# Cad-phase3

[3]:
```
# This Python 3 environment comes with many helpful analytics
 libraries␣ ↪installed
# It is defined by the kaggle/python Docker image:
 https://github.com/kaggle/ ↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing Shift+Enter)
 will list␣ ↪all files under the input directory

import os for dirname, _, filenames in
os.walk('HomeC.csv'): for filename in
filenames:
       print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/)
that␣
 ↪gets preserved as output when you create a version using "Save & Run
 All"
# You can also write temporary files to /kaggle/temp/, but they won't
 be saved␣ ↪outside of the current session
```

## 1 Data Exploration

### 1.0.1 Data reading

[9]:
```
df = pd.read_csv(os.path.join('HomeC.csv'), low_memory=False)
df.head()
```

[9]:
```
        time use [kW] gen [kW] House overall [kW] Dishwasher [kW] \
0 1451624400 0.932833 0.003483          0.932833        0.000033
1 1451624401 0.934333 0.003467          0.934333        0.000000
2 1451624402 0.931817 0.003467          0.931817        0.000017
3 1451624403 1.022050 0.003483          1.022050        0.000017
4 1451624404 1.139400 0.003467          1.139400        0.000133
   Furnace 1 [kW] Furnace 2 [kW] Home office [kW] Fridge [kW] \
```

```
0       0.020700 0.061917   0.442633   0.124150

1       0.020717 0.063817   0.444067   0.124000

2       0.020700 0.062317   0.446067   0.123533

3       0.106900 0.068517   0.446583   0.123133

4       0.236933 0.063983   0.446533   0.122850

        Wine cellar [kW] … visibility summary apparentTemperature pressure
        \
     0         0.006983 …       10.0   Clear             29.26  1016.91
     1         0.006983 …       10.0   Clear             29.26  1016.91
     2         0.006983 …       10.0   Clear             29.26  1016.91
     3         0.006983 …       10.0   Clear             29.26  1016.91
     4         0.006850 …       10.0   Clear             29.26  1016.91
        windSpeed cloudCover windBearing precipIntensity dewPoint \
     0     9.18 cloudCover  282.0 0.0   24.4 1     9.18
     cloudCover 282.0 0.0   24.4 2     9.18 cloudCover  282.0
     0.0   24.4 3     9.18 cloudCover  282.0 0.0   24.4
     4      9.18 cloudCover      282.0            0.0     24.4

        precipProbability
     0     0.0 1 0.0 2
     0.0 3 0.0
     4              0.0

     [5 rows x 32 columns]
```

[10]:
```
df.columns
```

```
[10]: Index(['time', 'use [kW]', 'gen [kW]', 'House overall [kW]',
'Dishwasher [kW]',
           'Furnace 1 [kW]', 'Furnace 2 [kW]', 'Home office [kW]', 'Fridge
           [kW]',
           'Wine cellar [kW]', 'Garage door [kW]', 'Kitchen 12 [kW]',
           'Kitchen 14 [kW]', 'Kitchen 38 [kW]', 'Barn [kW]', 'Well [kW]',
           'Microwave [kW]', 'Living room [kW]', 'Solar [kW]',
           'temperature',
           'icon', 'humidity', 'visibility', 'summary',
           'apparentTemperature',
           'pressure', 'windSpeed', 'cloudCover', 'windBearing',
           'precipIntensity',
           'dewPoint', 'precipProbability'],
          dtype='object')
```

### 1.0.2 Data Preprocessing

[11]:
```
# Rename columns to remove spaces and the kW unit df.columns =
[col[:-5].replace(' ','_') if 'kW' in col else col for col in df.
↪columns]

# Drop rows with nan
values df = df.dropna()

# The columns "use" and "house_overall" are the same, so let's remove
the␣
 ↪'house_overall' column
df.drop(['House_overall'], axis=1,
inplace=True)

# The columns "gen" and "solar" are the same, so let's remove the
'solar' column df.drop(['Solar'], axis=1, inplace=True)

# drop rows with cloudCover column values that are not numeric (bug in
sensors)␣
 ↪and convert column to numeric df =
df[df['cloudCover']!='cloudCover']
df["cloudCover"] =
pd.to_numeric(df["cloudCover"])

# Create columns that regroup kitchens and furnaces
df['kitchen'] = df['Kitchen_12'] + df['Kitchen_14'] + df['Kitchen_38']
df['Furnace'] = df['Furnace_1'] + df['Furnace_2']

# Convert "time" column (which is a unix timestamp) to a Y-m-d H-M-
S import time start_time = time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(int(df['time'].
 ↪iloc[0]))) time_index = pd.date_range(start_time,
periods=len(df), freq='min') time_index =
pd.DatetimeIndex(time_index) df = df.set_index(time_index)
df = df.drop(['time'], axis=1)
```

### 1.0.3 Data Analysis

[5]:
```
df.shape
```

[5]: (503852, 31)

[6]:
```
df.columns
```

[6]: Index(['use', 'gen', 'Dishwasher', 'Furnace_1', 'Furnace_2',
'Home_office',

```
       'Fridge', 'Wine_cellar', 'Garage_door', 'Kitchen_12',
       'Kitchen_14',
'Kitchen_38', 'Barn', 'Well', 'Microwave', 'Living_room', 'temperature',
       'icon', 'humidity', 'visibility', 'summary',
       'apparentTemperature',
'pressure', 'windSpeed', 'cloudCover', 'windBearing', 'precipIntensity',
       'dewPoint', 'precipProbability', 'kitchen', 'Furnace'],
      dtype='object')
```

[7]:
```
# lower frist letter of a string
func = lambda s: s[:1].lower() + s[1:] if s else ''
```

[8]:
```
cols = list(df.dtypes.keys())
categ_cols = [col for col in cols if df[col].dtype=='O']
num_cols = [col for col in cols if col not in categ_cols]
print('categ_cols : ', categ_cols)
print('num_cols : ', num_cols)
```

```
categ_cols :  ['icon', 'summary'] num_cols :  ['use', 'gen',
'Dishwasher', 'Furnace_1', 'Furnace_2', 'Home_office',
'Fridge', 'Wine_cellar', 'Garage_door', 'Kitchen_12',
'Kitchen_14', 'Kitchen_38', 'Barn', 'Well', 'Microwave',
'Living_room',
'temperature', 'humidity', 'visibility', 'apparentTemperature',
'pressure',
'windSpeed', 'cloudCover', 'windBearing', 'precipIntensity',
'dewPoint',
'precipProbability', 'kitchen', 'Furnace']
```

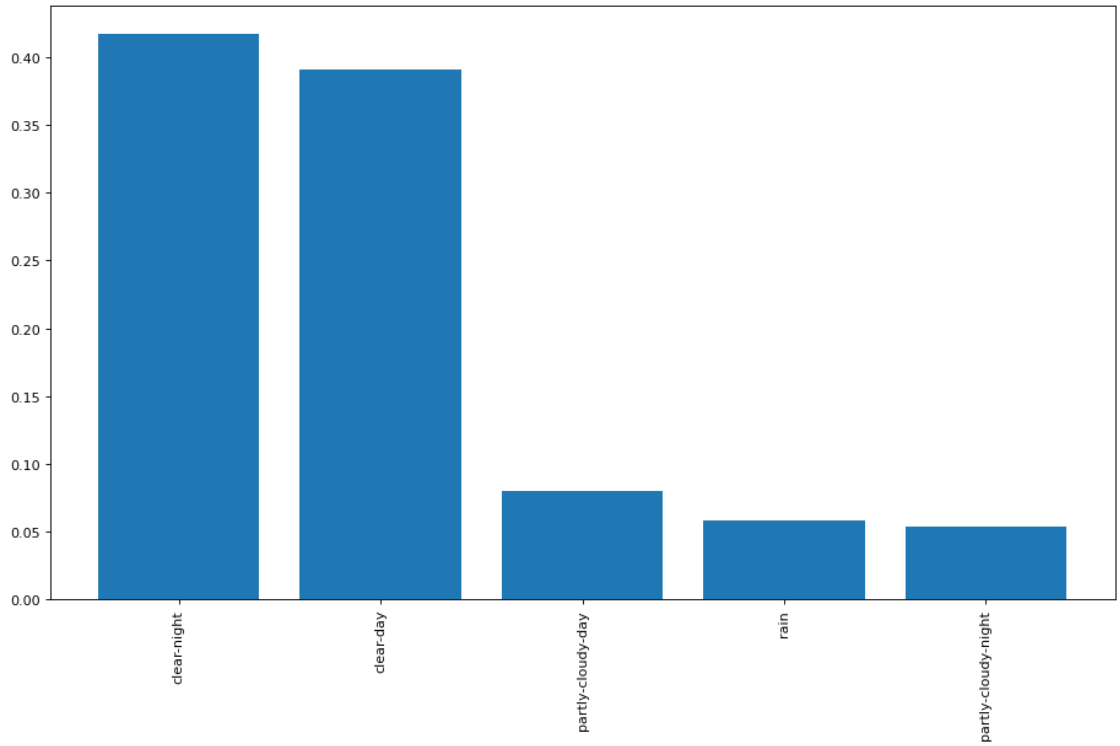
[9]:
```
# Let's remove rows with values that appear less than a certain percentage %

def remove_less_percent(col, percent):
    keys_to_conserve = [key for key,value in df[col].
 ↪value_counts(normalize=True).items() if value>=percent]
    return df[df[col].isin(keys_to_conserve)]

print(len(df))
df = remove_less_percent('summary', 0.05)
print(len(df))
df = remove_less_percent('icon', 0.05)
print(len(df))
```

```
503852
466308
```

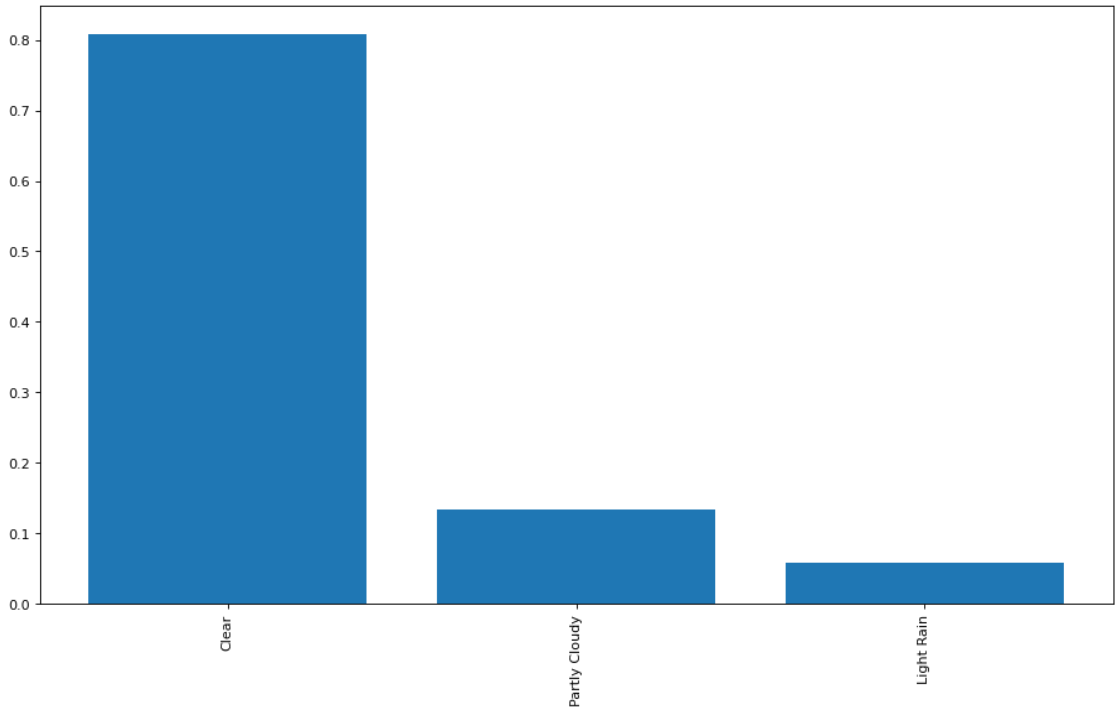
```
466308
```

[10]:
```
# plot bars of unique values of categorical columns

def plot_bars(col):

    import matplotlib.pyplot as plt
    from matplotlib.pyplot import figure

    figure(figsize=(14, 8), dpi=80)
    plt.xticks(rotation = 90)

    D = df[col].value_counts(normalize=True).to_dict()

    plt.bar(*zip(*D.items()))
    plt.show()
```

[11]:
```
plot_bars('icon')
```

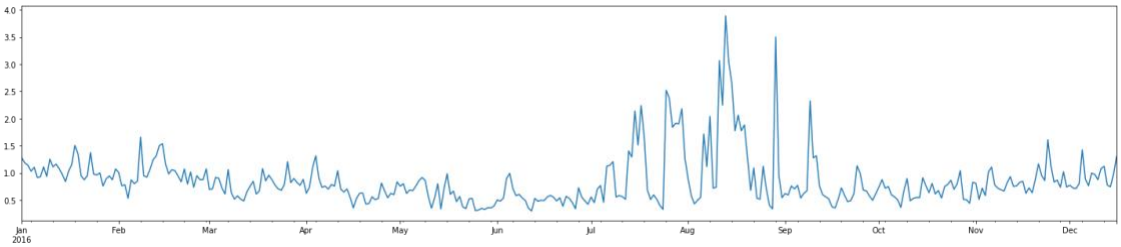



[12]:
```
plot_bars('summary')
```



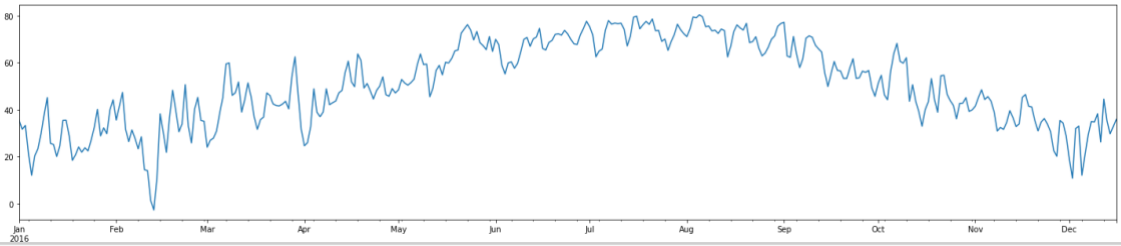

[13]:
```
df['use'].resample(rule='D').mean().plot(figsize=(25,5))
```

[13]: <AxesSubplot:>



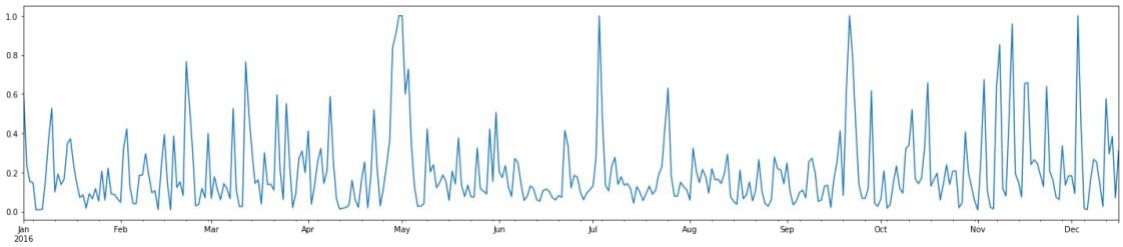

[14]:
```
df['temperature'].resample(rule='D').mean().plot(figsize=(25,5))
```

[14]: <AxesSubplot:>



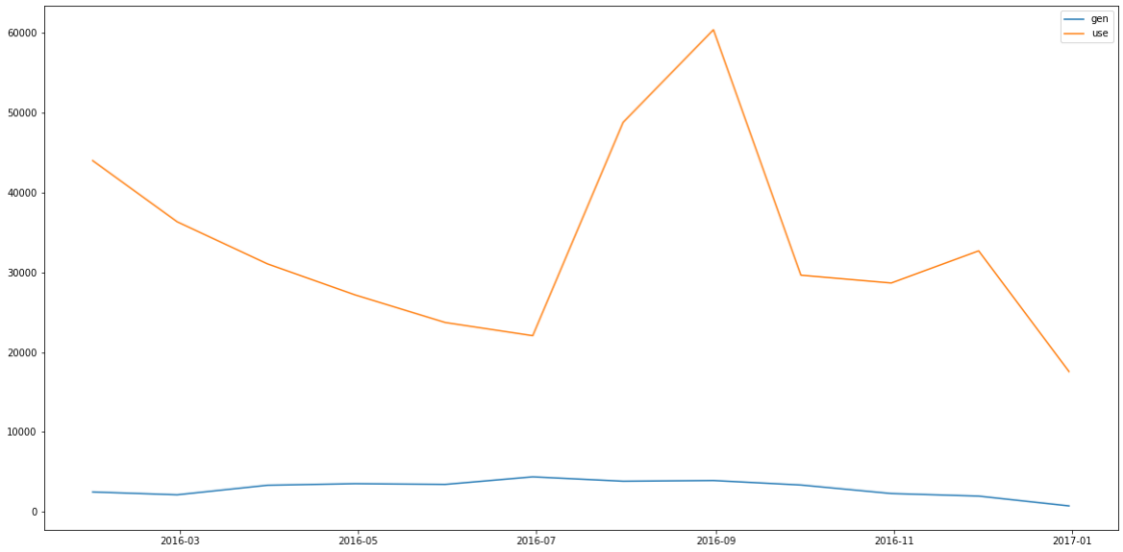

[15]:
```
df['cloudCover'].resample(rule='D').mean().plot(figsize=(25,5))
```

[15]: <AxesSubplot:>



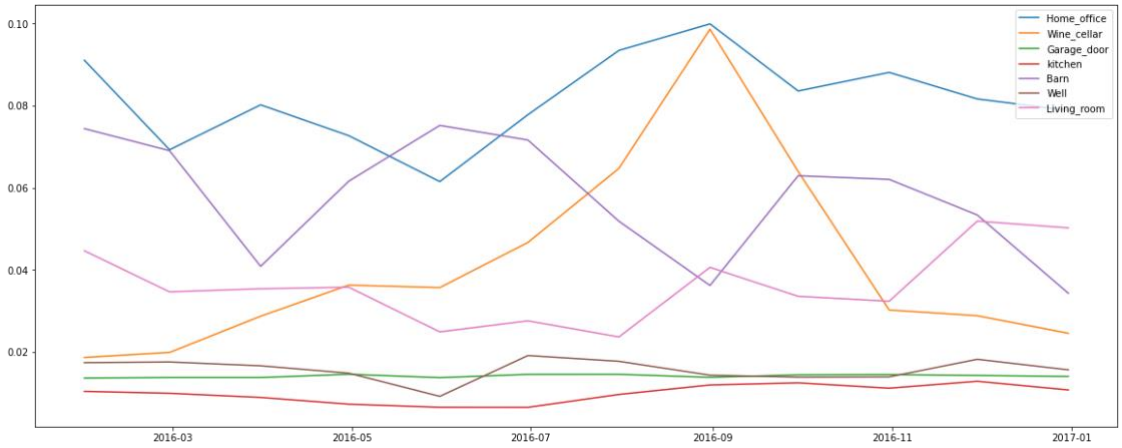

[16]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

general_energy_cols = ['gen', 'use']
general_energy_per_month = df[general_energy_cols].resample('M').sum() # for␣
 ↪energy we use sum to calculate overall consumption in period

plt.figure(figsize=(20,10))

sns.lineplot(data=general_energy_per_month, dashes=False)
```

[16]: <AxesSubplot:>



[17]:
```
rooms_energy_cols = ['Home_office', 'Wine_cellar','Garage_door',
                     'kitchen', 'Barn', 'Well','Living_room']

rooms_energy_per_month = df[rooms_energy_cols].resample('M').mean()

plt.figure(figsize=(20,8))

sns.lineplot(data=rooms_energy_per_month, dashes=False)
```

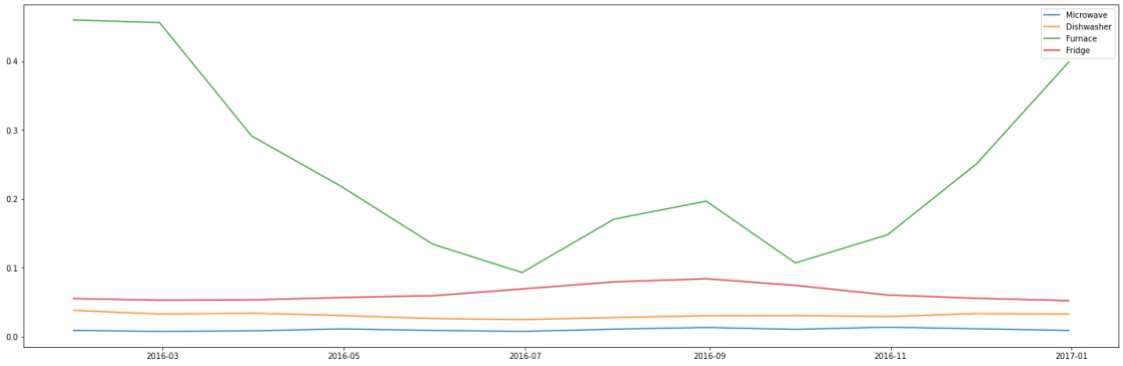
[17]: <AxesSubplot:>




- The energy consumption of kitchen, garage and the well remained almost the same throughout the year
- There's seasonality of energy consumption in other parts of the house :
  - A clear spike in september in the energy consumed by the wine cellar and the home office
  - A clear downtrend in the summer for the barn energy consumption

[18]:
```
equipements_cols = ['Microwave', 'Dishwasher', 'Furnace', 'Fridge']

equipements_energy_per_month = df[equipements_cols].resample('M').mean()

plt.figure(figsize=(25,8))

sns.lineplot(data= equipements_energy_per_month, dashes=False)
```

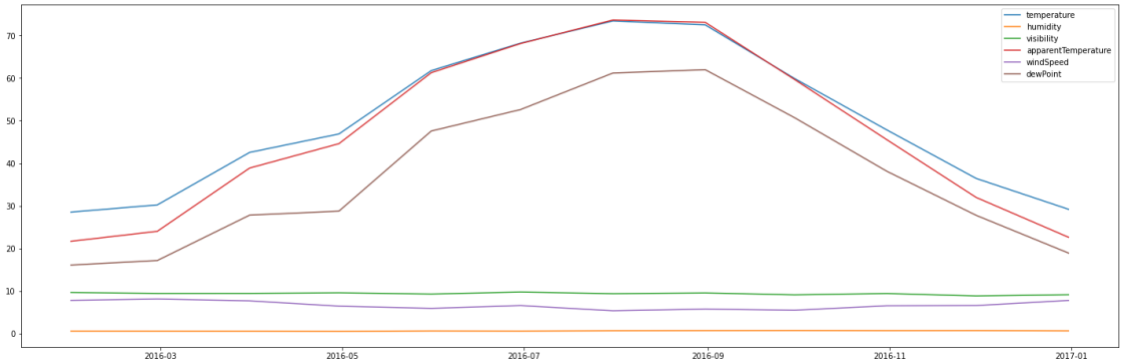
[18]: <AxesSubplot:>



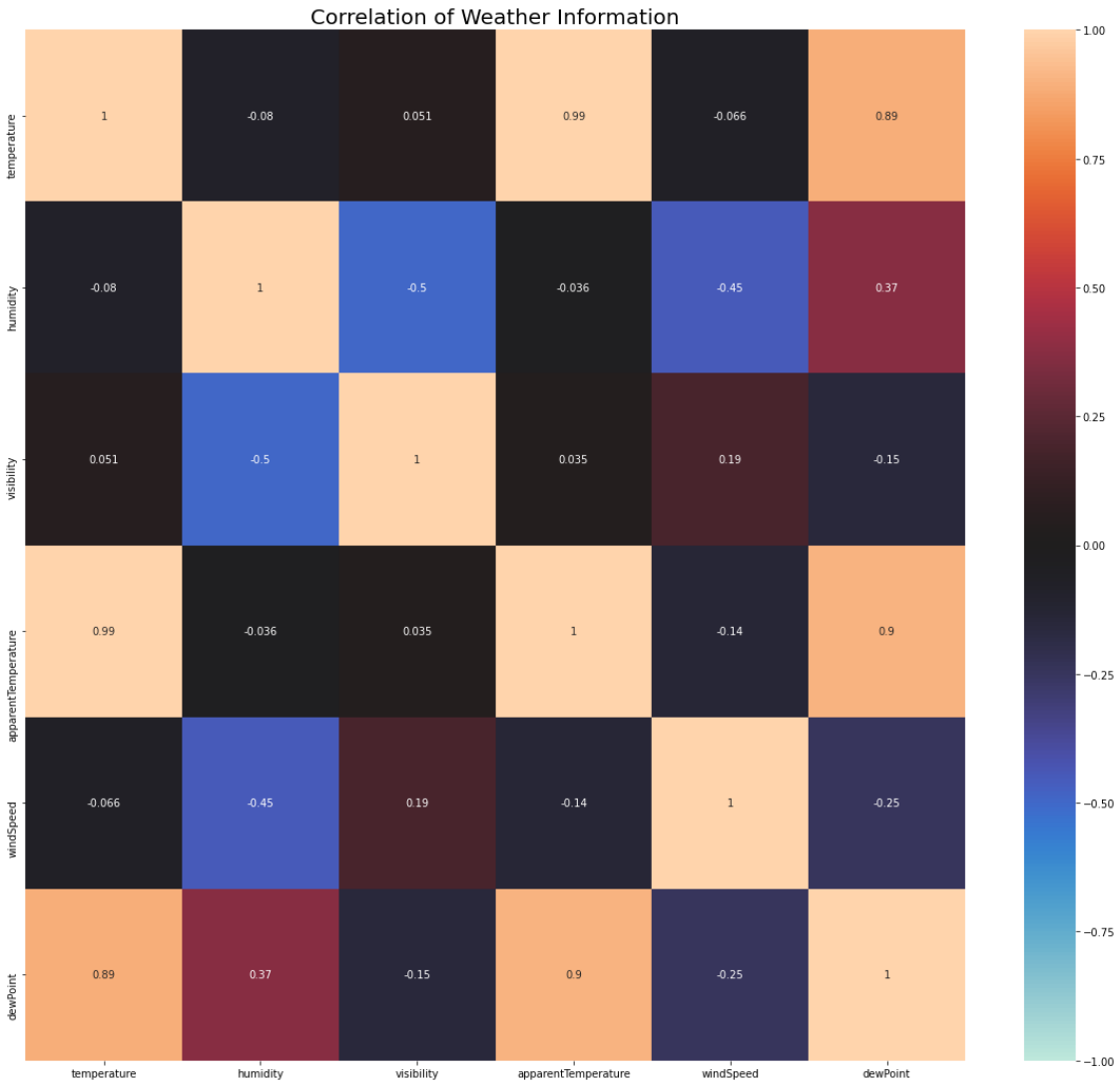

The usage of the furnace decreases in the summer

[19]:
```
weather_columns = ['temperature','humidity', 'visibility',␣
 ↪'apparentTemperature',
                   'windSpeed', 'dewPoint']

weather_per_month = df[weather_columns].resample('M').mean()

plt.figure(figsize=(25,8))

sns.lineplot(data=weather_per_month, dashes=False)
```

[19]: <AxesSubplot:>




[20]:
```
fig,ax = plt.subplots(figsize=(20,
18)) corr =
df[weather_columns].corr()
sns.heatmap(corr, annot=True, vmin=-1.0, vmax=1.0,
center=0) ax.set_title('Correlation of Weather
Information', size=20) plt.show()
```


Correlation of Weather Information
temperature
humidity
visibility
apparentTemperature
windSpeed
dewPoint
1
-0.08
0.051
0.99
-0.066
0.89
-0.08
1
-0.5
-0.036
-0.45
0.37
0.051
-0.5
1
0.035
0.19
-0.15
0.99
-0.036
0.035
1
-0.14
0.9
-0.066
-0.45
0.19
-0.14
1
-0.25
0.89
0.37
-0.15
0.9
-0.25
1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00