dde

December 20, 2021

[ ]:
```
import numpy as np
import pandas as pd
from datetime import date
import plotly.express as px
import plotly.graph_objects as go
```

adwdwadfaf

[ ]:
```
def set_timestamp_index(df,time_col):
    """
    Creates a timestamp (datetime-object) column and sets it as an index
    """

    df['timestamp'] = df[time_col].apply(lambda x: pd.Timestamp(x).
 ↪tz_convert("Europe/Berlin"))
    df = df.set_index(['timestamp'])
    df.drop(time_col,axis=1,inplace=True)

    df['day'] = df.index.day
    df['month'] = df.index.month_name()
    df['year'] = df.index.year
    df['date'] = df.index.date
    return df

def reorder_columns(df,order):
    """
    Reorders the columns by given order
    """

    new_order= order + [x for x in df.columns if x not in order]
    df = df.reindex(columns=new_order)#axis='columns',)
    return df

def remove_duplicates(df):
    """Removes duplicate indecies from dataframe"""

    df = df[~df.index.duplicated(keep='first')]
    return df
```

```
def process_weather_csv():
    """
    Loads weather_features.csv and preprocesses the data
    """

    df_weather = pd.read_csv('weather_features.csv')
    df_weather = set_timestamp_index(df_weather,'dt_iso')
    cities = df_weather.city_name.unique()
    df_cities = [df_weather.query(f"city_name == '{x}'") for x in cities]
    return df_cities

def process_energy_csv():
    """
    Loads energy_dataset.csv and preprocesses the data
    """

    df_energy = pd.read_csv('energy_dataset.csv')
    df_energy = set_timestamp_index(df_energy,'time')
    return df_energy

def average_over_year(df):
    """
    Filters the dataframe and groups them by their month day and averages them.
    Returns list with dataframes for each month
    """

    months_list =␣
 ↪['January','February','March','April','May','June','July','August','September','October','N
    return [df.query(f"month == '{x}'").groupby('day').mean() for x in␣
 ↪months_list]

def average_over_day(df):
    """
    Filters the dataframe and groups them by their specific data and averages␣
 ↪them.
    Returns list with dataframe for each month and every year.
    """

    months_list =␣
 ↪['January','February','March','April','May','June','July','August','September','October','N
    return [df.query(f"month == '{x}'").groupby('date').mean() for x in␣
 ↪months_list]

months_list =␣
 ↪['January','February','March','April','May','June','July','August','September','October','N
```

```
df_cities = process_weather_csv()
```

[ ]:
```
city_names = []
for item in df_cities:
    city_names.append(str(item['city_name'].unique()))
print(city_names)
for i, city in enumerate(df_cities):
    df_cities[i] = remove_duplicates(city)
df_valencia = df_cities[0]
df_valencia_list = average_over_year(df_valencia)
df_madrid_list = average_over_year(df_cities[1])
```

```
["['Valencia']", "['Madrid']", "['Bilbao']", "[' Barcelona']", "['Seville']"]
```

[ ]:
```
# display double timestamps in dataframe
def display_double_timestamps():
    """
    Returns double Timestamps from Weather Dataset
    """
    df_cities = process_weather_csv()
    df_city_dup=[]
    for i,item in enumerate(df_cities):
        df_city_dup.append(item[item.index.duplicated(keep=False)])
    return df_city_dup
```

[ ]:
```
def display_monthly_averaged_data(df_list,description,feature):
    fig = px.line()
    for item,desc in zip(df_list,description):
        fig.add_traces(go.Scatter(x=item.index,y=item[feature],name=desc))

    fig.show()
```

[ ]:
```
display_monthly_averaged_data(df_madrid_list,months_list,'temp')
```

[ ]:
```
display_monthly_averaged_data(df_valencia_list,months_list,'temp')
```

[ ]:
```

```