

```

1  #Script di unione dei file, prima iterazione analisi video
2
3  # -*- coding: utf-8 -*-
4  """
5  Created on Mon Jul  9 14:56:43 2018
6
7  @author: Alberto
8  """
9
10 import argparse
11 import pandas as pd
12 import glob
13
14 parser = argparse.ArgumentParser(
15     description='Inserire il nome della cartella dei file')
16 parser.add_argument('file_time', help='Nome cartella dei file')
17 args = parser.parse_args()
18
19 sensor_path = args.file_time + "/can" + args.file_time + ".csv"
20 avm_path = args.file_time + "/avm" + args.file_time + ".csv"
21 pax_path = args.file_time + "/pax" + args.file_time + ".xlsx"
22
23 df_sensor = pd.read_csv(sensor_path, sep = ";", skipinitialspace=True,
24     usecols = ["TIME ", "MESSAGEID ", "VALUE"],
25     converters = {"VALUE": lambda x: int(x, 16)})
26 df_sensor.columns = df_sensor.columns.str.strip()
27 df_sensor = df_sensor[(df_sensor["VALUE"] <= 10000) & (df_sensor["VALUE"] >= 1000)]
28 df_sensor = df_sensor.drop_duplicates().reset_index(drop=True).rename(index=str,
29     columns={"TIME" : "TIMESTAMP"})
30
31 df_avm = pd.read_csv(avm_path, sep = ";", skipinitialspace=True,
32     usecols = ["TIMESTAMP ", "DOORS "])
33 df_avm.columns = df_avm.columns.str.strip()
34 df_avm = df_avm.drop_duplicates().reset_index(drop=True)
35
36 pax_excel = pd.ExcelFile(pax_path)
37
38 video_path = "analisi_video/"
39 video_files = glob.glob(video_path + "*.xlsx")
40 df_list_video = [pd.DataFrame(columns=['CAM1', 'TIMESTAMP']),
41     pd.DataFrame(columns=['CAM2', 'TIMESTAMP']),
42     pd.DataFrame(columns=['CAM3', 'TIMESTAMP'])]
43
44 for file_name in video_files:
45     file_name_cut = file_name[14:]
46     if df_avm.iloc[0,0] <= int(file_name_cut[:10]) <= df_avm.iloc[-1,0]:
47         cam = int(file_name_cut[14])
48         df_video = pd.ExcelFile(video_path + file_name_cut).parse(sheet_name = 0,
49             usecols = ("B", "E")).astype(int)
50         df_video.columns = ['CAM' + str(cam), 'TIMESTAMP']
51         df_list_video[cam-1] = df_list_video[cam-1].append(df_video,
52             ignore_index=True)
53
54 pos = 1
55 while pos < len(df_avm.index):
56     if ((df_avm.iloc[pos]["TIMESTAMP"]) - (df_avm.iloc[pos-1]["TIMESTAMP"]))>1:
57         new_timestamp = df_avm.iloc[pos-1]["TIMESTAMP"] + 1
58         new_doors = df_avm.iloc[pos-1]["DOORS"]
59         df_avm = df_avm.append(pd.Series([new_timestamp, new_doors],
60             index=["TIMESTAMP", "DOORS"]),
61             ignore_index=True).sort_values(by=["TIMESTAMP"])
62     pos += 1
63
64 for df_cam in df_list_video:
65     df_cam["TIMESTAMP"] = df_cam["TIMESTAMP"].apply(int)
66     df_avm = df_avm.merge(df_cam, on = "TIMESTAMP", how = "left")
67
68 open_start = 0
69 open_end = 0
70 if (df_avm.iloc[0]["DOORS"] == 0):
71     open_start = df_avm.iloc[0]["TIMESTAMP"]
72 lenght = len(df_avm.index)
73 df_avm["CHANGE"] = ""

```

```

66 for x in range (1, lenght):
67     if (df_avm.iloc[x]["DOORS"] != df_avm.iloc[x-1]["DOORS"]):
68         if (df_avm.iloc[x]["DOORS"] == 0):
69             open_start = df_avm.iloc[x]["TIMESTAMP"]
70         else:
71             df_avm.at[x, "CHANGE"] = 18000
72             if (open_start != 0):
73                 open_end = df_avm.iloc[x]["TIMESTAMP"]
74                 for x in range (1,4):
75                     cam = "CAM" + str(x)
76                     mode_calc = df_avm[(df_avm["TIMESTAMP"] >= open_start) &
77                                     (df_avm["TIMESTAMP"] <= open_end)][cam].mode()
78                     if (mode_calc.empty):
79                         df_avm[(df_avm["TIMESTAMP"] >= open_start) &
80                             (df_avm["TIMESTAMP"] <= open_end)][cam] = 0
81                     else:
82                         df_avm[(df_avm["TIMESTAMP"] >= open_start) &
83                             (df_avm["TIMESTAMP"] <= open_end)][cam] = mode_calc[0]
84                 open_start = 0
85 df_avm = df_avm.fillna(method = "ffill")
86 df_avm = df_avm[(df_avm["DOORS"] == 1)]
87
88 df_result = df_avm
89
90 for sensor_number in range(1, 5):
91     df_sensor_single = df_sensor[df_sensor["MESSAGEID"] ==
92     (180+sensor_number)].drop("MESSAGEID",
93     1).groupby("TIMESTAMP").max().reset_index().rename(index=str, columns={"VALUE" :
94     "SENSOR_" + str(sensor_number)})
95     df_result = df_result.merge(df_sensor_single, on = "TIMESTAMP", how ="left")
96
97 df_result = df_result.fillna(method="ffill")
98 col_sum = df_result.loc[:, "SENSOR_1":"SENSOR_4"].sum(axis = 1)
99 df_result["SUM"] = col_sum
100 df_result["SUM_FILTER"] = col_sum
101
102 lenght = len(df_result.index)
103 for x in range (3, lenght):
104     if (df_avm.iloc[x]["CHANGE"] == 18000):
105         df_result.at[x-2, "SUM_FILTER"] = df_result.iloc[x-3]["SUM"]
106         df_result.at[x-1, "SUM_FILTER"] = df_result.iloc[x-3]["SUM"]
107
108 writer = pd.ExcelWriter(args.file_time + "/results_complete_rides" + args.file_time
109 + ".xlsx", engine="xlsxwriter")
110
111 for x in range ((len(pax_excel.sheet_names)-1), -1, -1):
112     df_pax = pax_excel.parse(sheet_name = x, usecols =("B, I:Q"))
113     df_pax = df_pax.drop_duplicates().reset_index(drop=True)
114     df_pax.columns = df_pax.columns.str.strip()
115     df_result_temp = df_result[(df_result["TIMESTAMP"] >=
116     df_pax.iloc[0]["TIMESTAMP"]) & (df_result["TIMESTAMP"] <=
117     df_pax.iloc[-1]["TIMESTAMP"])]
118     df_result_temp = df_result_temp.merge(df_pax, on = "TIMESTAMP", how ="left")
119     df_result_move = df_result_temp[["TIMESTAMP", "CAM1", "CAM2", "CAM3"]]
120     df_result_temp.drop(labels=["CAM1", "CAM2", "CAM3"], axis=1,inplace = True)
121     df_result_temp = df_result_temp.merge(df_result_move, on = "TIMESTAMP", how
122     ="left")
123     df_result_temp = df_result_temp.fillna(method="ffill")
124     col_video_sum = df_result_temp.loc[:, "CAM1":"CAM3"].sum(axis = 1)
125     df_result_temp["VIDEO_SUM"] = col_video_sum
126
127     if (x == 0):
128         result_path = args.file_time + "/results_complete" + args.file_time + ".csv"
129     else:
130         result_path = args.file_time + "/results_ride" + args.file_time + "[ride " +
131         str(len(pax_excel.sheet_names)-x) +"].csv"
132         df_result_temp.to_excel(writer, sheet_name = "Corsa" +
133         str(len(pax_excel.sheet_names)-x), index = False, freeze_panes = (1, 0))
134
135     df_result_temp.to_csv(result_path, sep=";", index=False, float_format='%0.0f')
136
137 writer.save()
138 writer.close()

```

```

126
127
128 #Script di unione dei file, seconda iterazione analisi video
129
130 # -*- coding: utf-8 -*-
131 """
132 Created on Mon Jul 9 14:56:43 2018
133
134 @author: Alberto
135 """
136
137 import argparse
138 import pandas as pd
139 import glob
140
141 parser = argparse.ArgumentParser(
142     description='Inserire il nome della cartella dei file')
143 parser.add_argument('file_time', help='Nome cartella dei file')
144 args = parser.parse_args()
145
146 sensor_path = args.file_time + "/can" + args.file_time + ".csv"
147 avm_path = args.file_time + "/avm" + args.file_time + ".csv"
148 pax_path = args.file_time + "/pax" + args.file_time + ".xlsx"
149
150 df_sensor = pd.read_csv(sensor_path, sep = ";", skipinitialspace=True,
151     usecols = ["TIME ", "MESSAGEID ", "VALUE"],
152     converters = {"VALUE": lambda x: int(x, 16)})
153 df_sensor.columns = df_sensor.columns.str.strip()
154 df_sensor = df_sensor[(df_sensor["VALUE"] <= 10000) & (df_sensor["VALUE"] >= 1000)]
155 df_sensor = df_sensor.drop_duplicates().reset_index(drop=True).rename(index=str,
156     columns={"TIME" : "TIMESTAMP"})
157
158 df_avm = pd.read_csv(avm_path, sep = ";", skipinitialspace=True,
159     usecols=["TIMESTAMP ", "DOORS "])
160 df_avm.columns = df_avm.columns.str.strip()
161 df_avm = df_avm.drop_duplicates().reset_index(drop=True)
162
163 pax_excel = pd.ExcelFile(pax_path)
164
165 video_path = "analisi_video/"
166 video_files = glob.glob(video_path + "*.xlsx")
167 df_list_video = [pd.DataFrame(columns=['CAM1', 'TIMESTAMP']),
168     pd.DataFrame(columns=['CAM2', 'TIMESTAMP']),
169     pd.DataFrame(columns=['CAM3', 'TIMESTAMP'])]
170
171 for file_name in video_files:
172     file_name_cut = file_name[14:]
173     if df_avm.iloc[0,0] <= int(file_name_cut[:10]) <= df_avm.iloc[-1,0]:
174         cam = int(file_name_cut[14])
175         df_video = pd.ExcelFile(video_path + file_name_cut).parse(sheet_name = 0,
176             usecols = ("B, E")).astype(int)
177         df_video.columns = ['CAM' + str(cam), 'TIMESTAMP']
178         df_list_video[cam-1] = df_list_video[cam-1].append(df_video,
179             ignore_index=True)
180
181 pos = 1
182 while pos < len(df_avm.index):
183     if ((df_avm.iloc[pos]["TIMESTAMP"]) - (df_avm.iloc[pos-1]["TIMESTAMP"]))>1:
184         new_timestamp = df_avm.iloc[pos-1]["TIMESTAMP"] + 1
185         new_doors = df_avm.iloc[pos-1]["DOORS"]
186         df_avm = df_avm.append(pd.Series([new_timestamp, new_doors],
187             index=["TIMESTAMP", "DOORS"]),
188             ignore_index=True).sort_values(by=["TIMESTAMP"])
189     pos += 1
190
191 for df_cam in df_list_video:
192     df_cam["TIMESTAMP"] = df_cam["TIMESTAMP"].apply(int)
193     df_avm = df_avm.merge(df_cam, on = "TIMESTAMP", how = "left")
194
195 #open_start = 0
196 #open_end = 0
197 #if (df_avm.iloc[0]["DOORS"] == 0):
198 #    open_start = df_avm.iloc[0]["TIMESTAMP"]

```

```

191 lenght = len(df_avm.index)
192 df_avm["CHANGE"] = ""
193 for x in range (1, lenght):
194     if (df_avm.iloc[x]["DOORS"] != df_avm.iloc[x-1]["DOORS"]):
195         #         if (df_avm.iloc[x]["DOORS"] == 0):
196         #             open_start = df_avm.iloc[x]["TIMESTAMP"]
197         #         else:
198         if (df_avm.iloc[x]["DOORS"] != 0):
199             df_avm.at[x, "CHANGE"] = 18000
200         #             df_avm.at[x, "CHANGE"] = 18000
201         #             if (open_start != 0):
202         #                 open_end = df_avm.iloc[x]["TIMESTAMP"]
203         #                 for x in range (1,4):
204         #                     cam = "CAM" + str(x)
205         #                     mode_calc = df_avm[(df_avm["TIMESTAMP"] >= open_start) &
206         # (df_avm["TIMESTAMP"] <= open_end)][cam].mode()
207         #                     if (mode_calc.empty):
208         #                         df_avm[(df_avm["TIMESTAMP"] >= open_start) &
209         # (df_avm["TIMESTAMP"] <= open_end)][cam] = 0
210         #                     else:
211         #                         df_avm[(df_avm["TIMESTAMP"] >= open_start) &
212         # (df_avm["TIMESTAMP"] <= open_end)][cam] = mode_calc[0]
213         #                 open_start = 0
214 df_avm = df_avm.fillna(method = "ffill")
215 df_avm = df_avm[(df_avm["DOORS"] == 1)]
216
217 df_result = df_avm
218 for sensor_number in range(1, 5):
219     df_sensor_single = df_sensor[df_sensor["MESSAGEID"] ==
220     (180+sensor_number)].drop("MESSAGEID",
221     1).groupby("TIMESTAMP").max().reset_index().rename(index=str, columns={"VALUE" :
222     "SENSOR_" + str(sensor_number)})
223     df_result = df_result.merge(df_sensor_single, on = "TIMESTAMP", how ="left")
224
225 df_result = df_result.fillna(method="ffill")
226 col_sum = df_result.loc[:, "SENSOR_1":"SENSOR_4"].sum(axis = 1)
227 df_result["SUM"] = col_sum
228 df_result["SUM_FILTER"] = col_sum
229
230 lenght = len(df_result.index)
231 for x in range (3, lenght):
232     if (df_avm.iloc[x]["CHANGE"] == 18000):
233         df_result.at[x-2, "SUM_FILTER"] = df_result.iloc[x-3]["SUM"]
234         df_result.at[x-1, "SUM_FILTER"] = df_result.iloc[x-3]["SUM"]
235
236 writer = pd.ExcelWriter(args.file_time + "/results_complete_rides" + args.file_time
237 + ".xlsx", engine="xlsxwriter")
238
239 for x in range ((len(pax_excel.sheet_names)-1), -1, -1):
240     df_pax = pax_excel.parse(sheet_name = x, usecols =("B, I:Q"))
241     df_pax = df_pax.drop_duplicates().reset_index(drop=True)
242     df_pax.columns = df_pax.columns.str.strip()
243     df_result_temp = df_result[(df_result["TIMESTAMP"] >=
244     df_pax.iloc[0]["TIMESTAMP"]) & (df_result["TIMESTAMP"] <=
245     df_pax.iloc[-1]["TIMESTAMP"])]
246     df_result_temp = df_result_temp.merge(df_pax, on = "TIMESTAMP", how ="left")
247     df_result_move = df_result_temp[["TIMESTAMP", "CAM1", "CAM2", "CAM3"]]
248     df_result_temp.drop(labels=["CAM1", "CAM2", "CAM3"], axis=1,inplace = True)
249     df_result_temp = df_result_temp.merge(df_result_move, on = "TIMESTAMP", how
250     ="left")
251     df_result_temp = df_result_temp.fillna(method="ffill")
252     col_video_sum = df_result_temp.loc[:, "CAM1":"CAM3"].sum(axis = 1)
253     df_result_temp["VIDEO_SUM"] = col_video_sum
254
255     if (x == 0):
256         result_path = args.file_time + "/results_complete" + args.file_time + ".csv"
257     else:
258         result_path = args.file_time + "/results_ride" + args.file_time + "[ride " +
259         str(len(pax_excel.sheet_names)-x) + "].csv"
260     df_result_temp.to_excel(writer, sheet_name = "Corsa" +
261     str(len(pax_excel.sheet_names)-x), index = False, freeze_panes = (1, 0))

```

```
251         df_result_temp.to_csv(result_path, sep=";", index=False, float_format='%0.0f')
252
253     writer.save()
254     writer.close()
```