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

1 import json

2 import os

3 import shutil

4 import uuid

5

6 import pandas as pd

7 import utils.analyser\_utils as analyser\_utils

8 import requests

9 import cv2.cv2 as cv2

10 import datetime

11

12 from const import USER\_NAME, USER\_PSW

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

14 from requests\_ntlm import HttpNtlmAuth

15

16

17 class Analyser(object):

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

19 self.root = root

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

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

22 self.window = None

23 self.temp\_window = None

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

25 self.header = {

26 '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',

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

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

29 self.index = 0

30 self.save\_folder\_path = save\_path

31 self.day = day

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

33

34 def initialize\_param\_dict(self):

35 *# 普通分析*

36 if self.day == '4':

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

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

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

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

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

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

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

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

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

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

47 return param\_dict

48 *# 智能分析*

49 elif self.day == '3':

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

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

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

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

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

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

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

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

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

59 return param\_dict

60

61 @staticmethod

62 def initialize\_df(first\_df):

63 *"""*

64  *取一切需要打标签的内容*

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

66  *Issue Category 为空的情况 （没填上）*

67  *"""*

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

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

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

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

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

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

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

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

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

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

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

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

80 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\012FFj7LXpbKjSaV71j1S6I9zjtTLurIi1ddgqe+xsIRU84cjg0Sktu\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

707 img = cv2.imread(out\_path)

708 min\_side = 768

709 size = img.shape

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

711 *# 长边缩放为 min\_side*

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

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

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

715 *# 填充至 min\_side \* min\_side*

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

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

718 min\_side - new\_w) / 2

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

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

721 min\_side - new\_w) / 2

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

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

724 min\_side - new\_w) / 2

725 else:

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

727 min\_side - new\_w) / 2

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

729 value=[0, 0, 0])

730 return upscale\_img

731

732

733 def run\_cmd(cmd):

734 *"""*

735  *运行命令并返回返回值*

736  *:param cmd: 命令*

737  *:return: 命令输出*

738  *"""*

739 var = os.popen(cmd)

740 result = var.read()

741 var.close()

742 return result

743

744

745 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 |

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

43 *# 如果全是空的，直接复制*

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

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

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

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

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

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

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

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

52

53 return data\_frame\_row

54 *# 如果不是空的，加一个 / 再将内容附着上*

55 else:

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

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

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

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

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

61 return data\_frame\_row

62

63

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

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

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

67

68 *# 如果 pick 当天送达*

69 if pickup\_diff == 0:

70 *# 如果 pick 当天晚于 12*

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

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

73

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

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

76 *# 判断 delivery 当天*

77 if delivery\_diff == 0:

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

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

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

81 return data\_frame\_row

82 *# 配送晚于 2 天*

83 else:

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

85

86 if delivery\_diff == 1:

87 return data\_frame\_row

88 else:

89 if delivery\_diff < 2:

90 return data\_frame\_row

91 else:

92

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

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

95 return data\_frame\_row

96 *# 如果 pick 早于 12 点*

97 else:

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

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

100 *# 判断 delivery 当天*

101 if delivery\_diff == 0:

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

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

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

105 return data\_frame\_row

106 *# 配送晚于 2 天*

107 else:

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

109

110 if delivery\_diff == 1:

111 return data\_frame\_row

112 else:

113 if delivery\_diff < 2:

114 return data\_frame\_row

115 else:

116

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

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

119 return data\_frame\_row

120 return data\_frame\_row

121

122 *# 如果 pick 晚了 n 天*

123 if pickup\_diff > 1:

124 *# 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,

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

204 return data\_frame\_row

205 return data\_frame\_row

206

207 *# pick up 晚于 1 天，看 delivery*

208 elif pickup\_diff > 1:

209

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

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

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

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

214 return data\_frame\_row

215 return data\_frame\_row

216

217 *# pickup 没晚*

218 else:

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

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

221 if delivery\_diff == 0:

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

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

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

225 return data\_frame\_row

226 else:

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

228 if delivery\_diff == 1:

229

230 return data\_frame\_row

231 else:

232 if delivery\_diff < 2:

233 return data\_frame\_row

234

235 else:

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

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

238 return data\_frame\_row

239 return data\_frame\_row

240

241

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

243 *# 判断 shipment status 完了*

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

245 *# 这里还要继续分析，但是比较复杂，不过多写了*

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

247 return data\_frame\_row

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

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

250 return data\_frame\_row

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

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

253 return data\_frame\_row

254

255 *# 判断 Inbound status Missing*

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

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

258 return data\_frame\_row

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

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

261 return data\_frame\_row

262

263 *# 开始逐一检查 Drop off status*

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

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

266 if day == '4':

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

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

269 return data\_frame\_row

270 if day == '3':

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

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

273 return data\_frame\_row

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

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

276 return data\_frame\_row

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

278 if day == '3':

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

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

281 return data\_frame\_row

282 if day == '4':

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

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

285 return data\_frame\_row

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

287 if day == '3':

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

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

290 return data\_frame\_row

291 if day == '4':

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

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

294 return data\_frame\_row

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

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

297 return data\_frame\_row

298

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

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

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

302 return data\_frame\_row

303 if 'missing by outbound' 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 return data\_frame\_row

307 if day == '4':

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

309 return data\_frame\_row

310

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

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

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] == 'PENDING':

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

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

319 return data\_frame\_row

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

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

322 if day == '4':

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

324 return data\_frame\_row

325 elif day == '3':

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

327 return data\_frame\_row

328

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

330 *# ok*

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

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

333 return data\_frame\_row

334 *# 如果 remark 不是空*

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

336 *# ok*

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

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

339 return data\_frame\_row

340 *# ok*

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

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

343 return data\_frame\_row

344 *# ok*

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

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

347 return data\_frame\_row

348 *# ok*

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

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

351 *# return data\_frame\_row*

352 *# ok*

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

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

355 return data\_frame\_row

356 *# ok*

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

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

359 return data\_frame\_row

360 *# ok*

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

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

363 return data\_frame\_row

364 *# ok*

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

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

367 return data\_frame\_row

368 *# ok*

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

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

371 return data\_frame\_row

372 *# ok*

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

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

379 return data\_frame\_row

380 *# ok*

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

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

383 return data\_frame\_row

384 *# ok*

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

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

387 return data\_frame\_row

388 *# ok*

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

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

391

392 *# 如果是 SUCCEEDED 状态*

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

394 *# 先查看一些明显的问题*

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

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

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

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

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

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

401

402 *# 再看时间差，附着到之前的结果*

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

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

405 *# 如果 inbound\_diff 等于 0*

406 if inbound\_diff == 0:

407 *# 如果 inbound 当天晚于 12 点*

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

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

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

411 return data\_frame\_row

412 *# 如果 inbound 当天早于 12 点*

413 else:

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

415

416 *# 如果 inbound\_diff 大于一天*

417 elif inbound\_diff > 0:

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

419 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]:

464 return data\_frame\_row

465 else:

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

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

468 return data\_frame\_row

469

470

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

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

473

474 try:

475 if pd.isna(region):

476 return data\_frame\_row

477

478 *# 判断 Region Code 属于那个地区*

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

480 *# early 时间 latest 时间*

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

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

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

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

485

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

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

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

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

490 *# 针对 inbound*

491 *# 如果 时间有空的，跳过*

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

493 new\_time = None

494 else:

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

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

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

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

499

500 *# 针对 pickup time*

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

502 pass

503 else:

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

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

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

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

508

509 *# 针对 drop off time*

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

511 pass

512 else:

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

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

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

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

517

518 *# 如果前进了一天*

519 if new\_time is None:

520 return data\_frame\_row

521 else:

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

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

524

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

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

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

528

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

530 return data\_frame\_row

531 else:

532 return data\_frame\_row

533

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

535 *# early 时间 latest 时间*

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

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

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

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

540

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

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

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

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

545 *# 针对 inbound*

546 *# 如果 时间有空的，跳过*

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

548 new\_time = None

549 else:

550 inbound\_time\_str = str(data\_frame\_row['Inbound Scan Time'].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

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

640 return data\_frame\_row

641 else:

642 return data\_frame\_row

643

644 else:

645 return data\_frame\_row

646 except ValueError:

647 return data\_frame\_row

648

649

650 def time\_subtract(time\_str, hours, days):

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

652 new\_time = time\_object - datetime.timedelta(hours=hours, days=days)

653 return new\_time

654

655

656 def change\_Scheduled\_Delivery\_Date(data\_frame\_row):

657 s\_date\_str = data\_frame\_row['Scheduled Delivery Date'].values[0]

658 if format\_1(s\_date\_str):

659 s\_str = datetime.datetime.strptime(s\_date\_str, '%Y/%m/%d')

660 s\_str = s\_str.strftime('%Y-%m-%d')

661 data\_frame\_row['Scheduled Delivery Date'] = [s\_str]

662 return data\_frame\_row

663 elif format\_2(s\_date\_str):

664 s\_str = datetime.datetime.strptime(s\_date\_str, '%m/%d/%Y')

665 s\_str = s\_str.strftime('%Y-%m-%d')

666 data\_frame\_row['Scheduled Delivery Date'] = [s\_str]

667 return data\_frame\_row

668 elif format\_3(s\_date\_str):

669 return data\_frame\_row

670 else:

671 return data\_frame\_row

672

673

674 def format\_1(date):

675 try:

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

677 return True

678 except:

679 return False

680

681

682 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*

26 self.conv1 = norm(nn.Conv2d(num\_feat, num\_feat \* 2, 4, 2, 1, bias=False))

27 self.conv2 = norm(nn.Conv2d(num\_feat \* 2, num\_feat \* 4, 4, 2, 1, bias=False))

28 self.conv3 = norm(nn.Conv2d(num\_feat \* 4, num\_feat \* 8, 4, 2, 1, bias=False))

29 *# upsample*

30 self.conv4 = norm(nn.Conv2d(num\_feat \* 8, num\_feat \* 4, 3, 1, 1, bias=False))

31 self.conv5 = norm(nn.Conv2d(num\_feat \* 4, num\_feat \* 2, 3, 1, 1, bias=False))

32 self.conv6 = norm(nn.Conv2d(num\_feat \* 2, num\_feat, 3, 1, 1, bias=False))

33 *# extra convolutions*

34 self.conv7 = norm(nn.Conv2d(num\_feat, num\_feat, 3, 1, 1, bias=False))

35 self.conv8 = norm(nn.Conv2d(num\_feat, num\_feat, 3, 1, 1, bias=False))

36 self.conv9 = nn.Conv2d(num\_feat, 1, 3, 1, 1)

37

38 def forward(self, x):

39 *# downsample*

40 x0 = F.leaky\_relu(self.conv0(x), negative\_slope=0.2, inplace=True)

41 x1 = F.leaky\_relu(self.conv1(x0), negative\_slope=0.2, inplace=True)

42 x2 = F.leaky\_relu(self.conv2(x1), negative\_slope=0.2, inplace=True)

43 x3 = F.leaky\_relu(self.conv3(x2), negative\_slope=0.2, inplace=True)

44

45 *# upsample*

46 x3 = F.interpolate(x3, scale\_factor=2, mode='bilinear', align\_corners=False)

47 x4 = F.leaky\_relu(self.conv4(x3), negative\_slope=0.2, inplace=True)

48

49 if self.skip\_connection:

50 x4 = x4 + x2

51 x4 = F.interpolate(x4, scale\_factor=2, mode='bilinear', align\_corners=False)

52 x5 = F.leaky\_relu(self.conv5(x4), negative\_slope=0.2, inplace=True)

53

54 if self.skip\_connection:

55 x5 = x5 + x1

56 x5 = F.interpolate(x5, scale\_factor=2, mode='bilinear', align\_corners=False)

57 x6 = F.leaky\_relu(self.conv6(x5), negative\_slope=0.2, inplace=True)

58

59 if self.skip\_connection:

60 x6 = x6 + x0

61

62 *# extra convolutions*

63 out = F.leaky\_relu(self.conv7(x6), negative\_slope=0.2, inplace=True)

64 out = F.leaky\_relu(self.conv8(out), negative\_slope=0.2, inplace=True)

65 out = self.conv9(out)

66

67 return out

68

|  |
| --- |
| downloader.py |

1 import os

2 import datetime

3 import time

4 import requests

5 import json

6 import pandas as pd

7

8 from const import USER\_PSW, USER\_NAME

9 from requests\_ntlm import HttpNtlmAuth

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

11

12

13 class DownLoader(object):

14 def \_\_init\_\_(self, root=None):

15 self.root = root

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

17 self.url = 'https://dataorch.beta.axlehire.com/reports/all/request'

18 self.header = {

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

20 'Chrome/95.0.4638.69 Safari/537.36',

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

22 'cookie': r'fp=1a39e1225ea764ca9f2abf599fafba34; xtoken="dE9DbW1wYkZDI/B28g5MkirtzwljFDty7THWI75r/mVq4do8Y'

23 r'KOJBeUtONSQ1d3L1Yb5JCAEZPTk\012FFj7LXpbKjSaV71j1S6I9zjtTLurIi1ddgqe+xsIRU84cjg0Sktu\012"'

24 }

25 self.user\_name = USER\_NAME

26 self.password = USER\_PSW

27

28 def download\_from\_url(self, url, file\_name):

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 if response.status\_code == 200:

34 json\_data = json.loads(response.text)

35 response = session.get(url=url+'/download', headers=self.header)

36 if 'url' in json\_data.keys():

37 with open(file\_name, 'wb') as fp:

38 fp.write(response.content)

39 else:

40 self.download\_from\_url(url, file\_name)

41

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\012FFj7LXpbKjSaV71j1S6I9zjtTLurIi1ddgqe+xsIRU84cjg0Sktu\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

13 class Generator(object):

14 def \_\_init\_\_(self, root=None):

15 self.root = root

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

17 self.reason\_code = pd.read\_csv('utils/files/JJ - Reason code.csv')

18

19 def options(self):

20 pass

21

22 def get\_final(self, csv\_file):

23 *# 1. 读入传入的 csv*

24 res\_df = pd.read\_csv(csv\_file)

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

26 *# 2. 遍历 res\_df*

27 for idx, row in res\_df.iterrows():

28 res\_df.iloc[idx: idx + 1, :] = self.parse\_rows(res\_df.iloc[idx: idx + 1, :])

29 print(f'\rwrite {idx} rows', end='')

30 return res\_df

31

32 def parse\_rows(self, data\_frame\_rows):

33 *# 解析每一行*

34 if pd.isna(data\_frame\_rows['Answer Number'].values[0]):

35 return data\_frame\_rows

36

37 number = int(data\_frame\_rows['Answer Number'].values[0]) \

38 if type(data\_frame\_rows['Answer Number'].values[0]) == 'float' else data\_frame\_rows['Answer Number'].values[0]

39

40 *# 不是数字，一定是 14 15 apt 这种形式*

41 if not str(number).isnumeric():

42 answer\_list = str(number).split(' ')

43 *# 三种形式 pic shows building #,the correct is #*

44 if 'apt' in answer\_list:

45 data\_frame\_rows = self.copy\_rows(data\_frame\_rows, int(109))

46 apt\_answer = str(data\_frame\_rows['POD Quality'].values[0]). \

47 replace('shows apt #', f'shows apt #{answer\_list[0]}'). \

48 replace('the correct is apt#', f'the correct is apt#{answer\_list[1]}')

49 data\_frame\_rows['POD Quality'] = [apt\_answer]

50 return data\_frame\_rows

51 elif 'st' in answer\_list:

52 data\_frame\_rows = self.copy\_rows(data\_frame\_rows, int(108))

53 s\_answer = str(data\_frame\_rows['POD Quality'].values[0]). \

54 replace('shows street #', f'shows street #{answer\_list[0]}'). \

55 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)

229 finally:

230 os.system(os.getcwd() + '/main.py')

231 if 'Already up to date' in execute\_str:

232 messagebox.showinfo(title='无可用更新', message='无可用更新')

233 return False

234 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')

235 else:

236 return False

237

238 @staticmethod

239 def show\_update():

240

241 message = '版本 V1.0\n更新内容:\n'

242 message += '''- 新增若干功能

243 1. 新增清除缓存时，显示实际清除缓存的内存

244 2. 新增 download 按钮，可以随时下载 all\_report '''

245 messagebox.showinfo(

246 title='更新内容',

247 message=message

248 )

249

250 def run(self):

251 self.window.title(f'AxleHireTools : version: {self.version}')

252 self.window.geometry('850x450')

253

254 *# label ending*

255 self.select\_box = ttk.Combobox(

256 master=self.window,

257 textvariable=self.ending\_show

258 )

259 self.select\_box.place(x=100, y=100)

260 self.select\_box['values'] = ['智能分析', '普通分析']

261 self.select\_box.bind("<<ComboboxSelected>>", self.\_get\_ending)

262

263 *# label boss2me*

264 Label(self.window, text="original file:").place(x=100, y=150)

265 Entry(self.window, textvariable=self.boss2me\_path, width='60').place(x=220, y=150)

266 Button(self.window, text="select", command=self.\_get\_boss2me, width='10').place(x=680, y=150)

267

268 *# label all\_download*

269 Label(self.window, text="all report file:").place(x=100, y=200)

270 Entry(self.window, textvariable=self.all\_report\_path, width='60').place(x=220, y=200)

271 Button(self.window, text="select", command=self.\_get\_all\_report, width='10').place(x=680, y=200)

272

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.*

17  *num\_conv (int): Number of convolution layers in the body network. Default: 16.*

18  *upscale (int): Upsampling factor. Default: 4.*

19  *act\_type (str): Activation type, options: 'relu', 'prelu', 'leakyrelu'. Default: prelu.*

20  *"""*

21

22 def \_\_init\_\_(self, num\_in\_ch=3, num\_out\_ch=3, num\_feat=64, num\_conv=16, upscale=4, act\_type='prelu'):

23 super(SRVGGNetCompact, self).\_\_init\_\_()

24 self.num\_in\_ch = num\_in\_ch

25 self.num\_out\_ch = num\_out\_ch

26 self.num\_feat = num\_feat

27 self.num\_conv = num\_conv

28 self.upscale = upscale

29 self.act\_type = act\_type

30

31 self.body = nn.ModuleList()

32 *# the first conv*

33 self.body.append(nn.Conv2d(num\_in\_ch, num\_feat, 3, 1, 1))

34 *# the first activation*

35 if act\_type == 'relu':

36 activation = nn.ReLU(inplace=True)

37 elif act\_type == 'prelu':

38 activation = nn.PReLU(num\_parameters=num\_feat)

39 elif act\_type == 'leakyrelu':

40 activation = nn.LeakyReLU(negative\_slope=0.1, inplace=True)

41 self.body.append(activation)

42

43 *# the body structure*

44 for \_ in range(num\_conv):

45 self.body.append(nn.Conv2d(num\_feat, num\_feat, 3, 1, 1))

46 *# activation*

47 if act\_type == 'relu':

48 activation = nn.ReLU(inplace=True)

49 elif act\_type == 'prelu':

50 activation = nn.PReLU(num\_parameters=num\_feat)

51 elif act\_type == 'leakyrelu':

52 activation = nn.LeakyReLU(negative\_slope=0.1, inplace=True)

53 self.body.append(activation)

54

55 *# the last conv*

56 self.body.append(nn.Conv2d(num\_feat, num\_out\_ch \* upscale \* upscale, 3, 1, 1))

57 *# upsample*

58 self.upsampler = nn.PixelShuffle(upscale)

59

60 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