stacInterv23_2022

May 20, 2025

1 Trazenje stacionarnih intervala

1.1 U ovom jupyter notebooku nalazimo intervale u kojima je zone_fan_speed u stacionarnom stanju(nije se mijenjao određeno vrijeme; mi smo odabrali 1 sat)

1.1.1 Učitavanje filtriranih podataka za kaloritemar i zone

[3]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

[4]:
```
file_path = "../../../results/2022/23/calorimeter_23_year_2022filtriranje.csv"
file_path1="../../../results/2022/23/zones_23_year_2022filtriranje.csv"

data = pd.read_csv(file_path, index_col=[0])
data1=pd.read_csv(file_path1, index_col=[0])
```

1.1.2 Odredivanje intervala i odredivanje njegove duljine

[6]:
```
import pandas as pd

# pretvorba u datetime format
data1['timestamp'] = pd.to_datetime(data1['timestamp'])

all_intervals = []

zone_ids = list(range(193, 195)) + list(range(208, 213))

for zone_id in zone_ids:
    # filtriramo za trenuto zone_id
    zone_data = data1[data1['zone_id'] == zone_id]

    # Sortirmo po timestampu
    zone_data_sorted = zone_data.sort_values(by='timestamp')

    # grupiramo po promjenama u  zone_fan_speed
```

```
    zone_data_sorted['change'] = zone_data_sorted['zone_fan_speed'] !=␣
 ↪zone_data_sorted['zone_fan_speed'].shift()
    zone_data_sorted['group'] = zone_data_sorted['change'].cumsum()

    # Zdruzimo kako bi dobili imtervale
    intervals = zone_data_sorted.groupby('group').agg(
        start_time=('timestamp', 'first'),
        end_time=('timestamp', 'last'),
        zone_fan_speed=('zone_fan_speed', 'first')
    ).reset_index(drop=True)

    # racunamo trajanje intervala
    intervals['duration'] = (intervals['end_time'] - intervals['start_time']).
 ↪dt.total_seconds() / 3600

    # dodajemo zone_id
    intervals['zone_id'] = zone_id

    # dodajemo u listi intervala
    all_intervals.append(intervals)

# pretvaramo u dataframa
all_intervals_df = pd.concat(all_intervals, ignore_index=True)


print("Gotovo")
```

```
Gotovo
```

1.1.3 Uklanjanje duplikata i filtriranje duljih od 1h

[8]:
```
# stvaramo dataframe od all_intervals
all_intervals_df = pd.concat(all_intervals, ignore_index=True)

# pretvorba u datetime format
all_intervals_df['start_time'] = pd.to_datetime(all_intervals_df['start_time'])
all_intervals_df['end_time'] = pd.to_datetime(all_intervals_df['end_time'])

# filtriranje intervala duljih od 1h
all_intervals_df['duration'] = all_intervals_df['end_time'] -␣
 ↪all_intervals_df['start_time']
filtered_intervals_df = all_intervals_df[all_intervals_df['duration'] >= pd.
 ↪Timedelta(hours=1)]

# micemo duplikate
unique_intervals = filtered_intervals_df[['start_time', 'end_time', 'zone_id']].
 ↪drop_duplicates()
```

```
print("Gotovo'")
```

```
Gotovo'
```

1.1.4 Trazenje presjeka kod intervala

[10]:
```
intersections = []


for _, interval in unique_intervals.iterrows():
    start = interval['start_time']
    end = interval['end_time']
    current_zone = interval['zone_id']

    # gledmo ima li interval presjek u drugim zonama
    overlapping_zones = filtered_intervals_df[
        (filtered_intervals_df['zone_id'] != current_zone) &  # Exclude current␣
 ↪zone
        (filtered_intervals_df['start_time'] <= end) &
        (filtered_intervals_df['end_time'] >= start)
    ]

    # provjera ima li presjek u SVIM zonama
    covered_zones = set(overlapping_zones['zone_id'])
    required_zones =  set(range(193, 195)).union(set(range(208, 213))) -␣
 ↪{current_zone}

    if required_zones.issubset(covered_zones):  # ako je presjek svih zona␣
 ↪dodaj u tablicu

        intersection_start = max(overlapping_zones['start_time'].min(), start)
        intersection_end = min(overlapping_zones['end_time'].max(), end)
        intersection_duration = intersection_end - intersection_start

        # pohrani rezultat
        intersections.append({
            'zone_id': current_zone,  # Store the reference zone
            'intersection_start': intersection_start,
            'intersection_end': intersection_end,
            'intersection_duration': intersection_duration
        })


intersections_df = pd.DataFrame(intersections)
```

```
intersections_df.to_csv('../../../results/2021/23/
 ↪calorimeter_23_year_2021presjeci.csv ', index=False)

print("Gotovo'")
```

```
Gotovo'
```

1.1.5 Uredivanje tablice za intervale

[12]:
```
# Fitracija za zone_id 195
intersections_df = intersections_df[intersections_df['zone_id'] != 193]

# micem zone_id
intersections_df = intersections_df.drop(columns=['zone_id'])

# Spremanje rezultata
intersections_df.to_csv( '../../../results/2022/23/
 ↪calorimeter_23_year_2022presjeci.csv ', index=False)
print("Gotov")
```

```
Gotov
```

[ ]:
```

```