|  |
| --- |
| analyser.py |

1 import json

2 import os

3 import shutil

4 import pandas as pd

5 import utils.analyser\_utils as analyser\_utils

6 import requests

7 import cv2.cv2 as cv2

8 import datetime

9

10 from tkinter import ttk, Button, messagebox, StringVar, Label, Entry, Toplevel

11 from requests\_ntlm import HttpNtlmAuth

12

13

14 class Analyser(object):

15 def \_\_init\_\_(self, root, first\_analysed\_df, save\_path, day):

16 self.root = root

17 self.source\_df = first\_analysed\_df.reset\_index(drop=True)

18 self.data\_frame = self.initialize\_df(first\_analysed\_df)

19 self.window = None

20 self.temp\_window = None

21 self.url = 'https://dataorch.axlehire.com/shipments/search'

22 self.header = {

23 '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',

24 'content-type': 'application/json',

25 'cookie': r'fp=1a39e1225ea764ca9f2abf599fafba34; xtoken="dE9DbW1wYkZDI/B28g5MkirtzwljFDty7THWI75r/mVq4do8YKOJBeUtONSQ1d3L1Yb5JCAEZPTk\012FFj7LXpbKjSaV71j1S6I9zjtTLurIi1ddgqe+xsIRU84cjg0Sktu\012"'}

26 self.index = 0

27 self.save\_folder\_path = save\_path

28 self.day = day

29 self.mission\_length = len(self.data\_frame['Tracking Code'])

30

31 def initialize\_param\_dict(self):

32 *# 周四*

33 if self.day == '4':

34 param\_dict = {'Reason for Complaint': StringVar(), 'Details of Complaint': StringVar(),

35 'Tracking Code': StringVar(), 'Drop off status': StringVar(),

36 'Earliest Dropoff Time': StringVar(), 'Latest Dropoff Time': StringVar(),

37 'Scheduled Delivery Date': StringVar(), 'Shipment status': StringVar(),

38 'Inbound Scan Date 减 Scheduled Delivery Date': StringVar(),

39 'Inbound Scan Date (Linehaul)': StringVar(), 'Inbound Scan Time': StringVar(),

40 'Inbound status': StringVar(), 'Pickup Date 减 Scheduled Delivery Date': StringVar(),

41 'Pickup Date': StringVar(), 'Pickup Time': StringVar(), 'Pickup Status': StringVar(),

42 'Drop off date 减 Pickup Date': StringVar(), 'Drop off date': StringVar(),

43 'Drop off time': StringVar(), 'Drop off remark': StringVar()}

44 return param\_dict

45 *# 周三*

46 elif self.day == '3':

47 param\_dict = {'Tracking Code': StringVar(), 'Drop off status': StringVar(),

48 'Earliest Dropoff Time': StringVar(), 'Latest Dropoff Time': StringVar(),

49 'Scheduled Delivery Date': StringVar(), 'Shipment status': StringVar(),

50 'Inbound Scan Date 减 Scheduled Delivery Date': StringVar(),

51 'Inbound Scan Date (Linehaul)': StringVar(), 'Inbound Scan Time': StringVar(),

52 'Inbound status': StringVar(), 'Pickup Date 减 Scheduled Delivery Date': StringVar(),

53 'Pickup Date': StringVar(), 'Pickup Time': StringVar(), 'Pickup Status': StringVar(),

54 'Drop off date 减 Pickup Date': StringVar(), 'Drop off date': StringVar(),

55 'Drop off Time': StringVar(), 'Drop off remark': StringVar()}

56 return param\_dict

57

58 @staticmethod

59 def initialize\_df(first\_df):

60 *"""*

61  *取一切需要打标签的内容*

62  *包含：drop off status == 'Success' 的情况*

63  *Issue Category 为空的情况 （没填上）*

64  *"""*

65 first\_df['Inbound Scan Date 减 Scheduled Delivery Date'] = None

66 first\_df['Pickup Date 减 Scheduled Delivery Date'] = None

67 first\_df['Pickup Date 减 Drop off date'] = None

68 for index, row in first\_df.iterrows():

69 first\_df.loc[index, 'Inbound Scan Date 减 Scheduled Delivery Date'] \

70 = analyser\_utils.date\_subtract(first\_df.loc[index, 'Inbound Scan Date (Linehaul)'],

71 first\_df.loc[index, 'Scheduled Delivery Date'])

72 first\_df.loc[index, 'Pickup Date 减 Scheduled Delivery Date'] \

73 = analyser\_utils.date\_subtract(first\_df.loc[index, 'Pickup Date'],

74 first\_df.loc[index, 'Scheduled Delivery Date'])

75 first\_df.loc[index, 'Drop off date 减 Pickup Date'] \

76 = analyser\_utils.date\_subtract(first\_df.loc[index, 'Drop off date'],

77 first\_df.loc[index, 'Pickup Date'])

78 first\_df\_up = first\_df[first\_df['Drop off status'] == 'SUCCEEDED']

79 first\_df\_down = first\_df[pd.isna(first\_df['Issue Category'])]

80 res\_df = pd.concat([first\_df\_up, first\_df\_down]).drop\_duplicates().reset\_index(drop=True)

81 return res\_df

82

83 def run(self):

84 *"""*

85  *run 的逻辑*

86  *再此创建一个新的 tkinter 界面，并提供两个按钮，上一页，下一页*

87  *上一页 依然运行 run 函数，只不过 self.index + 1*

88  *"""*

89 *# 一堆逻辑 列出当前 index 的 dataframe，周四*

90 *# 周四*

91 if self.day == '4':

92 self.window = Toplevel(master=self.root)

93 self.param\_dict = self.initialize\_param\_dict()

94 self.temp\_window = ttk.Treeview(self.window, show='headings')

95 *# 加入各种列*

96 self.temp\_window['columns'] = ('Reason for Complaint', 'Details of Complaint', 'Tracking Code',

97 'Drop off status', 'Earliest dropoff time', 'Latest dropoff time',

98 'Scheduled Date', 'Shipment status', 'Inbound 减 Scheduled', 'Inbound Date',

99 'Inbound scan time', 'Inbound status', 'Pickup 减 Scheduled', 'Pickup Date',

100 'Pickup Time', 'Pickup status',

101 'Drop off 减 Pickup', 'Drop off date', 'Drop off time', 'Drop off remark')

102

103 self.temp\_window.column('Reason for Complaint', width=65)

104 self.temp\_window.column('Details of Complaint', width=65)

105 self.temp\_window.column('Tracking Code', width=100)

106 self.temp\_window.column('Drop off status', width=100)

107 self.temp\_window.column('Earliest dropoff time', width=50) *#*

108 self.temp\_window.column('Latest dropoff time', width=50) *#*

109 self.temp\_window.column('Scheduled Date', width=90)

110 self.temp\_window.column('Shipment status', width=80) *#*

111 self.temp\_window.column('Inbound 减 Scheduled', width=40)

112 self.temp\_window.column('Inbound Date', width=80)

113 self.temp\_window.column('Inbound scan time', width=50) *#*

114 self.temp\_window.column('Inbound status', width=80) *#*

115 self.temp\_window.column('Pickup 减 Scheduled', width=40)

116 self.temp\_window.column('Pickup Date', width=90)

117 self.temp\_window.column('Pickup Time', width=65)

118 self.temp\_window.column('Pickup status', width=65) *#*

119 self.temp\_window.column('Drop off 减 Pickup', width=40)

120 self.temp\_window.column('Drop off date', width=65)

121 self.temp\_window.column('Drop off time', width=65)

122 self.temp\_window.column('Drop off remark', width=120)

123

124 self.temp\_window.heading('Reason for Complaint', text='Reason for Complaint')

125 self.temp\_window.heading('Details of Complaint', text='Details of Complaint')

126 self.temp\_window.heading('Tracking Code', text='Tracking Code')

127 self.temp\_window.heading('Drop off status', text='Drop off status')

128 self.temp\_window.heading('Earliest dropoff time', text='Earliest dropoff time')

129 self.temp\_window.heading('Latest dropoff time', text='Latest dropoff time')

130 self.temp\_window.heading('Scheduled Date', text='Scheduled Date')

131 self.temp\_window.heading('Shipment status', text='Shipment status')

132 self.temp\_window.heading('Inbound 减 Scheduled', text='Inbound 减 Scheduled')

133 self.temp\_window.heading('Inbound Date', text='Inbound Date')

134 self.temp\_window.heading('Inbound scan time', text='Inbound scan time')

135 self.temp\_window.heading('Inbound status', text='Inbound status')

136 self.temp\_window.heading('Pickup 减 Scheduled', text='Pickup 减 Scheduled')

137 self.temp\_window.heading('Pickup Date', text='Pickup Date')

138 self.temp\_window.heading('Pickup Time', text='Pickup Time')

139 self.temp\_window.heading('Pickup status', text='Pickup status')

140 self.temp\_window.heading('Drop off 减 Pickup', text='Drop off 减 Pickup')

141 self.temp\_window.heading('Drop off date', text='Drop off date')

142 self.temp\_window.heading('Drop off time', text='Drop off time')

143 self.temp\_window.heading('Drop off remark', text='Drop off remark')

144

145 *# 初次设置值*

146 self.change\_data(data\_index=0)

147 self.temp\_window.pack(pady=20)

148

149 *# button\_next 的函数为 next\_page*

150 prev\_button = Button(self.window, text='上一页', command=self.prev\_page)

151 prev\_button.place(x=100, y=100)

152

153 next\_button = Button(self.window, text='下一页', command=self.next\_page)

154 next\_button.place(x=300, y=100)

155

156 confirm\_button = Button(self.window, text='确定', command=self.confirm)

157 confirm\_button.place(x=900, y=100)

158

159 Button(self.window, text='清除缓存', command=self.clear\_cache).place(x=1200, y=100)

160 Button(self.window, text='显示照片', command=self.show\_pic).place(x=1100, y=100)

161 Button(self.window, text='提交', command=self.hand\_in\_result).place(x=1300, y=100)

162 Button(self.window, text='打开字典', command=self.open\_dictionary).place(x=1300, y=200)

163

164 *# 显示进度*

165 self.process = StringVar()

166 Entry(self.window, width='10', textvariable=self.process).place(x=100, y=300)

167 self.process.set(str(self.index) + '/' + str(self.mission\_length))

168

169 *# 绑定按键*

170 self.window.bind('<Down>', self.next\_page)

171 self.window.bind('<Up>', self.prev\_page)

172 self.window.bind('<Return>', self.confirm)

173 self.window.bind('<s>', self.show\_pic)

174

175 *# 设置一个框，用于填对应的序号*

176 self.answer = StringVar()

177 Label(self.window, text="此条记录的问题，对应的 JJ 序号:").place(x=500, y=100)

178 Entry(self.window, width='5', textvariable=self.answer).place(x=720, y=100)

179

180 *# 显示 tracking code*

181 self.tracking\_code = StringVar()

182 Label(self.window, text="Tracking code:").place(x=600, y=200)

183 Entry(self.window, width='20', textvariable=self.tracking\_code).place(x=700, y=200)

184 self.tracking\_code.set(self.data\_frame.loc[self.index, 'Tracking Code'])

185

186 *# 显示 顾客的 notes*

187 self.client\_comment = StringVar()

188 Label(self.window, text="note:").place(x=100, y=150)

189 Entry(self.window, width='100', textvariable=self.client\_comment).place(x=150, y=150)

190 result\_dict = self.get\_dict\_from\_tracking\_code(

191 tracking\_code=self.data\_frame.loc[self.index, 'Tracking Code']

192 )

193 if 'dropoff\_note' in result\_dict['results'][0]['shipment'].keys():

194 self.client\_comment.set(result\_dict['shipment']['dropoff\_note'])

195 else:

196 self.client\_comment.set('')

197

198 *# 显示 customer id*

199 self.customer\_id = StringVar()

200 Label(self.window, text="note:").place(x=800, y=150)

201 Entry(self.window, width='100', textvariable=self.customer\_id).place(x=900, y=150)

202 if 'customer' in result\_dict['results'][0]['shipment'].keys():

203 self.customer\_id.set(result\_dict['shipment']['customer']['phone\_number'])

204 else:

205 self.customer\_id.set('')

206

207 *# 一堆逻辑 显示出图片和详细地址文字*

208 self.window.mainloop()

209

210 *# 周三*

211 elif self.day == '3':

212 self.window = Toplevel(master=self.root)

213 self.param\_dict = self.initialize\_param\_dict()

214 self.temp\_window = ttk.Treeview(self.window, show='headings')

215 *# 加入各种列*

216 self.temp\_window['columns'] = ('Tracking Code',

217 'Drop off status',

218 'Scheduled Date', 'Earliest Dropoff Time', 'Latest Dropoff Time',

219 'Shipment status', 'Inbound 减 Scheduled', 'Inbound Date',

220 'Inbound scan time', 'Inbound status', 'Pickup 减 Scheduled', 'Pickup Date',

221 'Pickup Time', 'Pickup status',

222 'Drop off 减 Pickup', 'Drop off date', 'Drop off Time', 'Drop off remark')

223

224 self.temp\_window.column('Tracking Code', width=100)

225 self.temp\_window.column('Drop off status', width=100)

226 self.temp\_window.column('Scheduled Date', width=120)

227 self.temp\_window.column('Earliest Dropoff Time', width=50) *#*

228 self.temp\_window.column('Latest Dropoff Time', width=50) *#*

229 self.temp\_window.column('Shipment status', width=120) *#*

230 self.temp\_window.column('Inbound 减 Scheduled', width=40)

231 self.temp\_window.column('Inbound Date', width=120)

232 self.temp\_window.column('Inbound scan time', width=80) *#*

233 self.temp\_window.column('Inbound status', width=80) *#*

234 self.temp\_window.column('Pickup 减 Scheduled', width=40)

235 self.temp\_window.column('Pickup Date', width=50)

236 self.temp\_window.column('Pickup Time', width=50)

237 self.temp\_window.column('Pickup status', width=120) *#*

238 self.temp\_window.column('Drop off 减 Pickup', width=40)

239 self.temp\_window.column('Drop off date', width=100)

240 self.temp\_window.column('Drop off Time', width=50)

241 self.temp\_window.column('Drop off remark', width=120)

242

243 self.temp\_window.heading('Tracking Code', text='Tracking Code')

244 self.temp\_window.heading('Drop off status', text='Drop off status')

245 self.temp\_window.heading('Scheduled Date', text='Scheduled Date')

246 self.temp\_window.heading('Earliest Dropoff Time', text='Earliest dropoff Time')

247 self.temp\_window.heading('Latest Dropoff Time', text='Latest dropoff Time')

248 self.temp\_window.heading('Shipment status', text='Shipment status')

249 self.temp\_window.heading('Inbound 减 Scheduled', text='Inbound 减 Scheduled')

250 self.temp\_window.heading('Inbound Date', text='Inbound Date')

251 self.temp\_window.heading('Inbound scan time', text='Inbound scan time')

252 self.temp\_window.heading('Inbound status', text='Inbound status')

253 self.temp\_window.heading('Pickup 减 Scheduled', text='Pickup 减 Scheduled')

254 self.temp\_window.heading('Pickup Date', text='Pickup Date')

255 self.temp\_window.heading('Pickup Time', text='Pickup Time')

256 self.temp\_window.heading('Pickup status', text='Pickup status')

257 self.temp\_window.heading('Drop off 减 Pickup', text='Drop off 减 Pickup')

258 self.temp\_window.heading('Drop off date', text='Drop off date')

259 self.temp\_window.heading('Drop off Time', text='Drop off Time')

260 self.temp\_window.heading('Drop off remark', text='Drop off remark')

261

262 *# 初次设置值*

263 self.change\_data(data\_index=0)

264 self.temp\_window.pack(pady=20)

265

266 *# button\_next 的函数为 next\_page*

267 prev\_button = Button(self.window, text='上一页', command=self.prev\_page)

268 prev\_button.place(x=100, y=100)

269

270 next\_button = Button(self.window, text='下一页', command=self.next\_page)

271 next\_button.place(x=300, y=100)

272

273 confirm\_button = Button(self.window, text='确定', command=self.confirm)

274 confirm\_button.place(x=900, y=100)

275 Button(self.window, text='清除缓存', command=self.clear\_cache).place(x=1200, y=100)

276 Button(self.window, text='显示照片', command=self.show\_pic).place(x=1100, y=100)

277 Button(self.window, text='提交', command=self.hand\_in\_result).place(x=1300, y=100)

278 Button(self.window, text='打开字典', command=self.open\_dictionary).place(x=1300, y=200)

279

280 *# 显示进度*

281 self.process = StringVar()

282 Entry(self.window, width='10', textvariable=self.process).place(x=100, y=300)

283 self.process.set(str(self.index) + '/' + str(self.mission\_length))

284

285 *# 绑定按键*

286 self.window.bind('<Down>', self.next\_page)

287 self.window.bind('<Up>', self.prev\_page)

288 self.window.bind('<Return>', self.confirm)

289 self.window.bind('<s>', self.show\_pic)

290

291 *# 设置一个框，用于填对应的序号*

292 self.answer = StringVar()

293 Label(self.window, text="此条记录的问题，对应的 JJ 序号:").place(x=500, y=100)

294 entry = Entry(self.window, width='5', textvariable=self.answer).place(x=720, y=100)

295

296 *# 显示 tracking code*

297 self.tracking\_code = StringVar()

298 Label(self.window, text="Tracking Code:").place(x=600, y=200)

299 Entry(self.window, width='20', textvariable=self.tracking\_code).place(x=700, y=200)

300 self.tracking\_code.set(self.data\_frame.loc[self.index, 'Tracking Code'])

301

302 *# 显示 顾客的 notes*

303 self.client\_comment = StringVar()

304 Label(self.window, text="note:").place(x=100, y=150)

305 Entry(self.window, width='100', textvariable=self.client\_comment).place(x=150, y=150)

306 result\_dict = self.get\_dict\_from\_tracking\_code(

307 tracking\_code=self.data\_frame.loc[self.index, 'Tracking Code']

308 )

309 if 'dropoff\_note' in result\_dict['results'][0]['shipment'].keys():

310 self.client\_comment.set(result\_dict['results'][0]['shipment']['dropoff\_note'])

311 else:

312 self.client\_comment.set('')

313

314 *# 显示 customer id*

315 self.customer\_id = StringVar()

316 Label(self.window, text="note:").place(x=800, y=150)

317 Entry(self.window, width='100', textvariable=self.customer\_id).place(x=900, y=150)

318 if 'customer' in result\_dict['results'][0]['shipment'].keys():

319 self.customer\_id.set(result\_dict['shipment']['customer']['phone\_number'])

320 else:

321 self.customer\_id.set('')

322

323 *# 进度条*

324 self.process.set(str(self.index) + '/' + str(self.mission\_length))

325

326 *# 一堆逻辑 显示出图片和详细地址文字*

327 self.window.mainloop()

328

329 def next\_page(self, event=None):

330 self.index = self.index + 1

331

332 if self.index >= len(self.data\_frame['Tracking Code']):

333 *# 到达最底下了*

334 messagebox.showinfo(title='警告', message='没有下一页了')

335 self.window.focus\_force()

336 self.index = self.index - 1

337 self.change\_data(self.index)

338

339 self.change\_data(self.index)

340

341 *# tracing code*

342 self.tracking\_code.set(self.data\_frame.loc[self.index, 'Tracking Code'])

343

344 *# dropoff note*

345 result\_dict = self.get\_dict\_from\_tracking\_code(

346 tracking\_code=self.data\_frame.loc[self.index, 'Tracking Code']

347 )

348 if 'dropoff\_note' in result\_dict['results'][0]['shipment'].keys():

349 self.client\_comment.set(result\_dict['results'][0]['shipment']['dropoff\_note'])

350 else:

351 self.client\_comment.set('')

352

353 *# customer\_id*

354 self.customer\_id = StringVar()

355 if 'customer' in result\_dict['results'][0]['shipment'].keys():

356 self.customer\_id.set(result\_dict['shipment']['customer']['phone\_number'])

357 else:

358 self.customer\_id.set('')

359

360 *# 进度条*

361 self.process.set(str(self.index) + '/' + str(self.mission\_length))

362

363 self.temp\_window.delete(f'item{self.index - 1}')

364

365 def prev\_page(self, event=None):

366 self.index = self.index - 1

367 if self.index < 0:

368 *# 到达最开始了*

369 messagebox.showinfo(title='警告', message='没有上一页了')

370 self.window.focus\_force()

371 self.index = self.index + 1

372 self.change\_data(self.index)

373

374 self.change\_data(self.index)

375

376 *# tracing code*

377 self.tracking\_code.set(self.data\_frame.loc[self.index, 'Tracking Code'])

378

379 *# dropoff note*

380 result\_dict = self.get\_dict\_from\_tracking\_code(

381 tracking\_code=self.data\_frame.loc[self.index, 'Tracking Code']

382 )

383 if 'dropoff\_note' in result\_dict['results'][0]['shipment'].keys():

384 self.client\_comment.set(result\_dict['results'][0]['shipment']['dropoff\_note'])

385 else:

386 self.client\_comment.set('')

387

388 *# customer\_id*

389 self.customer\_id = StringVar()

390 if 'customer' in result\_dict['results'][0]['shipment'].keys():

391 self.customer\_id.set(result\_dict['shipment']['customer']['phone\_number'])

392 else:

393 self.customer\_id.set('')

394

395 self.temp\_window.delete(f'item{self.index + 1}')

396

397 def change\_data(self, data\_index):

398 *# 设置 StringVar*

399 if self.day == '4':

400 self.param\_dict['Reason for Complaint'].set(self.data\_frame.loc[data\_index, 'Reason for Complaint'])

401 self.param\_dict['Details of Complaint'].set(self.data\_frame.loc[data\_index, 'Details of Complaint'])

402 self.param\_dict['Tracking Code'].set(self.data\_frame.loc[data\_index, 'Tracking Code'])

403 self.param\_dict['Drop off status'].set(self.data\_frame.loc[data\_index, 'Drop off status'])

404 self.param\_dict['Earliest Dropoff Time'].set(self.data\_frame.loc[data\_index, 'Earliest Dropoff Time'])

405 self.param\_dict['Latest Dropoff Time'].set(self.data\_frame.loc[data\_index, 'Latest Dropoff Time'])

406 self.param\_dict['Scheduled Delivery Date'].set(self.data\_frame.loc[data\_index, 'Scheduled Delivery Date'])

407 self.param\_dict['Shipment status'].set(self.data\_frame.loc[data\_index, 'Shipment status'])

408 self.param\_dict['Inbound Scan Date 减 Scheduled Delivery Date'].set(

409 self.data\_frame.loc[data\_index, 'Inbound Scan Date 减 Scheduled Delivery Date'])

410 self.param\_dict['Inbound Scan Date (Linehaul)'].set(

411 self.data\_frame.loc[data\_index, 'Inbound Scan Date (Linehaul)'])

412 self.param\_dict['Inbound Scan Time'].set(self.data\_frame.loc[data\_index, 'Inbound Scan Time'])

413 self.param\_dict['Inbound status'].set(self.data\_frame.loc[data\_index, 'Inbound status'])

414 self.param\_dict['Pickup Date 减 Scheduled Delivery Date'].set(

415 self.data\_frame.loc[data\_index, 'Pickup Date 减 Scheduled Delivery Date'])

416 self.param\_dict['Pickup Date'].set(self.data\_frame.loc[data\_index, 'Pickup Date'])

417 self.param\_dict['Pickup Time'].set(self.data\_frame.loc[data\_index, 'Pickup Time'])

418 self.param\_dict['Pickup Status'].set(self.data\_frame.loc[data\_index, 'Pickup Status'])

419 self.param\_dict['Drop off date 减 Pickup Date'].set(

420 self.data\_frame.loc[data\_index, 'Drop off date 减 Pickup Date'])

421 self.param\_dict['Drop off date'].set(self.data\_frame.loc[data\_index, 'Drop off date'])

422 self.param\_dict['Drop off time'].set(self.data\_frame.loc[data\_index, 'Drop off time'])

423 self.param\_dict['Drop off remark'].set(self.data\_frame.loc[data\_index, 'Drop off remark'])

424

425 self.temp\_window.insert('', 0, f'item{self.index}', values=(

426 self.param\_dict['Reason for Complaint'].get(),

427 self.param\_dict['Details of Complaint'].get(),

428 self.param\_dict['Tracking Code'].get(),

429 self.param\_dict['Drop off status'].get(),

430 self.param\_dict['Earliest Dropoff Time'].get(),

431 self.param\_dict['Latest Dropoff Time'].get(),

432 self.param\_dict['Scheduled Delivery Date'].get(),

433 self.param\_dict['Shipment status'].get(),

434 self.param\_dict['Inbound Scan Date 减 Scheduled Delivery Date'].get(),

435 self.param\_dict['Inbound Scan Date (Linehaul)'].get(),

436 self.param\_dict['Inbound Scan Time'].get(),

437 self.param\_dict['Inbound status'].get(),

438 self.param\_dict['Pickup Date 减 Scheduled Delivery Date'].get(),

439 self.param\_dict['Pickup Date'].get(),

440 self.param\_dict['Pickup Time'].get(),

441 self.param\_dict['Pickup Status'].get(),

442 self.param\_dict['Drop off date 减 Pickup Date'].get(),

443 self.param\_dict['Drop off date'].get(),

444 self.param\_dict['Drop off time'].get(),

445 self.param\_dict['Drop off remark'].get()

446 ))

447 self.temp\_window.pack(pady=20)

448 print(self.param\_dict['Tracking Code'].get())

449

450 elif self.day == '3':

451 self.param\_dict['Tracking Code'].set(self.data\_frame.loc[data\_index, 'Tracking Code'])

452 self.param\_dict['Drop off status'].set(self.data\_frame.loc[data\_index, 'Drop off status'])

453 self.param\_dict['Scheduled Delivery Date'].set(self.data\_frame.loc[data\_index, 'Scheduled Delivery Date'])

454 self.param\_dict['Earliest Dropoff Time'].set(self.data\_frame.loc[data\_index, 'Earliest Dropoff Time'])

455 self.param\_dict['Latest Dropoff Time'].set(self.data\_frame.loc[data\_index, 'Latest Dropoff Time'])

456 self.param\_dict['Shipment status'].set(self.data\_frame.loc[data\_index, 'Shipment status'])

457 self.param\_dict['Inbound Scan Date 减 Scheduled Delivery Date'].set(

458 self.data\_frame.loc[data\_index, 'Inbound Scan Date 减 Scheduled Delivery Date'])

459 self.param\_dict['Inbound Scan Date (Linehaul)'].set(

460 self.data\_frame.loc[data\_index, 'Inbound Scan Date (Linehaul)'])

461 self.param\_dict['Inbound Scan Time'].set(self.data\_frame.loc[data\_index, 'Inbound Scan Time'])

462 self.param\_dict['Inbound status'].set(self.data\_frame.loc[data\_index, 'Inbound status'])

463 self.param\_dict['Pickup Date 减 Scheduled Delivery Date'].set(

464 self.data\_frame.loc[data\_index, 'Pickup Date 减 Scheduled Delivery Date'])

465 self.param\_dict['Pickup Date'].set(self.data\_frame.loc[data\_index, 'Pickup Date'])

466 self.param\_dict['Pickup Time'].set(self.data\_frame.loc[data\_index, 'Pickup Time'])

467 self.param\_dict['Pickup Status'].set(self.data\_frame.loc[data\_index, 'Pickup Status'])

468 self.param\_dict['Drop off date 减 Pickup Date'].set(

469 self.data\_frame.loc[data\_index, 'Drop off date 减 Pickup Date'])

470 self.param\_dict['Drop off date'].set(self.data\_frame.loc[data\_index, 'Drop off date'])

471 self.param\_dict['Drop off Time'].set(self.data\_frame.loc[data\_index, 'Drop off Time'])

472 self.param\_dict['Drop off remark'].set(self.data\_frame.loc[data\_index, 'Drop off remark'])

473

474 self.temp\_window.insert('', 0, f'item{self.index}', values=(

475 self.param\_dict['Tracking Code'].get(),

476 self.param\_dict['Drop off status'].get(),

477 self.param\_dict['Scheduled Delivery Date'].get(),

478 self.param\_dict['Earliest Dropoff Time'].get(),

479 self.param\_dict['Latest Dropoff Time'].get(),

480 self.param\_dict['Shipment status'].get(),

481 self.param\_dict['Inbound Scan Date 减 Scheduled Delivery Date'].get(),

482 self.param\_dict['Inbound Scan Date (Linehaul)'].get(),

483 self.param\_dict['Inbound Scan Time'].get(),

484 self.param\_dict['Inbound status'].get(),

485 self.param\_dict['Pickup Date 减 Scheduled Delivery Date'].get(),

486 self.param\_dict['Pickup Date'].get(),

487 self.param\_dict['Pickup Time'].get(),

488 self.param\_dict['Pickup Status'].get(),

489 self.param\_dict['Drop off date 减 Pickup Date'].get(),

490 self.param\_dict['Drop off date'].get(),

491 self.param\_dict['Drop off Time'].get(),

492 self.param\_dict['Drop off remark'].get()

493 ))

494 self.temp\_window.pack(pady=20)

495 print(self.param\_dict['Tracking Code'].get())

496

497 @staticmethod

498 def get\_dict\_from\_tracking\_code(tracking\_code):

499 url = 'https://dataorch.axlehire.com/shipments/search'

500 header = {

501 'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36',

502 'content-type': 'application/json',

503 'cookie': r'fp=1a39e1225ea764ca9f2abf599fafba34; xtoken="dE9DbW1wYkZDI/B28g5MkirtzwljFDty7THWI75r/mVq4do8YKOJBeUtONSQ1d3L1Yb5JCAEZPTk\012FFj7LXpbKjSaV71j1S6I9zjtTLurIi1ddgqe+xsIRU84cjg0Sktu\012"'}

504 *# 生成 post 的 json\_data*

505 data\_dict = {'size': 15, 'q': tracking\_code,

506 'filters': {}, 'sorts': ['-dropoff\_earliest\_ts']}

507 json\_data = json.dumps(data\_dict)

508

509 session = requests.Session()

510 user = 'yanxia.ji'

511 password = 'Axl12345'

512 response = session.post(url=url, headers=header, data=json\_data, auth=HttpNtlmAuth(user, password))

513

514 result\_dict = json.loads(response.text)

515 return result\_dict

516

517 def show\_pic(self, event=None):

518 *# 生成 dict\_data*

519 result\_dict = self.get\_dict\_from\_tracking\_code(

520 tracking\_code=self.data\_frame.loc[self.index, 'Tracking Code']

521 )

522 *# data\_dict = {'size': 15, 'q': self.data\_frame.loc[self.index, 'Tracking Code'],*

523 *# 'filters': {}, 'sorts': ['-dropoff\_earliest\_ts']}*

524 *# json\_data = json.dumps(data\_dict)*

525

526 session = requests.Session()

527 *# user = 'yanxia.ji'*

528 *# password = 'Axl12345'*

529 *# response = session.post(url=self.url, headers=self.header, data=json\_data, auth=HttpNtlmAuth(user, password))*

530 *#*

531 *# result\_dict = json.loads(response.text)*

532 *# print(result\_dict)*

533

534 *# 如果存在照片，就显示*

535 if result\_dict['results'][0]['pod']['images'] != []:

536 *# 如果是单张照片*

537 if len(result\_dict['results'][0]['pod']['images']) == 1:

538 img\_url = result\_dict['results'][0]['pod']['images'][0]['url']

539 img\_url\_response = session.get(img\_url)

540

541 *# 写入文件到 cache*

542 with open(f'tools/img\_cache/{img\_url[-10:]}.png', 'wb') as fp:

543 fp.write(img\_url\_response.content)

544

545 if 'street2' in result\_dict['results'][0]['shipment']['dropoff\_address'].keys():

546 address\_street2 = result\_dict['results'][0]['shipment']['dropoff\_address']['street2'] + ' '

547 else:

548 address\_street2 = "" + ' '

549 address\_street = result\_dict['results'][0]['shipment']['dropoff\_address']['street'] + ' '

550 address\_city = result\_dict['results'][0]['shipment']['dropoff\_address']['city'] + ' '

551 address\_state = result\_dict['results'][0]['shipment']['dropoff\_address']['state'] + ' '

552 address\_zipcode = result\_dict['results'][0]['shipment']['dropoff\_address']['zipcode'] + ' '

553 address = address\_street2 + address\_street + address\_city + address\_state + address\_zipcode

554

555 img = cv2.imread(f'tools/img\_cache/{img\_url[-10:]}.png')

556 img = process\_image(img)

557 cv2.imshow(f"address: {address}", img)

558 cv2.waitKey()

559 *# 多张照片*

560 if len(result\_dict['results'][0]['pod']['images']) > 1:

561 imgs = []

562 for img\_url in result\_dict['results'][0]['pod']['images']:

563 img\_url\_response = session.get(img\_url['url'])

564

565 *# 写入文件到 cache*

566 with open(f'tools/img\_cache/{img\_url["url"][-10:]}.png', 'wb') as fp:

567 fp.write(img\_url\_response.content)

568 imgs.append(f'tools/img\_cache/{img\_url["url"][-10:]}.png')

569

570 if 'street2' in result\_dict['results'][0]['shipment']['dropoff\_address'].keys():

571 address\_street2 = result\_dict['results'][0]['shipment']['dropoff\_address']['street2'] + ' '

572 else:

573 address\_street2 = "" + ' '

574

575 address\_street = result\_dict['results'][0]['shipment']['dropoff\_address']['street'] + ' '

576 address\_city = result\_dict['results'][0]['shipment']['dropoff\_address']['city'] + ' '

577 address\_state = result\_dict['results'][0]['shipment']['dropoff\_address']['state'] + ' '

578 address\_zipcode = result\_dict['results'][0]['shipment']['dropoff\_address']['zipcode'] + ' '

579 address = address\_street2 + address\_street + address\_city + address\_state + address\_zipcode

580

581 *# 依次显示照片*

582 for img in imgs:

583 img = cv2.imread(img)

584 img = process\_image(img)

585 cv2.imshow(f"address: {address} have more than one pic", img)

586 cv2.waitKey()

587 else:

588 if 'street2' in result\_dict['results'][0]['shipment']['dropoff\_address'].keys():

589 address\_street2 = result\_dict['results'][0]['shipment']['dropoff\_address']['street2'] + ' '

590 else:

591 address\_street2 = " "

592 address\_street = result\_dict['results'][0]['shipment']['dropoff\_address']['street'] + ' '

593 address\_city = result\_dict['results'][0]['shipment']['dropoff\_address']['city'] + ' '

594 address\_state = result\_dict['results'][0]['shipment']['dropoff\_address']['state'] + ' '

595 address\_zipcode = result\_dict['results'][0]['shipment']['dropoff\_address']['zipcode'] + ' '

596 address = address\_street2 + address\_street + address\_city + address\_state + address\_zipcode

597

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

599 self.window.focus\_force()

600

601 def clear\_cache(self):

602 if not os.path.exists('tools/img\_cache'):

603 os.mkdir('tools/img\_cache')

604 del\_list = os.listdir('tools/img\_cache')

605 if len(del\_list) == 0:

606 messagebox.showinfo(title='清除失败', message='无缓存')

607 return

608 file\_size\_sum = 0

609 for f in del\_list:

610 file\_path = os.path.join('tools/img\_cache', f)

611 if os.path.isfile(file\_path):

612 file\_size\_sum += self.get\_filesize(file\_path)

613 os.remove(file\_path)

614 elif os.path.isdir(file\_path):

615 shutil.rmtree(file\_path)

616 messagebox.showinfo(title='清除成功', message=f'清除缓存共 {round(file\_size\_sum, 1)}mb')

617

618 @staticmethod

619 def get\_filesize(file\_path):

620 file\_size = os.path.getsize(file\_path)

621 file\_size = file\_size / float(1024 \* 1024)

622 return round(file\_size, 2)

623

624 def confirm(self, event=None):

625 print(self.answer)

626 print(self.answer.get())

627 answer\_index = self.answer.get()

628 print('answer\_index', int(answer\_index))

629 analyser\_utils.copy\_reason(

630 data\_frame\_row=self.data\_frame.iloc[self.index: self.index + 1, :],

631 index=int(answer\_index)

632 )

633 messagebox.showinfo(title='确定', message='您的输入已写入')

634 self.window.focus\_force()

635

636 def hand\_in\_result(self):

637 *# 生成 csv*

638 date\_time = datetime.datetime.now().strftime('%Y-%m-%d %H-%M-%S')

639 path = str(self.save\_folder\_path) + '/最终版' + date\_time + '.csv'

640

641 res\_df = self.write\_in()

642

643 *# 周三的需要改动列名*

644 if self.day == '3':

645 res\_df.rename(columns={'AH Assessment': 'HF Reason Code'}, inplace=True)

646 *# 这里 res\_df 中的五列将带 x 的写回去，并 drop 掉*

647 res\_df.to\_csv(path, index=False)

648

649 messagebox.showinfo(title='成功', message=f'已生成 {path}')

650

651 def write\_in(self):

652 def get\_index(tracking\_code, source\_df):

653 return source\_df[source\_df['Tracking Code'] == tracking\_code].index

654

655 data\_frame = self.data\_frame.copy()

656 source\_df = self.source\_df.copy()

657 *# 把 data\_frame 根据相同的 tracking code 将五列写入 source\_df 返回 res\_df*

658 for index, row in data\_frame.iterrows():

659 source\_df.loc[get\_index(data\_frame.loc[index, 'Tracking Code'], source\_df),

660 'Issue Category'] = data\_frame.loc[index, 'Issue Category']

661 source\_df.loc[get\_index(data\_frame.loc[index, 'Tracking Code'], source\_df),

662 'Delivery Comments'] = data\_frame.loc[index, 'Delivery Comments']

663 source\_df.loc[get\_index(data\_frame.loc[index, 'Tracking Code'], source\_df),

664 'AH Assessment'] = data\_frame.loc[index, 'AH Assessment']

665 source\_df.loc[get\_index(data\_frame.loc[index, 'Tracking Code'], source\_df),

666 'POD Quality'] = data\_frame.loc[index, 'POD Quality']

667 source\_df.loc[get\_index(data\_frame.loc[index, 'Tracking Code'], source\_df),

668 'POD Valid?'] = data\_frame.loc[index, 'POD Valid?']

669 return source\_df

670

671 @staticmethod

672 def open\_dictionary():

673 os.system(os.getcwd() + '/tools/files/dictionary.xlsx')

674

675

676 def process\_image(img):

677 min\_side = 768

678 size = img.shape

679 h, w = size[0], size[1]

680 *# 长边缩放为min\_side*

681 scale = max(w, h) / float(min\_side)

682 new\_w, new\_h = int(w / scale), int(h / scale)

683 resize\_img = cv2.resize(img, (new\_w, new\_h))

684 *# 填充至min\_side \* min\_side*

685 if new\_w % 2 != 0 and new\_h % 2 == 0:

686 top, bottom, left, right = (min\_side - new\_h) / 2, (min\_side - new\_h) / 2, (min\_side - new\_w) / 2 + 1, (

687 min\_side - new\_w) / 2

688 elif new\_h % 2 != 0 and new\_w % 2 == 0:

689 top, bottom, left, right = (min\_side - new\_h) / 2 + 1, (min\_side - new\_h) / 2, (min\_side - new\_w) / 2, (

690 min\_side - new\_w) / 2

691 elif new\_h % 2 == 0 and new\_w % 2 == 0:

692 top, bottom, left, right = (min\_side - new\_h) / 2, (min\_side - new\_h) / 2, (min\_side - new\_w) / 2, (

693 min\_side - new\_w) / 2

694 else:

695 top, bottom, left, right = (min\_side - new\_h) / 2 + 1, (min\_side - new\_h) / 2, (min\_side - new\_w) / 2 + 1, (

696 min\_side - new\_w) / 2

697 pad\_img = cv2.copyMakeBorder(resize\_img, int(top), int(bottom), int(left), int(right), cv2.BORDER\_CONSTANT,

698 value=[0, 0, 0]) *# 从图像边界向上,下,左,右扩的像素数目*

699 return pad\_img

700

701

702 class Thursday(object):

703 def \_\_init\_\_(self, init\_df, policy):

704 self.init\_df = init\_df

705 self.policy = policy

706

707 def analyse(self):

708 res\_data = self.init\_df.copy()

709 result = pd.DataFrame(columns=res\_data.columns)

710 for index, row in self.init\_df.iterrows():

711 *# 修改 Scheduled Delivery Date 成为 %Y-%m-%d*

712 temp = analyser\_utils.change\_Scheduled\_Delivery\_Date(res\_data.iloc[index: index + 1, :])

713 *# 填入 week √*

714 temp = analyser\_utils.get\_week\_num(temp)

715 *# 修改时区*

716 *# temp = analyser\_utils.data\_frame\_row\_time\_change(temp)*

717 res\_data.iloc[index: index + 1, :] = analyser\_utils.get\_status(temp, day='4', policy=self.policy)

718 result = pd.concat([result, temp])

719 result['Updated Reason Code'] = result['AH Assessment']

720 return result

721

722

723 class Wednesday(object):

724 def \_\_init\_\_(self, init\_df, policy):

725 self.init\_df = init\_df

726 self.policy = policy

727

728 def analyse(self):

729 res\_data = self.init\_df.copy()

730 res\_data.rename(columns={'HF Reason Code': 'AH Assessment', 'POD Qaulity': 'POD Quality'}, inplace=True)

731 result = pd.DataFrame(columns=res\_data.columns)

732 for index, row in self.init\_df.iterrows():

733 *# 修改 Scheduled Delivery Date 成为 %Y-%m-%d*

734 temp = analyser\_utils.change\_Scheduled\_Delivery\_Date(res\_data.iloc[index: index + 1, :])

735 *# 填入 week √*

736 temp = analyser\_utils.get\_week\_num(temp)

737 *# 修改时区*

738 *# temp = analyser\_utils.data\_frame\_row\_time\_change(temp)*

739 *# 分析 status*

740 res\_data.iloc[index: index + 1, :] = analyser\_utils.get\_status(temp, day='3', policy=self.policy)

741 result = pd.concat([result, temp])

742 result['Updated Reason Code'] = result['AH Assessment']

743 return result

744

|  |
| --- |
| analyser\_utils.py |

1 import os

2 import datetime

3 import pandas as pd

4

5

6 DICT\_DF = pd.read\_excel(os.getcwd() + '/utils/files/dictionary.xlsx')

7

8

9 def is\_mouth\_day\_year(date):

10 try:

11 datetime.datetime.strptime(date, "%m/%d/%Y")

12 return True

13 except:

14 return False

15

16

17 def get\_week\_num(data\_frame\_row):

18 date\_str = data\_frame\_row['Scheduled Delivery Date'].values[0]

19 if pd.isna(date\_str):

20

21 return data\_frame\_row

22 *# 如果 scheduled date 是月日年*

23 if is\_mouth\_day\_year(date\_str):

24 res\_date = datetime.datetime.strptime(date\_str, '%m/%d/%Y')

25 data\_frame\_row['Week#'] = [res\_date.isocalendar()[1]]

26 return data\_frame\_row

27 else:

28 res\_date = datetime.datetime.strptime(date\_str, '%Y-%m-%d')

29 *# res\_date = datetime.datetime.strptime(date\_str, '%Y-%m-%d %H:%M:%S')*

30 data\_frame\_row['Week#'] = [res\_date.isocalendar()[1]]

31 return data\_frame\_row

32

33

34

35 def nan\_to\_none(x):

36 if str(x) == 'nan' or pd.isna(x):

37 return ''

38 return x

39

40

41 def copy\_reason(data\_frame\_row, index):

42 *# print('values', data\_frame\_row['POD Valid?'].values[0])*

43 *# print('values', data\_frame\_row['Issue Category'].values[0])*

44 copy\_df = DICT\_DF[DICT\_DF.index == index]

45 *# 如果全是空的，直接复制*

46 if pd.isna(data\_frame\_row['POD Valid?'].values[0]) and pd.isna(data\_frame\_row['POD Quality'].values[0]) and \

47 pd.isna(data\_frame\_row['Issue Category'].values[0]) and pd.isna(data\_frame\_row['Delivery Comments'].values[0]) and \

48 pd.isna(data\_frame\_row['AH Assessment'].values[0]):

49 *# print('全是空的')*

50 data\_frame\_row['POD Valid?'] = [nan\_to\_none(copy\_df['POD Valid?'].values[0])]

51 data\_frame\_row['POD Quality'] = [nan\_to\_none(copy\_df['POD Quality'].values[0])]

52 data\_frame\_row['Issue Category'] = copy\_df['Issue Category'].values

53 data\_frame\_row['Delivery Comments'] = copy\_df['Delivery Comments'].values

54 data\_frame\_row['AH Assessment'] = copy\_df['AH Assessment'].values

55 *# print('写进去了')*

56 return data\_frame\_row

57 *# 如果不是空的，加一个 / 再将内容附着上*

58 else:

59 *# print('不是空的')*

60 data\_frame\_row['POD Valid?'] = [nan\_to\_none(str(copy\_df['POD Valid?'].values[0]))]

61 data\_frame\_row['POD Quality'] = [nan\_to\_none(str(copy\_df['POD Quality'].values[0]))]

62 data\_frame\_row['Issue Category'] = [str(data\_frame\_row['Issue Category'].values[0]) + '/' + str(copy\_df['Issue Category'].values[0])]

63 data\_frame\_row['Delivery Comments'] = [str(data\_frame\_row['Delivery Comments'].values[0]) + '/' + str(copy\_df['Delivery Comments'].values[0])]

64 data\_frame\_row['AH Assessment'] = [str(data\_frame\_row['AH Assessment'].values[0]) + '/' + str(copy\_df['AH Assessment'].values[0])]

65 *# print('附着进去了')*

66 return data\_frame\_row

67

68

69 def get\_pickup\_and\_delivery\_status(data\_frame\_row, day, policy):

70 pickup\_diff = date\_subtract(data\_frame\_row['Pickup Date'].values[0],

71 data\_frame\_row['Scheduled Delivery Date'].values[0])

72

73 *# 如果 pick 当天送达*

74 if pickup\_diff == 0:

75 *# 如果 pick 当天晚于 12*

76 if time\_upper\_than(data\_frame\_row['Pickup Time'].values[0], '12:00', 0):

77 *# print('Pickup Time', data\_frame\_row['Pickup Time'].values[0])*

78 data\_frame\_row['Pickup Comments'] = ['Pickup after 12pm']

79

80 delivery\_diff = date\_subtract(data\_frame\_row['Drop off date'].values[0],

81 data\_frame\_row['Pickup Date'].values[0])

82 *# 判断 delivery 当天*

83 if delivery\_diff == 0:

84 if time\_upper\_than(data\_frame\_row['Drop off time'].values[0],

85 data\_frame\_row['Latest Dropoff Time'].values[0], policy):

86 data\_frame\_row = copy\_reason(data\_frame\_row, 118)

87 return data\_frame\_row

88 *# 配送晚于 2 天*

89 else:

90 data\_frame\_row = copy\_reason(data\_frame\_row, 119)

91

92 if delivery\_diff == 1:

93 return data\_frame\_row

94 else:

95 if delivery\_diff < 2:

96 return data\_frame\_row

97 else:

98 *# data\_frame\_row['Delivery Comments'] = [*

99 *# f'pickup ok but delivery late for {delivery\_diff} days']*

100 data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row,

101 f'pickup ok but delivery late for {delivery\_diff} days')

102 return data\_frame\_row

103 *# 如果 pick 早于 12 点*

104 else:

105 delivery\_diff = date\_subtract(data\_frame\_row['Drop off date'].values[0],

106 data\_frame\_row['Pickup Date'].values[0])

107 *# 判断 delivery 当天*

108 if delivery\_diff == 0:

109 if time\_upper\_than(data\_frame\_row['Drop off time'].values[0],

110 data\_frame\_row['Latest Dropoff Time'].values[0], policy):

111 data\_frame\_row = copy\_reason(data\_frame\_row, 118)

112 return data\_frame\_row

113 *# 配送晚于 2 天*

114 else:

115 data\_frame\_row = copy\_reason(data\_frame\_row, 119)

116

117 if delivery\_diff == 1:

118 return data\_frame\_row

119 else:

120 if delivery\_diff < 2:

121 return data\_frame\_row

122 else:

123 *# data\_frame\_row['Delivery Comments'] = [*

124 *# f'pickup ok but delivery late for {delivery\_diff} days']*

125 data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row,

126 f'pickup ok but delivery late for {delivery\_diff} days')

127 return data\_frame\_row

128 return data\_frame\_row

129

130 *# 如果 pick 晚了 n 天*

131 if pickup\_diff > 1:

132 *# 对于周三*

133 if day == '3':

134 data\_frame\_row = copy\_reason(data\_frame\_row, 41)

135 if pickup\_diff == 1:

136 *# data\_frame\_row['Delivery Comments'] = [*

137 *# f'Inbound ontime but outbound late for 1 day']*

138 data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row,

139 f'Inbound ontime but outbound late for 1 day')

140 data\_frame\_row['Pickup Comments'] = [

141 f'Inbound ontime but outbound late for 1 day']

142

143 delivery\_diff = date\_subtract(data\_frame\_row['Drop off date'].values[0],

144 data\_frame\_row['Pickup Date'].values[0])

145 *# pickup 晚了一天，delivery 当天*

146 if delivery\_diff == 0:

147 *# 当天晚了*

148 if time\_upper\_than(data\_frame\_row['Drop off time'].values[0],

149 data\_frame\_row['Latest Dropoff Time'].values[0], policy):

150 data\_frame\_row = copy\_reason(data\_frame\_row, 118)

151

152 data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row,

153 f'pickup late for {pickup\_diff} day and delivery late for same day')

154

155 return data\_frame\_row

156 *# pickup 一天， delivery 多天*

157 else:

158 data\_frame\_row = copy\_reason(data\_frame\_row, 119)

159 if delivery\_diff == 1:

160 data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row,

161 f'pickup late for 1 day and delivery late for 1 day')

162 return data\_frame\_row

163 else:

164 data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row,

165 f'pickup late for 1 day and delivery late for {delivery\_diff} days')

166 return data\_frame\_row

167 return data\_frame\_row

168

169 *# pick up 晚于 1 天，看 delivery*

170 elif pickup\_diff > 1:

171 *# data\_frame\_row['Delivery Comments'] = [*

172 *# f'Inbound ontime but outbound late for {pickup\_diff} days']*

173 data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row,

174 f'Inbound ontime but outbound late for {pickup\_diff} days')

175 data\_frame\_row['Pickup Comments'] = [

176 f'Inbound ontime but outbound late for {pickup\_diff} days']

177 return data\_frame\_row

178 return data\_frame\_row

179

180 *# 周四*

181 if day == '4':

182 data\_frame\_row = copy\_reason(data\_frame\_row, 0)

183 if pickup\_diff == 1:

184 *# data\_frame\_row['Delivery Comments'] = [*

185 *# f'Inbound ontime but outbound late for 1 day']*

186 data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row,

187 f'Inbound ontime but outbound late for 1 day')

188 data\_frame\_row['Pickup Comments'] = [

189 f'Inbound ontime but outbound late for 1 day']

190

191 delivery\_diff = date\_subtract(data\_frame\_row['Drop off date'].values[0],

192 data\_frame\_row['Pickup Date'].values[0])

193 *# pickup 晚了一天，delivery 当天*

194 if delivery\_diff == 0:

195 *# 当天晚了*

196 if time\_upper\_than(data\_frame\_row['Drop off time'].values[0],

197 data\_frame\_row['Latest Dropoff Time'].values[0], policy):

198 data\_frame\_row = copy\_reason(data\_frame\_row, 118)

199

200 data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row,

201 f'pickup late for {pickup\_diff} day and delivery late for same day')

202

203 return data\_frame\_row

204

205 *# pickup 一天， delivery 多天*

206 else:

207 data\_frame\_row = copy\_reason(data\_frame\_row, 119)

208 if delivery\_diff == 1:

209 data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row,

210 f'pickup late for 1 day and delivery late for 1 day')

211 return data\_frame\_row

212 else:

213 data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row,

214 f'pickup late for 1 day and delivery late for {delivery\_diff} days')

215 return data\_frame\_row

216 return data\_frame\_row

217

218 *# pick up 晚于 1 天，看 delivery*

219 elif pickup\_diff > 1:

220 *# data\_frame\_row['Delivery Comments'] = [*

221 *# f'Inbound ontime but outbound late for {pickup\_diff} days']*

222 data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row,

223 f'Inbound ontime but outbound late for {pickup\_diff} days')

224 data\_frame\_row['Pickup Comments'] = [

225 f'Inbound ontime but outbound late for {pickup\_diff} days']

226 return data\_frame\_row

227 return data\_frame\_row

228

229 *# pickup 没晚*

230 else:

231 delivery\_diff = date\_subtract(data\_frame\_row['Drop off date'].values[0],

232 data\_frame\_row['Pickup Date'].values[0])

233 if delivery\_diff == 0:

234 if time\_upper\_than(data\_frame\_row['Drop off time'].values[0],

235 data\_frame\_row['Latest Dropoff Time'].values[0], policy):

236 data\_frame\_row = copy\_reason(data\_frame\_row, 118)

237 return data\_frame\_row

238 else:

239 data\_frame\_row = copy\_reason(data\_frame\_row, 119)

240 if delivery\_diff == 1:

241 *# data\_frame\_row['Delivery Comments'] = [*

242 *# f'pickup ok but delivery late for {delivery\_diff} day']*

243 *# data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row,*

244 *# f'pickup ok but delivery late for {delivery\_diff} day')*

245 return data\_frame\_row

246 else:

247 if delivery\_diff < 2:

248 return data\_frame\_row

249 *# data\_frame\_row['Delivery Comments'] = [*

250 *# f'pickup ok but delivery late for {delivery\_diff} days']*

251 else:

252 data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row,

253 f'pickup ok but delivery late for {delivery\_diff} days')

254 return data\_frame\_row

255 return data\_frame\_row

256

257

258 def get\_status(data\_frame\_row, day, policy):

259 *# 判断 shipment status 完了*

260 print(data\_frame\_row['Shipment status'])

261 if 'GEOCODED'.lower() in str(data\_frame\_row['Shipment status'].values[0]).lower():

262 *# 这里还要继续分析，但是比较复杂，不过多写了*

263 data\_frame\_row = copy\_reason(data\_frame\_row, 29)

264 return data\_frame\_row

265 if 'CANCELLED\_BEFORE\_PICKUP'.lower() in str(data\_frame\_row['Shipment status'].values[0]).lower():

266 data\_frame\_row = copy\_reason(data\_frame\_row, 20)

267 return data\_frame\_row

268 if 'GEOCODE\_FAILED'.lower() in str(data\_frame\_row['Shipment status'].values[0]).lower():

269 data\_frame\_row = copy\_reason(data\_frame\_row, 22)

270 return data\_frame\_row

271

272 *# 判断 Inbound status Missing*

273 if 'MISSING'.lower() in str(data\_frame\_row['Inbound status'].values[0]).lower():

274 data\_frame\_row = copy\_reason(data\_frame\_row, 25)

275 return data\_frame\_row

276 if 'DAMAGED'.lower() in str(data\_frame\_row['Inbound status'].values[0]).lower():

277 data\_frame\_row = copy\_reason(data\_frame\_row, 26)

278 return data\_frame\_row

279

280 *# 开始逐一检查 Drop off status*

281 if data\_frame\_row['Drop off status'].values[0] == 'DISCARDED':

282 if "Damaged".lower() in str(data\_frame\_row['Drop off remark'].values[0]).lower():

283 if day == '4':

284 data\_frame\_row = copy\_reason(data\_frame\_row, 5)

285 data\_frame\_row['Pickup Comments'] = ['Inbound ok but pickup damaged']

286 return data\_frame\_row

287 if day == '3':

288 data\_frame\_row = copy\_reason(data\_frame\_row, 34)

289 data\_frame\_row['Pickup Comments'] = ['Inbound ok but pickup damaged']

290 return data\_frame\_row

291 if 'RECEIVED\_DAMAGED'.lower() in str(data\_frame\_row['Drop off remark'].values[0]).lower():

292 data\_frame\_row = copy\_reason(data\_frame\_row, 26)

293 return data\_frame\_row

294 if 'discard'.lower() in str(data\_frame\_row['Drop off remark'].values[0]).lower():

295 if day == '3':

296 data\_frame\_row = copy\_reason(data\_frame\_row, 37)

297 data\_frame\_row['Pickup Comments'] = ['Inbound ok but pickup failed']

298 return data\_frame\_row

299 if day == '4':

300 data\_frame\_row = copy\_reason(data\_frame\_row, 3)

301 data\_frame\_row['Pickup Comments'] = ['Inbound ok but pickup failed']

302 return data\_frame\_row

303 if "Missing".lower() in str(data\_frame\_row['Drop off remark'].values[0]).lower():

304 if day == '3':

305 data\_frame\_row = copy\_reason(data\_frame\_row, 35)

306 data\_frame\_row['Pickup Comments'] = ['Inbound ok but pickup failed']

307 return data\_frame\_row

308 if day == '4':

309 data\_frame\_row = copy\_reason(data\_frame\_row, 4)

310 data\_frame\_row['Pickup Comments'] = ['Inbound ok but pickup failed']

311 return data\_frame\_row

312 if pd.isna(data\_frame\_row['Drop off remark'].values[0]):

313 data\_frame\_row = copy\_reason(data\_frame\_row, 52)

314 return data\_frame\_row

315

316 if data\_frame\_row['Drop off status'].values[0] is None:

317 if 'missing by inbound' in str(data\_frame\_row['Drop off remark'].values[0]).lower():

318 data\_frame\_row = copy\_reason(data\_frame\_row, 25)

319 return data\_frame\_row

320 if 'missing by outbound' in str(data\_frame\_row['Drop off remark'].values[0]).lower():

321 if day == '3':

322 data\_frame\_row = copy\_reason(data\_frame\_row, 35)

323 return data\_frame\_row

324 if day == '4':

325 data\_frame\_row = copy\_reason(data\_frame\_row, 4)

326 return data\_frame\_row

327 *# if data\_frame\_row['Pickup Status'].values[0] == 'MISSING':*

328 *# if day == '4':*

329 *# data\_frame\_row = copy\_reason(data\_frame\_row, 4)*

330 *# return data\_frame\_row*

331 *# if day == '3':*

332 *# data\_frame\_row = copy\_reason(data\_frame\_row, 35)*

333 *# return data\_frame\_row*

334

335 if data\_frame\_row['Drop off status'].values[0] == 'EN\_ROUTE':

336 if data\_frame\_row['Pickup Status'].values[0] == 'SUCCEEDED':

337 data\_frame\_row = copy\_reason(data\_frame\_row, 52)

338 return data\_frame\_row

339

340 if data\_frame\_row['Drop off status'].values[0] == 'PENDING':

341 if data\_frame\_row['Pickup Status'].values[0] == 'SUCCEEDED':

342 data\_frame\_row = copy\_reason(data\_frame\_row, 52)

343 return data\_frame\_row

344 if data\_frame\_row['Pickup Status'].values[0] == 'FAILED' or \

345 data\_frame\_row['Pickup Status'].values[0] == 'PENDING':

346 if day == '4':

347 data\_frame\_row = copy\_reason(data\_frame\_row, 3)

348 return data\_frame\_row

349 elif day == '3':

350 data\_frame\_row = copy\_reason(data\_frame\_row, 31)

351 return data\_frame\_row

352

353 if data\_frame\_row['Drop off status'].values[0] == 'FAILED':

354 *# ok*

355 if pd.isna(data\_frame\_row['Drop off remark'].values[0]):

356 data\_frame\_row = copy\_reason(data\_frame\_row, 52)

357 return data\_frame\_row

358 *# 如果 remark 不是空*

359 if isinstance(data\_frame\_row['Drop off remark'].values[0], str):

360 *# ok*

361 if 'out of cold chain'.lower() in data\_frame\_row['Drop off remark'].values[0].lower():

362 data\_frame\_row = copy\_reason(data\_frame\_row, 51)

363 return data\_frame\_row

364 *# ok*

365 if 'missing'.lower() in data\_frame\_row['Drop off remark'].values[0].lower():

366 data\_frame\_row = copy\_reason(data\_frame\_row, 52)

367 return data\_frame\_row

368 *# ok*

369 if 'damaged'.lower() in data\_frame\_row['Drop off remark'].values[0].lower():

370 data\_frame\_row = copy\_reason(data\_frame\_row, 46)

371 return data\_frame\_row

372 *# ok*

373 *# if 'wrong'.lower() in data\_frame\_row['Drop off remark'].values[0].lower():*

374 *# data\_frame\_row = copy\_reason(data\_frame\_row, 14)*

375 *# return data\_frame\_row*

376 *# ok*

377 if 'no access'.lower() in data\_frame\_row['Drop off remark'].values[0].lower():

378 data\_frame\_row = copy\_reason(data\_frame\_row, 14)

379 return data\_frame\_row

380 *# ok*

381 if 'access code'.lower() in data\_frame\_row['Drop off remark'].values[0].lower():

382 data\_frame\_row = copy\_reason(data\_frame\_row, 14)

383 return data\_frame\_row

384 *# ok*

385 if 'no answer'.lower() in data\_frame\_row['Drop off remark'].values[0].lower():

386 data\_frame\_row = copy\_reason(data\_frame\_row, 14)

387 return data\_frame\_row

388 *# ok*

389 if 'can\'t be reach'.lower() in data\_frame\_row['Drop off remark'].values[0].lower():

390 data\_frame\_row = copy\_reason(data\_frame\_row, 13)

391 return data\_frame\_row

392 *# ok*

393 if 'cant be reach'.lower() in data\_frame\_row['Drop off remark'].values[0].lower():

394 data\_frame\_row = copy\_reason(data\_frame\_row, 13)

395 return data\_frame\_row

396 *# ok*

397 if 'closed'.lower() in data\_frame\_row['Drop off remark'].values[0].lower():

398 data\_frame\_row = copy\_reason(data\_frame\_row, 14)

399 return data\_frame\_row

400 *# ok*

401 if 'requested redelivery'.lower() in data\_frame\_row['Drop off remark'].values[0].lower():

402 data\_frame\_row = copy\_reason(data\_frame\_row, 23)

403 return data\_frame\_row

404 *# ok*

405 if 'redelivery requested'.lower() in data\_frame\_row['Drop off remark'].values[0].lower():

406 data\_frame\_row = copy\_reason(data\_frame\_row, 23)

407 return data\_frame\_row

408 *# ok*

409 if 'Cancel'.lower() in data\_frame\_row['Drop off remark'].values[0].lower():

410 data\_frame\_row = copy\_reason(data\_frame\_row, 20)

411 return data\_frame\_row

412 *# ok*

413 if 'refused'.lower() in data\_frame\_row['Drop off remark'].values[0].lower():

414 data\_frame\_row = copy\_reason(data\_frame\_row, 19)

415

416 *# 如果是 SUCCEEDED 状态*

417 if data\_frame\_row['Drop off status'].values[0] == 'SUCCEEDED':

418 *# 先查看一些明显的问题*

419 if 'no access'.lower() in str(data\_frame\_row['Drop off remark'].values[0]).lower():

420 data\_frame\_row = copy\_reason(data\_frame\_row, 14)

421 elif 'no answer'.lower() in str(data\_frame\_row['Drop off remark'].values[0]).lower():

422 data\_frame\_row = copy\_reason(data\_frame\_row, 14)

423 elif 'no code'.lower() in str(data\_frame\_row['Drop off remark'].values[0]).lower():

424 data\_frame\_row = copy\_reason(data\_frame\_row, 14)

425

426 *# 再看时间差，附着到之前的结果*

427 inbound\_diff = date\_subtract(data\_frame\_row['Inbound Scan Date (Linehaul)'].values[0],

428 data\_frame\_row['Scheduled Delivery Date'].values[0])

429 *# 如果 inbound\_diff 等于 0*

430 if inbound\_diff == 0:

431 *# 如果 inbound 当天晚于 12 点*

432 if time\_upper\_than(data\_frame\_row['Inbound Scan Time'].values[0], '12:00', 0):

433 data\_frame\_row['Inbound Comments'] = ['Inbound late']

434 data\_frame\_row = copy\_reason(data\_frame\_row, 24)

435 return data\_frame\_row

436 *# 如果 inbound 当天早于 12 点*

437 else:

438 get\_pickup\_and\_delivery\_status(data\_frame\_row, day, policy)

439

440 *# 如果 inbound\_diff 大于一天*

441 elif inbound\_diff > 0:

442 data\_frame\_row['Inbound Comments'] = ['Inbound late']

443 data\_frame\_row = copy\_reason(data\_frame\_row, 24)

444 *# # 看 pick 减 inbound 的日子*

445 *# pickup\_diff = date\_subtract(data\_frame\_row['Pickup Date'].values[0],*

446 *# data\_frame\_row['Scheduled Delivery Date'].values[0]) - inbound\_diff*

447

448 *# inbound late 了，不看别的了*

449 *# if pickup\_diff > 0:*

450 *# if day == '3':*

451 *# data\_frame\_row = copy\_reason(data\_frame\_row, 41)*

452 *# if pickup\_diff == 1:*

453 *# # data\_frame\_row['Delivery Comments'] = [*

454 *# # f'Inbound ontime but outbound late for 1 day']*

455 *# data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row, f'Inbound late and outbound late for 1 day')*

456 *# data\_frame\_row['Pickup Comments'] = [*

457 *# f'Inbound late and outbound late for 1 day']*

458 *# return data\_frame\_row*

459 *# else:*

460 *# # data\_frame\_row['Delivery Comments'] = [*

461 *# # f'Inbound ontime but outbound late for {pickup\_diff} days']*

462 *# data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row, f'Inbound late and outbound late for {pickup\_diff} days')*

463 *# data\_frame\_row['Pickup Comments'] = [*

464 *# f'Inbound late and outbound late for {pickup\_diff} days']*

465 *# return data\_frame\_row*

466 *# if day == '4':*

467 *# data\_frame\_row = copy\_reason(data\_frame\_row, 0)*

468 *# if pickup\_diff == 1:*

469 *# # data\_frame\_row['Delivery Comments'] = [*

470 *# # f'Inbound ontime but outbound late for 1 day']*

471 *# data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row, f'Inbound late and outbound late for 1 day')*

472 *# data\_frame\_row['Pickup Comments'] = [*

473 *# f'Inbound late and outbound late for 1 day']*

474 *# return data\_frame\_row*

475 *# else:*

476 *# # data\_frame\_row['Delivery Comments'] = [*

477 *# # f'Inbound ontime but outbound late for {pickup\_diff} days']*

478 *# data\_frame\_row = write\_in\_delivery\_comments(data\_frame\_row, f'Inbound late and outbound late for {pickup\_diff} days')*

479 *# data\_frame\_row['Pickup Comments'] = [*

480 *# f'Inbound late and outbound late for {pickup\_diff} days']*

481 *# return data\_frame\_row*

482 *# return data\_frame\_row*

483 return data\_frame\_row

484

485 *# 当 Inbound 没 late*

486 else:

487 get\_pickup\_and\_delivery\_status(data\_frame\_row, day, policy)

488

489 return data\_frame\_row

490

491

492 def date\_subtract(compared\_date, schedule\_date):

493 if pd.isna(compared\_date) or str(compared\_date) == 'nan':

494 return -100

495 if pd.isna(schedule\_date) or str(schedule\_date) == 'nan':

496 return -100

497 else:

498 try:

499 print(compared\_date, schedule\_date)

500 compared\_date = datetime.datetime.strptime(compared\_date, '%Y-%m-%d')

501 *# 如果 scheduled date 是月日年*

502 if is\_mouth\_day\_year(schedule\_date):

503 schedule\_date = datetime.datetime.strptime(schedule\_date, '%m/%d/%Y')

504 else:

505 *# print('卧槽尼玛', schedule\_date)*

506 schedule\_date = datetime.datetime.strptime(schedule\_date, '%Y-%m-%d')

507 return (compared\_date - schedule\_date).days

508 except ValueError:

509 return -100

510

511

512 def time\_upper\_than(time\_str, upper, policy):

513 upper\_time = datetime.datetime.strptime(upper, '%H:%M')

514 upper\_time += datetime.timedelta(minutes=int(policy))

515 time\_str = datetime.datetime.strptime(time\_str, '%H:%M')

516 *# print(upper\_time, '-', time\_str, '=', end=' ')*

517 *# print(int(upper\_time.strftime('%H%M')) - int(time\_str.strftime('%H%M')))*

518 if (int(upper\_time.strftime('%H%M')) - int(time\_str.strftime('%H%M'))) > 0:

519 return False

520 else:

521 return True

522

523

524 def write\_in\_delivery\_comments(data\_frame\_row, string):

525 if pd.isna(data\_frame\_row['Delivery Comments'].values[0]):

526 data\_frame\_row['Delivery Comments'] = [string]

527 return data\_frame\_row

528 *# 如果已经有了，就不管了*

529 elif string in data\_frame\_row['Delivery Comments'].values[0]:

530 return data\_frame\_row

531 else:

532 *# data\_frame\_row['Delivery Comments'] = [data\_frame\_row['Delivery Comments'].values[0] + '/' + string]*

533 data\_frame\_row['Delivery Comments'] = [string]

534 return data\_frame\_row

535

536

537 def data\_frame\_row\_time\_change(data\_frame\_row):

538 region = data\_frame\_row['Region Code'].values[0]

539

540 try:

541 if pd.isna(region):

542 return data\_frame\_row

543

544 *# 判断 Region Code 属于那个地区*

545 elif region == 'CHI' or region == 'DFW' or region == 'HOU':

546 *# early 时间 latest 时间*

547 early\_time\_str = str(data\_frame\_row['Earliest Dropoff Time'].values[0])

548 new\_time = time\_subtract(early\_time\_str, hours=2, days=0)

549 new\_time\_str = new\_time.strftime('%H:%M')

550 data\_frame\_row['Earliest Dropoff Time'] = [new\_time\_str]

551

552 latest\_time\_str = str(data\_frame\_row['Latest Dropoff Time'].values[0])

553 new\_time = time\_subtract(latest\_time\_str, hours=2, days=0)

554 new\_time\_str = new\_time.strftime('%H:%M')

555 data\_frame\_row['Latest Dropoff Time'] = [new\_time\_str]

556 *# 针对 inbound*

557 *# 如果 时间有空的，跳过*

558 if pd.isna(data\_frame\_row['Inbound Scan Time'].values[0]):

559 new\_time = None

560 else:

561 inbound\_time\_str = str(data\_frame\_row['Inbound Scan Time'].values[0])

562 new\_time = time\_subtract(inbound\_time\_str, hours=2, days=0)

563 new\_time\_str = new\_time.strftime('%H:%M')

564 data\_frame\_row['Inbound Scan Time'] = [new\_time\_str]

565

566 *# 针对 pickup time*

567 if pd.isna(data\_frame\_row['Pickup Time'].values[0]):

568 pass

569 else:

570 pickup\_time\_str = str(data\_frame\_row['Pickup Time'].values[0])

571 new\_pickup\_time = time\_subtract(pickup\_time\_str, hours=2, days=0)

572 new\_pickup\_time\_str = new\_pickup\_time.strftime('%H:%M')

573 data\_frame\_row['Pickup Time'] = [new\_pickup\_time\_str]

574

575 *# 针对 drop off time*

576 if pd.isna(data\_frame\_row['Drop off time'].values[0]):

577 pass

578 else:

579 drop\_time\_str = str(data\_frame\_row['Drop off time'].values[0])

580 new\_drop\_time = time\_subtract(drop\_time\_str, hours=2, days=0)

581 new\_drop\_time\_str = new\_drop\_time.strftime('%H:%M')

582 data\_frame\_row['Drop off time'] = [new\_drop\_time\_str]

583

584 *# 如果前进了一天*

585 if new\_time is None:

586 return data\_frame\_row

587 else:

588 if str(new\_time.date()) == '1899-12-31':

589 date\_str = str(data\_frame\_row['Inbound Scan Date (Linehaul)'].values[0])

590

591 time\_object = datetime.datetime.strptime(date\_str, '%Y-%m-%d')

592 new\_date = time\_object - datetime.timedelta(days=1)

593 new\_date\_str = new\_date.strftime('%Y-%m-%d')

594

595 data\_frame\_row['Inbound Scan Date (Linehaul)'] = [new\_date\_str]

596 return data\_frame\_row

597 else:

598 return data\_frame\_row

599

600 elif region == 'JFK' or region == 'PHL' or region == 'EWR':

601 *# early 时间 latest 时间*

602 early\_time\_str = str(data\_frame\_row['Earliest Dropoff Time'].values[0])

603 new\_time = time\_subtract(early\_time\_str, hours=3, days=0)

604 new\_time\_str = new\_time.strftime('%H:%M')

605 data\_frame\_row['Earliest Dropoff Time'] = [new\_time\_str]

606

607 latest\_time\_str = str(data\_frame\_row['Latest Dropoff Time'].values[0])

608 new\_time = time\_subtract(latest\_time\_str, hours=3, days=0)

609 new\_time\_str = new\_time.strftime('%H:%M')

610 data\_frame\_row['Latest Dropoff Time'] = [new\_time\_str]

611 *# 针对 inbound*

612 *# 如果 时间有空的，跳过*

613 if pd.isna(data\_frame\_row['Inbound Scan Time'].values[0]):

614 new\_time = None

615 else:

616 inbound\_time\_str = str(data\_frame\_row['Inbound Scan Time'].values[0])

617 new\_time = time\_subtract(inbound\_time\_str, hours=3, days=0)

618 new\_time\_str = new\_time.strftime('%H:%M')

619 data\_frame\_row['Inbound Scan Time'] = [new\_time\_str]

620

621 *# 针对 pickup time*

622 if pd.isna(data\_frame\_row['Pickup Time'].values[0]):

623 pass

624 else:

625 pickup\_time\_str = str(data\_frame\_row['Pickup Time'].values[0])

626 new\_pickup\_time = time\_subtract(pickup\_time\_str, hours=3, days=0)

627 new\_pickup\_time\_str = new\_pickup\_time.strftime('%H:%M')

628 data\_frame\_row['Pickup Time'] = [new\_pickup\_time\_str]

629

630 *# 针对 drop off time*

631 if pd.isna(data\_frame\_row['Drop off time'].values[0]):

632 pass

633 else:

634 drop\_time\_str = str(data\_frame\_row['Drop off time'].values[0])

635 new\_drop\_time = time\_subtract(drop\_time\_str, hours=3, days=0)

636 new\_drop\_time\_str = new\_drop\_time.strftime('%H:%M')

637 data\_frame\_row['Drop off time'] = [new\_drop\_time\_str]

638

639 *# 如果前进了一天*

640 if new\_time is None:

641 return data\_frame\_row

642 else:

643 if str(new\_time.date()) == '1899-12-31':

644 date\_str = str(data\_frame\_row['Inbound Scan Date (Linehaul)'].values[0])

645

646 time\_object = datetime.datetime.strptime(date\_str, '%Y-%m-%d')

647 new\_date = time\_object - datetime.timedelta(days=1)

648 new\_date\_str = new\_date.strftime('%Y-%m-%d')

649

650 data\_frame\_row['Inbound Scan Date (Linehaul)'] = [new\_date\_str]

651 return data\_frame\_row

652 else:

653 return data\_frame\_row

654

655 elif region == 'PHX':

656 *# early 时间 latest 时间*

657 early\_time\_str = str(data\_frame\_row['Earliest Dropoff Time'].values[0])

658 new\_time = time\_subtract(early\_time\_str, hours=1, days=0)

659 new\_time\_str = new\_time.strftime('%H:%M')

660 data\_frame\_row['Earliest Dropoff Time'] = [new\_time\_str]

661

662 latest\_time\_str = str(data\_frame\_row['Latest Dropoff Time'].values[0])

663 new\_time = time\_subtract(latest\_time\_str, hours=1, days=0)

664 new\_time\_str = new\_time.strftime('%H:%M')

665 data\_frame\_row['Latest Dropoff Time'] = [new\_time\_str]

666 *# 针对 inbound*

667 *# 如果 时间有空的，跳过*

668 if pd.isna(data\_frame\_row['Inbound Scan Time'].values[0]):

669 new\_time = None

670 else:

671 inbound\_time\_str = str(data\_frame\_row['Inbound Scan Time'].values[0])

672 new\_time = time\_subtract(inbound\_time\_str, hours=1, days=0)

673 new\_time\_str = new\_time.strftime('%H:%M')

674 data\_frame\_row['Inbound Scan Time'] = [new\_time\_str]

675

676 *# 针对 pickup time*

677 if pd.isna(data\_frame\_row['Pickup Time'].values[0]):

678 pass

679 else:

680 pickup\_time\_str = str(data\_frame\_row['Pickup Time'].values[0])

681 new\_pickup\_time = time\_subtract(pickup\_time\_str, hours=1, days=0)

682 new\_pickup\_time\_str = new\_pickup\_time.strftime('%H:%M')

683 data\_frame\_row['Pickup Time'] = [new\_pickup\_time\_str]

684

685 *# 针对 drop off time*

686 if pd.isna(data\_frame\_row['Drop off time'].values[0]):

687 pass

688 else:

689 drop\_time\_str = str(data\_frame\_row['Drop off time'].values[0])

690 new\_drop\_time = time\_subtract(drop\_time\_str, hours=1, days=0)

691 new\_drop\_time\_str = new\_drop\_time.strftime('%H:%M')

692 data\_frame\_row['Drop off time'] = [new\_drop\_time\_str]

693

694 *# 如果前进了一天*

695 if new\_time is None:

696 return data\_frame\_row

697 else:

698 if str(new\_time.date()) == '1899-12-31':

699 date\_str = str(data\_frame\_row['Inbound Scan Date (Linehaul)'].values[0])

700

701 time\_object = datetime.datetime.strptime(date\_str, '%Y-%m-%d')

702 new\_date = time\_object - datetime.timedelta(days=1)

703 new\_date\_str = new\_date.strftime('%Y-%m-%d')

704

705 data\_frame\_row['Inbound Scan Date (Linehaul)'] = [new\_date\_str]

706 return data\_frame\_row

707 else:

708 return data\_frame\_row

709

710 else:

711 return data\_frame\_row

712 except ValueError:

713 return data\_frame\_row

714

715

716 def time\_subtract(time\_str, hours, days):

717 time\_object = datetime.datetime.strptime(time\_str, '%H:%M')

718 new\_time = time\_object - datetime.timedelta(hours=hours, days=days)

719 return new\_time

720

721

722 def change\_Scheduled\_Delivery\_Date(data\_frame\_row):

723 *# if pd.isna(data\_frame\_row['Scheduled Delivery Date'][0]):*

724 *# return data\_frame\_row*

725 *# else:*

726 s\_date\_str = data\_frame\_row['Scheduled Delivery Date'].values[0]

727 if format\_1(s\_date\_str):

728 s\_str = datetime.datetime.strptime(s\_date\_str, '%Y/%m/%d')

729 s\_str = s\_str.strftime('%Y-%m-%d')

730 data\_frame\_row['Scheduled Delivery Date'] = [s\_str]

731 return data\_frame\_row

732 elif format\_2(s\_date\_str):

733 s\_str = datetime.datetime.strptime(s\_date\_str, '%m/%d/%Y')

734 s\_str = s\_str.strftime('%Y-%m-%d')

735 data\_frame\_row['Scheduled Delivery Date'] = [s\_str]

736 return data\_frame\_row

737 elif format\_3(s\_date\_str):

738 return data\_frame\_row

739 else:

740 return data\_frame\_row

741

742

743 def format\_1(date):

744 try:

745 datetime.datetime.strptime(date, "%Y/%m/%d")

746 return True

747 except:

748 return False

749

750

751 def format\_2(date):

752 try:

753 datetime.datetime.strptime(date, "%m/%d/%Y")

754 return True

755 except:

756 return False

757

758

759 def format\_3(date):

760 try:

761 datetime.datetime.strptime(date, "%Y-%m-%d")

762 return True

763 except:

764 return False

765

766

767 if \_\_name\_\_ == '\_\_main\_\_':

768 d = '2021/2/2'

769 g = datetime.datetime.strptime(d, '%Y/%m/%d')

770 print(g)

|  |
| --- |
| concat\_csv.py |

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

|  |
| --- |
| downloader.py |

1 import os

2 import datetime

3 import time

4 import requests

5 import json

6 import pandas as pd

7 from requests\_ntlm import HttpNtlmAuth

8 from tkinter import Toplevel, Label, Entry, Button, StringVar, messagebox

9

10

11 class DownLoader(object):

12 def \_\_init\_\_(self, root=None):

13 self.root = root

14 self.window = Toplevel(master=self.root)

15 self.url = 'https://dataorch.beta.axlehire.com/reports/all/request'

16 self.header = {

17 'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) '

18 'Chrome/95.0.4638.69 Safari/537.36',

19 'content-type': 'application/json',

20 'cookie': r'fp=1a39e1225ea764ca9f2abf599fafba34; xtoken="dE9DbW1wYkZDI/B28g5MkirtzwljFDty7THWI75r/mVq4do8Y'

21 r'KOJBeUtONSQ1d3L1Yb5JCAEZPTk\012FFj7LXpbKjSaV71j1S6I9zjtTLurIi1ddgqe+xsIRU84cjg0Sktu\012"'

22 }

23 self.user\_name = 'yanxia.ji'

24 self.password = 'Axl12345'

25

26 def download\_from\_url(self, url, file\_name):

27 print(url)

28 *# url = 'https://dataorch.beta.axlehire.com/reports/uploaded/8d38816a-98f4-4461-9d4b-1f79cd360e34/download'*

29 self.make\_dir('all\_report\_history')

30 session = requests.Session()

31 time.sleep(5)

32 response = session.get(url=url, headers=self.header)

33 *# response = session.get(url=url, headers=self.header)*

34 if response.status\_code == 200:

35 json\_data = json.loads(response.text)

36 print('json\_data', json\_data)

37 response = session.get(url=url+'/download', headers=self.header)

38 if 'url' in json\_data.keys():

39 with open(file\_name, 'wb') as fp:

40 fp.write(response.content)

41 else:

42 self.download\_from\_url(url, file\_name)

43

44 else:

45 self.download\_from\_url(url, file\_name)

46

47 def get\_csv\_from\_date(self, client\_id, date, file\_name):

48 session = requests.Session()

49 if client\_id.find('，') == -1:

50 client\_id\_string = [str(client\_id)]

51 else:

52 client\_id\_string = client\_id.split('，')

53

54 *# date*

55 date = datetime.datetime.strptime(date, '%Y/%m/%d').strftime('%Y-%m-%d')

56

57 json\_data = {

58 'clients': client\_id\_string,

59 'date': date,

60 }

61

62 response = session.post(url=self.url, headers=self.header, json=json\_data,

63 auth=HttpNtlmAuth(self.user\_name, self.password))

64 json\_response = json.loads(response.content)

65 url = 'https://dataorch.beta.axlehire.com/reports/uploaded/'

66 url += json\_response['id']

67 self.download\_from\_url(url, file\_name)

68

69 def get\_date\_list(self, from\_date, to\_date):

70 from\_date = datetime.datetime.strptime(from\_date, '%Y/%m/%d')

71 to\_date = datetime.datetime.strptime(to\_date, '%Y/%m/%d')

72 diff = (to\_date - from\_date).days

73 if diff <= 0:

74 messagebox.showwarning(title='警告', message='日期范围有误')

75 else:

76 date\_list = [from\_date.strftime('%Y/%m/%d')]

77 for i in range(1, diff+1):

78 date = from\_date + datetime.timedelta(days=i)

79 date = date.strftime('%Y/%m/%d')

80 date\_list.append(date)

81 return date\_list

82

83 @staticmethod

84 def make\_dir(path):

85 if not os.path.exists(path):

86 os.mkdir(path)

87

88 def run(self):

89 *# 初始化界面*

90 self.window.geometry('700x240')

91 self.client = StringVar()

92 self.date = StringVar()

93 Label(self.window, text='输入 client 号（如有多个，请用，（中文逗号）分隔）:').place(x=50, y=20)

94 Entry(self.window, textvariable=self.client).place(x=50, y=60)

95 Label(self.window, text='输入日期（形如 2021/11/07 如有多个日期请用,分隔,如果为时间段,请输入形如 2021/11/07-2021/11/09）:')\

96 .place(x=50, y=100)

97 Entry(self.window, textvariable=self.date).place(x=50, y=140)

98 Button(self.window, text='生成 all\_report csv', command=self.confirm).place(x=50, y=190)

99 self.window.mainloop()

100

101 @staticmethod

102 def is\_date(date):

103 try:

104 datetime.datetime.strptime(date, "%Y/%m/%d")

105 return True

106 except:

107 return False

108

109 def confirm(self):

110 if self.check\_client():

111 *# date 为 range*

112 now = datetime.datetime.now().strftime('%m月%d日-%H点%M分%S秒')

113 if self.date.get().find('-') != -1:

114 date\_from\_to\_list = self.date.get().split('-')

115 date\_list = self.get\_date\_list(from\_date=date\_from\_to\_list[0], to\_date=date\_from\_to\_list[1])

116 *# 创建文件夹*

117 folder\_name = self.date.get().replace('-', 'to')

118 folder\_name = folder\_name.replace('/', '-')

119 folder\_name += '&client=' + self.client.get() + '\_' + now

120 self.make\_dir(f'all\_report\_history/{folder\_name}')

121 for date in date\_list:

122 date\_name = date.replace('/', '-')

123 self.get\_csv\_from\_date(

124 client\_id=self.client.get(),

125 date=date,

126 file\_name=f'all\_report\_history/{folder\_name}/client={self.client.get()}&date='f'{date\_name}&{now}.csv'

127 )

128 self.concat\_from\_folder(f'all\_report\_history/{folder\_name}')

129 *# 单个date*

130 elif self.is\_date(self.date.get()):

131 date = datetime.datetime.strptime(self.date.get(), '%Y/%m/%d').strftime('%Y-%m-%d')

132 self.get\_csv\_from\_date(

133 client\_id=self.client.get(),

134 date=self.date.get(),

135 file\_name=f'all\_report\_history/client={self.client.get()}&date={date}&{now}.csv'

136 )

137 else:

138 messagebox.showwarning(title='警告', message='日期格式有误')

139 else:

140 messagebox.showwarning(title='警告', message='client格式有误')

141

142 def check\_client(self):

143 *# client 是单个*

144 if self.client.get().find('，') == -1:

145 if not self.client.get().isnumeric():

146 return False

147 *# 是数字*

148 else:

149 if int(self.client.get()) <= 11 or int(self.client.get()) == 471 or int(self.client.get()) == 621 \

150 or (int(self.client.get()) >= 15 and int(self.client.get()) <= 214):

151 return True

152 return False

153 *# client 是多个*

154 else:

155 client\_list = self.client.get().split('，')

156 for client in client\_list:

157 if client.find('，') == -1:

158 if not client.isnumeric():

159 return False

160 *# 是数字*

161 else:

162 if int(client) <= 214 or int(client) == 471 or int(client) == 621:

163 return True

164 return False

165

166 @staticmethod

167 def get\_dict\_from\_tracking\_code(tracking\_code):

168 url = 'https://dataorch.axlehire.com/shipments/search'

169 header = {

170 'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36',

171 'content-type': 'application/json',

172 'cookie': r'fp=1a39e1225ea764ca9f2abf599fafba34; xtoken="dE9DbW1wYkZDI/B28g5MkirtzwljFDty7THWI75r/mVq4do8YKOJBeUtONSQ1d3L1Yb5JCAEZPTk\012FFj7LXpbKjSaV71j1S6I9zjtTLurIi1ddgqe+xsIRU84cjg0Sktu\012"'}

173 *# 生成 post 的 json\_data*

174 data\_dict = {'size': 15, 'q': tracking\_code,

175 'filters': {}, 'sorts': ['-dropoff\_earliest\_ts']}

176 json\_data = json.dumps(data\_dict)

177

178 session = requests.Session()

179 user = 'yanxia.ji'

180 password = 'Axl12345'

181 response = session.post(url=url, headers=header, data=json\_data, auth=HttpNtlmAuth(user, password))

182

183 result\_dict = json.loads(response.text)

184 return result\_dict

185

186 @staticmethod

187 def concat\_from\_folder(folder\_path):

188 dir\_list = os.listdir(folder\_path)

189 res\_df = pd.DataFrame()

190 for file in dir\_list:

191 file\_path = folder\_path + '/' + file

192 temp\_df = pd.read\_csv(file\_path)

193 res\_df = pd.concat([res\_df, temp\_df])

194 res\_df.to\_csv(folder\_path + '/all.csv', index=False)

195 path = folder\_path + '/all.csv'

196 messagebox.showinfo(title='成功', message=f'输出路径为: {path}')

197 return res\_df

198

199

200 if \_\_name\_\_ == '\_\_main\_\_':

201 downloader = DownLoader()

202 *# downloader.get\_csv\_from\_date(client\_id='159', date='2021-10-19')2021-10-19*

203

204 *# downloader.get\_csv\_from\_date(*

205 *# client\_id=['49'],*

206 *# date='2021-11-17',*

207 *# file\_name='ff.csv'*

208 *# )*

209 *# downloader.download\_from\_url(url='https://dataorch.beta.axlehire.com/reports/uploaded/f705aa7f-6b32-4868-9892-5fb9ab6e138d', file\_name='ff')*

210 *# st = '2021/11/17'*

211 *# s = str(int(datetime.datetime.strptime(st, '%Y/%m/%d').timestamp()))*

212 *# print(s)*

213 downloader.run()

214

|  |
| --- |
| generator.py |

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 from tkinter import Toplevel

6

7 import pandas as pd

8

9

10 class Generator(object):

11 def \_\_init\_\_(self, root=None):

12 self.root = root

13 self.window = Toplevel(master=self.root)

14 self.reason\_code = pd.read\_csv('utils/files/JJ - Reason code.csv')

15

16 def options(self):

17 pass

18

19 def get\_final(self, csv\_file):

20 *# 1. 读入传入的 csv*

21 res\_df = pd.read\_csv(csv\_file)

22 res\_df.rename(columns={'HF Reason Code': 'AH Assessment'}, inplace=True)

23 *# 2. 遍历 res\_df*

24 for idx, row in res\_df.iterrows():

25 res\_df.iloc[idx: idx + 1, :] = self.parse\_rows(res\_df.iloc[idx: idx + 1, :])

26 *# print(self.parse\_rows(res\_df.iloc[idx: idx + 1, :]))*

27 print(f'\rwrite {idx} rows', end='')

28 return res\_df

29

30 def parse\_rows(self, data\_frame\_rows):

31 *# 解析每一行*

32 if pd.isna(data\_frame\_rows['Answer Number'].values[0]):

33 return data\_frame\_rows

34

35 number = int(data\_frame\_rows['Answer Number'].values[0]) \

36 if type(data\_frame\_rows['Answer Number'].values[0]) == 'float' else data\_frame\_rows['Answer Number'].values[0]

37

38 *# 不是数字，一定是 14 15 apt 这种形式*

39 if not str(number).isnumeric():

40 answer\_list = str(number).split(' ')

41 *# 三种形式 pic shows building #,the correct is #*

42 if 'apt' in answer\_list:

43 data\_frame\_rows = self.copy\_rows(data\_frame\_rows, int(109))

44 apt\_answer = str(data\_frame\_rows['POD Quality'].values[0]). \

45 replace('shows apt #', f'shows apt #{answer\_list[0]}'). \

46 replace('the correct is apt#', f'the correct is apt#{answer\_list[1]}')

47 data\_frame\_rows['POD Quality'] = [apt\_answer]

48 return data\_frame\_rows

49 elif 'st' in answer\_list:

50 data\_frame\_rows = self.copy\_rows(data\_frame\_rows, int(108))

51 s\_answer = str(data\_frame\_rows['POD Quality'].values[0]). \

52 replace('shows street #', f'shows street #{answer\_list[0]}'). \

53 replace('the correct is #', f'the correct is #{answer\_list[1]}')

54 data\_frame\_rows['POD Quality'] = [s\_answer]

55 return data\_frame\_rows

56 elif 'b' in answer\_list:

57 data\_frame\_rows = self.copy\_rows(data\_frame\_rows, int(107))

58 b\_answer = str(data\_frame\_rows['POD Quality'].values[0]). \

59 replace('shows building #', f'shows building #{answer\_list[0]}'). \

60 replace('the correct is #', f'the correct is #{answer\_list[1]}')

61 data\_frame\_rows['POD Quality'] = [b\_answer]

62 return data\_frame\_rows

63 else:

64 return data\_frame\_rows

65 else:

66 data\_frame\_rows = self.copy\_rows(data\_frame\_rows, int(number))

67 return data\_frame\_rows

68

69 def copy\_rows(self, data\_frame\_row, index):

70 pd.set\_option("display.max\_columns", 50)

71

72 *# print(data\_frame\_row, 'index', index)*

73 def nan\_to\_none(x):

74 if str(x) == 'nan' or pd.isna(x):

75 return ''

76 return x

77

78 if pd.isna(data\_frame\_row['POD Valid?'].values[0]) and pd.isna(data\_frame\_row['POD Quality'].values[0]) and \

79 pd.isna(data\_frame\_row['Issue Category'].values[0]) and pd.isna(

80 data\_frame\_row['Delivery Comments'].values[0]) and \

81 pd.isna(data\_frame\_row['AH Assessment'].values[0]):

82 data\_frame\_row['POD Valid?'] = [nan\_to\_none(self.reason\_code.loc[index, 'POD'])]

83 data\_frame\_row['POD Quality'] = [nan\_to\_none(self.reason\_code.loc[index, 'POD Qaulity'])]

84 data\_frame\_row['Issue Category'] = [self.reason\_code.loc[index, 'Issue Category']]

85 data\_frame\_row['Delivery Comments'] = [self.reason\_code.loc[index, 'Delivery Comments']]

86 data\_frame\_row['AH Assessment'] = [self.reason\_code.loc[index, 'AH Assignment']]

87

88 return data\_frame\_row

89

90 *# 如果不是空的，加一个 / 再将内容附着上*

91 else:

92 *# print('不是空的')*

93 data\_frame\_row['POD Valid?'] = [nan\_to\_none(self.reason\_code.loc[index, 'POD'])]

94 data\_frame\_row['POD Quality'] = [nan\_to\_none(self.reason\_code.loc[index, 'POD Qaulity'])]

95 if index == 122 or index == 123:

96 return data\_frame\_row

97 *# 如果本来就有，比如已经是 Delivery 了，你再加个 Delivery 就不对了*

98 if self.reason\_code.loc[index, 'Issue Category'] in str(data\_frame\_row['Issue Category'].values[0]):

99 pass

100 else:

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 *# print('附着进去了')*

110 return data\_frame\_row

111

112

113 def run(files, dis\_files):

114 generator = Generator()

115 pd.set\_option('display.max\_columns', 50)

116 *# print(generator.reason\_code.iloc[122:123, :])*

117 *# print(generator.reason\_code.iloc[123:124, :])*

118 df = generator.get\_final(

119 csv\_file=files

120 )

121 *# print(df)*

122 df.to\_csv(dis\_files, index=False)

123

124

|  |
| --- |
| 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 *# print('\nres', res\_data.columns)*

50 for title in same.columns:

51 if title in res\_data.columns:

52 res\_data[title] = same[title]

53 *# print(title + ' success to write in')*

54

55 *# 6. 将 res\_data 的标题重置为 ending 的标题*

56 res\_data.columns = big\_sheet.columns

57

58 return res\_data

59

60 elif day == '3':

61 print(ending\_df.columns)

62 print(all\_report\_df.columns)

63 *# 0. 获取文件*

64 big\_sheet = ending\_df

65 boss2me = boss2me\_df

66 report = all\_report\_df

67

68 *# 3. initialize res\_data*

69 columns\_list = list(big\_sheet.columns)

70 columns\_list.append('delivery\_date')

71 columns\_list.append('Earliest Dropoff Time')

72 columns\_list.append('Latest Dropoff Time')

73 columns\_list.append('Earliest Dropoff Date')

74 columns\_list.append('Latest Dropoff Date')

75 res\_data = pd.DataFrame(columns=columns\_list, dtype='object')

76

77 *# 1. 将boss "tracking code" 改为和 report "Tracking Code" 一致*

78 *# boss 的 tracking code 有多种可能 "tracking #" or "Tracking Number"*

79 boss2me = change\_title\_name(boss2me, re.search(r'\'[Tt]racking(#| Number| code| Code|\_code|\_Code|)\'',

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

81

82 *# 2. 合并 boss 和 report 合并为 same*

83 same = pd.merge(boss2me, report, how='left', on='Tracking Code')

84

85 *# 4. 将 res\_data 的一些标题改为 same 的*

86 *# ending 与 same 的不同除 Region Code --> Region, REgion Code --> Region*

87 *# ending\_wednesday 的 Drop off Time ending\_thursday是 Drop off time --> Dropoff Time*

88 res\_data = change\_title\_name(res\_data, re.search(r'\'[Tt]racking(#| Number| code| Code)\'',

89 str(res\_data.columns)).group(0)[1:-1], "Tracking Code")

90 res\_data = change\_title\_name(res\_data, re.search(

91 r'\'((Region Code)|(REgion Code)|(region Code)|(rEgion Code)|(Region code)|(REgion code))\'',

92 str(res\_data.columns)).group(0)[1:-1], "Region")

93 res\_data = change\_title\_name(res\_data, 'Assignment ID', 'Assignment Id')

94 res\_data = change\_title\_name(res\_data,

95 re.search(r'\'(Issue)|(Reason for Complaint)\'', str(res\_data.columns)).group(0)[

96 :-1], 'Reason for Complaint')

97 res\_data = change\_title\_name(res\_data, 'Inbound Scan Date (Linehaul)', 'Inbound Scan Date')

98 res\_data = change\_title\_name(res\_data, 'Pickup remark', 'Pickup Remark')

99 res\_data = change\_title\_name(res\_data, 'Drop off date', 'Dropoff Date')

100 res\_data = change\_title\_name(res\_data, re.search(r'\'Drop off [Tt]ime\'',

101 str(res\_data.columns)).group(0)[1:-1], "Dropoff Time")

102 res\_data = change\_title\_name(res\_data, 'Drop off status', 'Dropoff Status')

103 res\_data = change\_title\_name(res\_data, 'Drop off remark', 'Dropoff Remark')

104 res\_data = change\_title\_name(res\_data, 'Requested Amount', 'Requested Credit Amount')

105

106 *# 解决 Reason for complaint 问题*

107 if 'Issue' in same.columns:

108 res\_data['Reason for Complaint'] = same['Issue']

109

110 *# 5. 遍历 same 的标题，将 same 的数据写入 same 和 res\_data 共有的标题下*

111 *# print('\nres', res\_data.columns)*

112 for title in same.columns:

113 if title in res\_data.columns:

114 res\_data[title] = same[title]

115 *# print(title + ' success to write in')*

116

117 *# 6. 将 res\_data 的标题重置为 ending 的标题*

118 columns\_list = list(big\_sheet.columns)

119 columns\_list.append('delivery\_date')

120 columns\_list.append('Earliest Dropoff Time')

121 columns\_list.append('Latest Dropoff Time')

122 columns\_list.append('Earliest Dropoff Date')

123 columns\_list.append('Latest Dropoff Date')

124 res\_data.columns = columns\_list

125

126 return res\_data

127

128

129 def change\_title\_name(pd, pd\_title, pd\_title\_change):

130 *# print(pd\_title, 'change to', pd\_title\_change)*

131 df = pd.rename(columns={pd\_title: pd\_title\_change})

132 return df

133