 'png'
128             if args.suffix == '':
129                 save_path = os.path.join(args.output, f'{imgname}.{extension}')
130             else:
131                 save_path = os.path.join(args.output, f'{imgname}_{args.suffix}.{extension}')
132             cv2.imwrite(save_path, output)
133 
134 
135 if __name__ == '__main__':
136     main()
137 
```

main.py

```
1   """
2       最新版 script 代码重构，流程分为几个部分
3       1. 直接运行，显示主窗口
4           包含了两个自行填写的内容
5           ① google sheet 的 url （boss 发来的 google sheet 对应的网址链接）
6           ② 需要将项目生成的文件夹 （选择一个空文件夹，会将整个工作项目自动生成到此文件夹下，结构如下：）
7               your_folder_name
8                   ├──all_report （存放 all_report.csv）
9                   ├──boss_to_me （存放 boss 发来的原始任务文件）
10                  └──result     （存放两个文件，vlook_ending_file.csv，vlook&processed_ending_file.csv）
11      2. 复制 url 和选定项目文件夹后，会显示进度窗口
12          包含了进度条以及说明
13          ① 进度条显示目前正在处理那个环节（获取 all_report 中，生成 vlook&processing_ending_file 中等）
14      3. 当进度条结束，显示看照片页面
```

```
15  """
16
17  import datetime
18  import os
19  import sys
20  import time
21  import warnings
22
23  import pandas as pd
24  import concat_csv
25  import downloader
26
27  from tkinter import messagebox, Tk, StringVar, Label, Button, Entry, Text, Toplevel, ttk
28  from tkinter.font import Font
29  from tkinter.filedialog import askopenfilename, askdirectory
30  from utils.preprocessing_data import preprocessing_data
31  from utils.analyser import Wednesday, Thursday, Analyser
32
33  warnings.filterwarnings('ignore')
34
35
36  class Main(object):
37      def __init__(self):
38          self.window = Tk()
39          self.ending_show = StringVar()
40          self.ending_path = StringVar()
41          self.ending_df = None
42          self.boss2me_path = StringVar()
43          self.boss2me_df = None
44          self.all_report_path = StringVar()
45          self.all_report_df = None
46          self.save_folder_path = StringVar()
47          self.result_df = None
48          self.version = 'V1.0'
49          self.day = None
50          self.structured_df = None
51          self.message = None
52          self.select_box = None
53
54      def _get_ending(self, *args):
55          if self.select_box.get() == '智能分析':
56              ending_path = os.getcwd() + '/utils/files/ending_wednesday.csv'
57              self.ending_show.set('智能分析')
58          else:
```

```
59              ending_path = os.getcwd() + '/utils/files/ending_thursday.csv'
60              self.ending_show.set('普通分析')
61
62          self.ending_path.set(ending_path)
63          self.window.update()
64
65      def _get_boss2me(self):
66          boss2me_path = askopenfilename()
67          self.boss2me_path.set(boss2me_path)
68          self.window.update()
69
70      def _get_all_report(self):
71          all_report_path = askopenfilename()
72          self.all_report_path.set(all_report_path)
73          self.window.update()
74
75      def _get_save_folder_path(self):
76          save_folder_path = askdirectory()
77          self.save_folder_path.set(save_folder_path)
78          self.window.update()
79
80      def get_message(self, structured_df, day):
81          if day == '3':
82              structured_df = structured_df[pd.isna(structured_df['Region Code'])]
83          elif day == '4':
84              structured_df = structured_df[pd.isna(structured_df['Client'])]
85
86          if structured_df.empty:
87              choice = messagebox.askyesno(title='成功', message='没有任何问题，是否继续')
88              if choice:
89                  self.next()
90              else:
91                  return None
92          else:
93              self.message = Toplevel(master=self.window)
94              self.message.geometry('1200x600')
95              self.message.title = '有错误'
96
97              # 设置一个 Text
98              font = Font(size=16)
99              text = Text(self.message, width=80, height=20, font=font)
100
101             date_list = structured_df['Scheduled Delivery Date'].to_list()
102             tracking_code_list = structured_df['Tracking Code'].to_list()
```

```
103
104             # 创建 dict
105             date_dict = {}
106             for date in date_list:
107                 date_dict[date] = []
108                 for tracking_code in tracking_code_list:
109                     date_dict[date].append(tracking_code)
110
111             message = ''
112             for date, tracking_list in date_dict.items():
113                 message += '未搜索到的 tracking_code 为: '
114                 for tracking_code in tracking_list:
115                     message += tracking_code + '/'
116                 break
117
118             text.pack()
119             text.insert('insert', message)
120
121             Button(self.message, text='退出并继续', command=self.next).place(x=600, y=500)
122             self.message.mainloop()
123
124     def pre_check(self):
125         if self.ending_path.get() == '' or self.boss2me_path.get() == '' or self.all_report_path.get() ==
'' or \
126                 self.save_folder_path.get() == '':
127             messagebox.showwarning(title='警告', message='有尚未选择的路径')
128
129     def generate_csv(self):
130         # 先检查是否选择路径
131         self.pre_check()
132         self.ending_df = pd.read_csv(self.ending_path.get())
133         self.boss2me_df = pd.read_csv(self.boss2me_path.get())
134         self.all_report_df = pd.read_csv(self.all_report_path.get())
135
136         # 首先，经过一个筛选函数，将各种客户进行初处理，合并到一起
137         structured_df = None
138         if 'wednesday' in self.ending_path.get().lower():
139             structured_df = preprocessing_data(self.ending_df, self.boss2me_df, self.all_report_df,
day='3')
140             structured_df = structured_df.rename(columns={'delivery_date': 'Scheduled Delivery Date'})
141         elif 'thursday' in self.ending_path.get().lower():
142             structured_df = preprocessing_data(self.ending_df, self.boss2me_df, self.all_report_df,
day='4')
143         else:
```

```
144                 messagebox.showinfo(title='错误', message='ending file 有误，请检查')
145
146             # 生成 csv
147             date_time = datetime.datetime.now().strftime('%Y-%m-%d %H-%M-%S')
148             structured_df.to_csv(str(self.save_folder_path.get()) + '/初版' + date_time + '.csv',
index=False)
149             self.structured_df = structured_df
150
151             # 检查policy有没有错
152             if self.policy.get().isnumeric():
153                 # 开始逐行分析
154                 if 'thursday' in str(self.ending_path.get()).lower():
155
156                     # 如果发现日期没对齐，显示出少了那些日期
157                     self.day = '4'
158                     self.get_message(structured_df, day=self.day)
159
160                 elif 'wednesday' in str(self.ending_path.get()).lower():
161                     # 如果发现日期没对齐，显示出少了那些日期
162                     self.day = '3'
163                     self.get_message(structured_df, day=self.day)
164             else:
165                 messagebox.showerror(title='policy错误', message='policy填写有误')
166
167         def next(self):
168             if self.day == '4':
169                 thursday = Thursday(self.structured_df, policy=self.policy.get())
170                 self.result_df = thursday.analyse()
171
172                 # 生成 csv
173                 date_time = datetime.datetime.now().strftime('%Y-%m-%d %H-%M-%S')
174                 self.result_df.drop_duplicates(subset=['Tracking Code'], inplace=True)
175                 self.result_df.to_csv(str(self.save_folder_path.get()) + '/first' + date_time + '.csv',
index=False)
176
177                 analyser = Analyser(self.window, self.result_df, self.save_folder_path.get(), '4')
178                 analyser.run()
179
180             elif self.day == '3':
181                 # 列名先改一下
182                 self.structured_df.rename(columns={'Drop off Time': 'Drop off time'}, inplace=True)
183
184                 wednesday = Wednesday(self.structured_df, policy=self.policy.get())
185                 self.result_df = wednesday.analyse()
```

```
186
187             # 列名先改回来
188             self.result_df.rename(columns={'Drop off time': 'Drop off Time'}, inplace=True)
189
190             # 生成 csv
191             res_df = self.result_df.copy()
192             try:
193                 res_df = res_df.drop(columns=['Week#', 'Updated Reason Code'])
194             except BaseException:
195                 pass
196             date_time = datetime.datetime.now().strftime('%Y-%m-%d %H-%M-%S')
197             res_df.to_csv(str(self.save_folder_path.get()) + '/HF first' + date_time + '.csv',
index=False)
198
199             analyser = Analyser(self.window, self.result_df, self.save_folder_path.get(), '3')
200             analyser.run()
201
202     def concat_all_csv(self):
203         concat = concat_csv.Concat(self.window)
204         concat.run()
205
206     def open_downloader(self):
207
208         download = downloader.DownLoader(self.window)
209         download.run()
210
211     def get_update(self):
212         # 获取当前文件夹地址
213         current_path = os.getcwd()
214         decision = messagebox.askokcancel(title='更新检测', message='是否检测更新？')
215         if decision:
216             # 开始执行 git pull
217             os.popen('cd ' + current_path)
218             os.popen('git reset --hard')
219             execute = os.popen('git pull')
220             for i in range(5):
221                 time.sleep(2)
222                 execute_str = execute.read()
223                 if 'file changed' in execute_str or 'files changed' in execute_str:
224                     messagebox.showinfo(title='更新成功', message='更新成功，请重新启动')
225                     self.window.destroy()
226                     # 打开新的
227                     try:
228                         sys.exit(0)
```

```
                    finally:
                        os.system(os.getcwd() + '/main.py')
                if 'Already up to date' in execute_str:
                    messagebox.showinfo(title='无可用更新', message='无可用更新')
                    return False
            messagebox.showinfo(title='失败', message='更新失败，可能无更新或多次尝试后更新失败')
        else:
            return False

    @staticmethod
    def show_update():

        message = '版本 V1.0\n 更新内容:\n'
        message += '''- 新增若干功能
    1. 新增清除缓存时，显示实际清除缓存的内存
    2. 新增 download 按钮，可以随时下载 all_report '''
        messagebox.showinfo(
            title='更新内容',
            message=message
        )

    def run(self):
        self.window.title(f'AxleHireTools : version: {self.version}')
        self.window.geometry('850x450')

        # label ending
        self.select_box = ttk.Combobox(
            master=self.window,
            textvariable=self.ending_show
        )
        self.select_box.place(x=100, y=100)
        self.select_box['values'] = ['智能分析', '普通分析']
        self.select_box.bind("<<ComboboxSelected>>", self._get_ending)

        # label boss2me
        Label(self.window, text="original file:").place(x=100, y=150)
        Entry(self.window, textvariable=self.boss2me_path, width='60').place(x=220, y=150)
        Button(self.window, text="select", command=self._get_boss2me, width='10').place(x=680, y=150)

        # label all_download
        Label(self.window, text="all report file:").place(x=100, y=200)
        Entry(self.window, textvariable=self.all_report_path, width='60').place(x=220, y=200)
        Button(self.window, text="select", command=self._get_all_report, width='10').place(x=680, y=200)

```

```
273             # label what_you_want
274             Label(self.window, text="generate path:").place(x=100, y=250)
275             Entry(self.window, textvariable=self.save_folder_path, width='60').place(x=220, y=250)
276             Button(self.window, text="select", command=self._get_save_folder_path, width='10').place(x=680,
y=250)
277
278             # label policy
279             self.policy = StringVar()
280             Label(self.window, text="policy:").place(x=100, y=300)
281             Entry(self.window, textvariable=self.policy, width='10').place(x=220, y=300)
282
283             # button
284             Button(self.window, text='next', width='10', command=self.generate_csv).place(x=680, y=350)
285             Button(self.window, text='merge', width='10', command=self.concat_all_csv).place(x=100, y=350)
286             Button(self.window, text='download', width='12', command=self.open_downloader).place(x=230,
y=350)
287             Button(self.window, text='update', width='10', command=self.get_update).place(x=380, y=350)
288             Button(self.window, text='update comments', width='15', command=self.show_update).place(x=510,
y=350)
289
290             self.window.mainloop()
291
292
293 if __name__ == '__main__':
294     main = Main()
295     main.run()
296
```

srvgg_arch.py

```
1   from basicsr.utils.registry import ARCH_REGISTRY
2   from torch import nn as nn
3   from torch.nn import functional as F
4
5
6   @ARCH_REGISTRY.register()
7   class SRVGGNetCompact(nn.Module):
8       """A compact VGG-style network structure for super-resolution.
9
10      It is a compact network structure, which performs upsampling in the last layer and no convolution is
11      conducted on the HR feature space.
12
13      Args:
14          num_in_ch (int): Channel number of inputs. Default: 3.
15          num_out_ch (int): Channel number of outputs. Default: 3.
16          num_feat (int): Channel number of intermediate features. Default: 64.
```

```
        num_conv (int): Number of convolution layers in the body network. Default: 16.
        upscale (int): Upsampling factor. Default: 4.
        act_type (str): Activation type, options: 'relu', 'prelu', 'leakyrelu'. Default: prelu.
    """

    def __init__(self, num_in_ch=3, num_out_ch=3, num_feat=64, num_conv=16, upscale=4, act_type='prelu'):
        super(SRVGGNetCompact, self).__init__()
        self.num_in_ch = num_in_ch
        self.num_out_ch = num_out_ch
        self.num_feat = num_feat
        self.num_conv = num_conv
        self.upscale = upscale
        self.act_type = act_type

        self.body = nn.ModuleList()
        # the first conv
        self.body.append(nn.Conv2d(num_in_ch, num_feat, 3, 1, 1))
        # the first activation
        if act_type == 'relu':
            activation = nn.ReLU(inplace=True)
        elif act_type == 'prelu':
            activation = nn.PReLU(num_parameters=num_feat)
        elif act_type == 'leakyrelu':
            activation = nn.LeakyReLU(negative_slope=0.1, inplace=True)
        self.body.append(activation)

        # the body structure
        for _ in range(num_conv):
            self.body.append(nn.Conv2d(num_feat, num_feat, 3, 1, 1))
            # activation
            if act_type == 'relu':
                activation = nn.ReLU(inplace=True)
            elif act_type == 'prelu':
                activation = nn.PReLU(num_parameters=num_feat)
            elif act_type == 'leakyrelu':
                activation = nn.LeakyReLU(negative_slope=0.1, inplace=True)
            self.body.append(activation)

        # the last conv
        self.body.append(nn.Conv2d(num_feat, num_out_ch * upscale * upscale, 3, 1, 1))
        # upsample
        self.upsampler = nn.PixelShuffle(upscale)

    def forward(self, x):
```

```
61          out = x
62          for i in range(0, len(self.body)):
63              out = self.body[i](out)
64
65          out = self.upsampler(out)
66          # add the nearest upsampled image, so that the network learns the residual
67          base = F.interpolate(x, scale_factor=self.upscale, mode='nearest')
68          out += base
69          return out
70
```

preprocessing_data.py

```
1   import re
2   import pandas as pd
3
4
5   def preprocessing_data(ending_df, boss2me_df, all_report_df, day):
6       if day == '4':
7           # 0. 获取文件
8           big_sheet = ending_df
9           boss2me = boss2me_df
10          report = all_report_df
11
12          # 3. initialize res_data
13          columns_list = list(big_sheet.columns)
14          columns_list.append('Earliest Dropoff Date')
15          columns_list.append('Latest Dropoff Date')
16          res_data = pd.DataFrame(columns=big_sheet.columns, dtype='object')
17
18          # 1. 将 boss "tracking code" 改为和 report "Tracking Code" 一致
19          # boss 的 tracking code 有多种可能 "tracking #" or "Tracking Number"
20          boss2me = change_title_name(boss2me, re.search(r'\'[Tt]racking(#| Number| code|
Code|_code|_Code)\'',
21                                                         str(boss2me.columns)).group(0)[1:-1], "Tracking
Code")
22
23          # 2. 合并 boss 和 report 合并为 same
24          same = pd.merge(boss2me, report, how='left', on='Tracking Code')
25
26          # 4. 将 res_data 的一些标题改为 same 的
27          # ending 与 same 的不同除 Region Code --> Region, REgion Code --> Region
28          # ending_wednesday 的 Drop off Time ending_thursday 是 Drop off time --> Dropoff Time
29          res_data = change_title_name(res_data, re.search(r'\'[Tt]racking(#| Number| code| Code)\'',
```

```
30                                                  str(res_data.columns)).group(0)[1:-1], "Tracking
Code")
31          res_data = change_title_name(res_data, re.search(r'\'((Region Code)|(REgion Code)|(region
Code)|(rEgion Code)|(Region code)|(REgion code))\'',
32                                                   str(res_data.columns)).group(0)[1:-1], "Region")
33          res_data = change_title_name(res_data, 'Assignment ID', 'Assignment Id')
34          res_data = change_title_name(res_data, re.search(r'\'(Issue)|(Reason for Complaint)\'',
str(res_data.columns)).group(0)[:-1], 'Reason for Complaint')
35          res_data = change_title_name(res_data, 'Inbound Scan Date (Linehaul)', 'Inbound Scan Date')
36          res_data = change_title_name(res_data, 'Pickup remark', 'Pickup Remark')
37          res_data = change_title_name(res_data, 'Drop off date', 'Dropoff Date')
38          res_data = change_title_name(res_data, re.search(r'\'Drop off [Tt]ime\'',
39                                                   str(res_data.columns)).group(0)[1:-1], "Dropoff
Time")
40          res_data = change_title_name(res_data, 'Drop off status', 'Dropoff Status')
41          res_data = change_title_name(res_data, 'Drop off remark', 'Dropoff Remark')
42          res_data = change_title_name(res_data, 'Requested Amount', 'Requested Credit Amount')
43
44          # 解决 Reason for complaint 问题
45          if 'Issue' in same.columns:
46              res_data['Reason for Complaint'] = same['Issue']
47
48          # 5. 遍历 same 的标题，将 same 的数据写入 same 和 res_data 共有的标题下
49          for title in same.columns:
50              if title in res_data.columns:
51                  res_data[title] = same[title]
52
53          # 6. 将 res_data 的标题重置为 ending 的标题
54          res_data.columns = big_sheet.columns
55
56          return res_data
57
58      elif day == '3':
59          # 0. 获取文件
60          big_sheet = ending_df
61          boss2me = boss2me_df
62          report = all_report_df
63
64          # 3. initialize res_data
65          columns_list = list(big_sheet.columns)
66          columns_list.append('delivery_date')
67          columns_list.append('Earliest Dropoff Time')
68          columns_list.append('Latest Dropoff Time')
69          columns_list.append('Earliest Dropoff Date')
```

```
70          columns_list.append('Latest Dropoff Date')
71          res_data = pd.DataFrame(columns=columns_list, dtype='object')
72
73          # 1. 将 boss "tracking code" 改为和 report "Tracking Code" 一致
74          # boss 的 tracking code 有多种可能 "tracking #" or "Tracking Number"
75          boss2me = change_title_name(boss2me, re.search(r'\'[Tt]racking(#| Number| code|
Code|_code|_Code|)\'',
76                                                         str(boss2me.columns)).group(0)[1:-1], "Tracking
Code")
77
78          # 2. 合并 boss 和 report 合并为 same
79          same = pd.merge(boss2me, report, how='left', on='Tracking Code')
80
81          # 4. 将 res_data 的一些标题改为 same 的
82          # ending 与 same 的不同除 Region Code --> Region, REgion Code --> Region
83          # ending_wednesday 的 Drop off Time ending_thursday 是 Drop off time --> Dropoff Time
84          res_data = change_title_name(res_data, re.search(r'\'[Tt]racking(#| Number| code| Code)\'',
85                                                           str(res_data.columns)).group(0)[1:-1], "Tracking
Code")
86          res_data = change_title_name(res_data, re.search(
87              r'\'((Region Code)|(REgion Code)|(region Code)|(rEgion Code)|(Region code)|(REgion code))\'',
88              str(res_data.columns)).group(0)[1:-1], "Region")
89          res_data = change_title_name(res_data, 'Assignment ID', 'Assignment Id')
90          res_data = change_title_name(res_data,
91                                       re.search(r'\'(Issue)|(Reason for Complaint)\'',
str(res_data.columns)).group(0)[
92                                       :-1], 'Reason for Complaint')
93          res_data = change_title_name(res_data, 'Inbound Scan Date (Linehaul)', 'Inbound Scan Date')
94          res_data = change_title_name(res_data, 'Pickup remark', 'Pickup Remark')
95          res_data = change_title_name(res_data, 'Drop off date', 'Dropoff Date')
96          res_data = change_title_name(res_data, re.search(r'\'Drop off [Tt]ime\'',
97                                                           str(res_data.columns)).group(0)[1:-1], "Dropoff
Time")
98          res_data = change_title_name(res_data, 'Drop off status', 'Dropoff Status')
99          res_data = change_title_name(res_data, 'Drop off remark', 'Dropoff Remark')
100         res_data = change_title_name(res_data, 'Requested Amount', 'Requested Credit Amount')
101
102         # 解决 Reason for complaint 问题
103         if 'Issue' in same.columns:
104             res_data['Reason for Complaint'] = same['Issue']
105
106         # 5. 遍历 same 的标题，将 same 的数据写入 same 和 res_data 共有的标题下
107         for title in same.columns:
108             if title in res_data.columns:
```

```
109                 res_data[title] = same[title]
110
111         # 6. 将 res_data 的标题重置为 ending 的标题
112         columns_list = list(big_sheet.columns)
113         columns_list.append('delivery_date')
114         columns_list.append('Earliest Dropoff Time')
115         columns_list.append('Latest Dropoff Time')
116         columns_list.append('Earliest Dropoff Date')
117         columns_list.append('Latest Dropoff Date')
118         res_data.columns = columns_list
119
120         return res_data
121
122
123 def change_title_name(pd, pd_title, pd_title_change):
124     df = pd.rename(columns={pd_title: pd_title_change})
125     return df
126
```