Combining datasources for further analysis

ANALYZING IOT DATA IN PYTHON

Matthias Voppichler
IT Developer

datacamp

Combining data sources

```
print(temp.head())
```

```
                     value
timestamp
2018-10-03 08:00:00   16.3
2018-10-03 09:00:00   17.7
2018-10-03 10:00:00   20.2
2018-10-03 11:00:00   20.9
2018-10-03 12:00:00   21.8
```

```
print(sun.head())
```

```
                      value
timestamp
2018-10-03 08:00:00  1798.7
2018-10-03 08:30:00  1799.9
2018-10-03 09:00:00  1798.1
2018-10-03 09:30:00  1797.7
2018-10-03 10:00:00  1798.0
```

Naming columns

```
temp.columns = ["temperature"]
sun.columns = ["sunshine"]

print(temp.head(2))
print(sun.head(2))
```

```
                     temperature
timestamp
2018-10-03 08:00:00         16.3
2018-10-03 09:00:00         17.7
                     sunshine
timestamp
2018-10-03 08:00:00    1798.7
2018-10-03 08:30:00    1799.9
```

Concat

```
environ = pd.concat([temp, sun], axis=1)
print(environ.head())
```

```
                     temperature  sunshine
timestamp
2018-10-03 08:00:00         16.3    1798.7
2018-10-03 08:30:00          NaN    1799.9
2018-10-03 09:00:00         17.7    1798.1
2018-10-03 09:30:00          NaN    1797.7
2018-10-03 10:00:00         20.2    1798.0
```

Resample

```
agg_dict = {"temperature": "max", "sunshine": "sum"}
env1h = environ.resample("1h").agg(agg_dict)
print(env1h.head())
```

```
                     temperature  sunshine
timestamp
2018-10-03 08:00:00         16.3    3598.6
2018-10-03 09:00:00         17.7    3595.8
2018-10-03 10:00:00         20.2    3596.2
2018-10-03 11:00:00         20.9    3594.1
2018-10-03 12:00:00         21.8    3599.9
```

Fillna

```
env30min = environ.fillna(method="ffill")
print(env30min.head())
```

```
                     temperature  sunshine
timestamp
2018-10-03 08:00:00         16.3    1798.7
2018-10-03 08:30:00         16.3    1799.9
2018-10-03 09:00:00         17.7    1798.1
2018-10-03 09:30:00         17.7    1797.7
2018-10-03 10:00:00         20.2    1798.0
```

Let's practice!

ANALYZING IOT DATA IN PYTHON

Correlation

ANALYZING IOT DATA IN PYTHON

Matthias Voppichler
IT Developer

datacamp

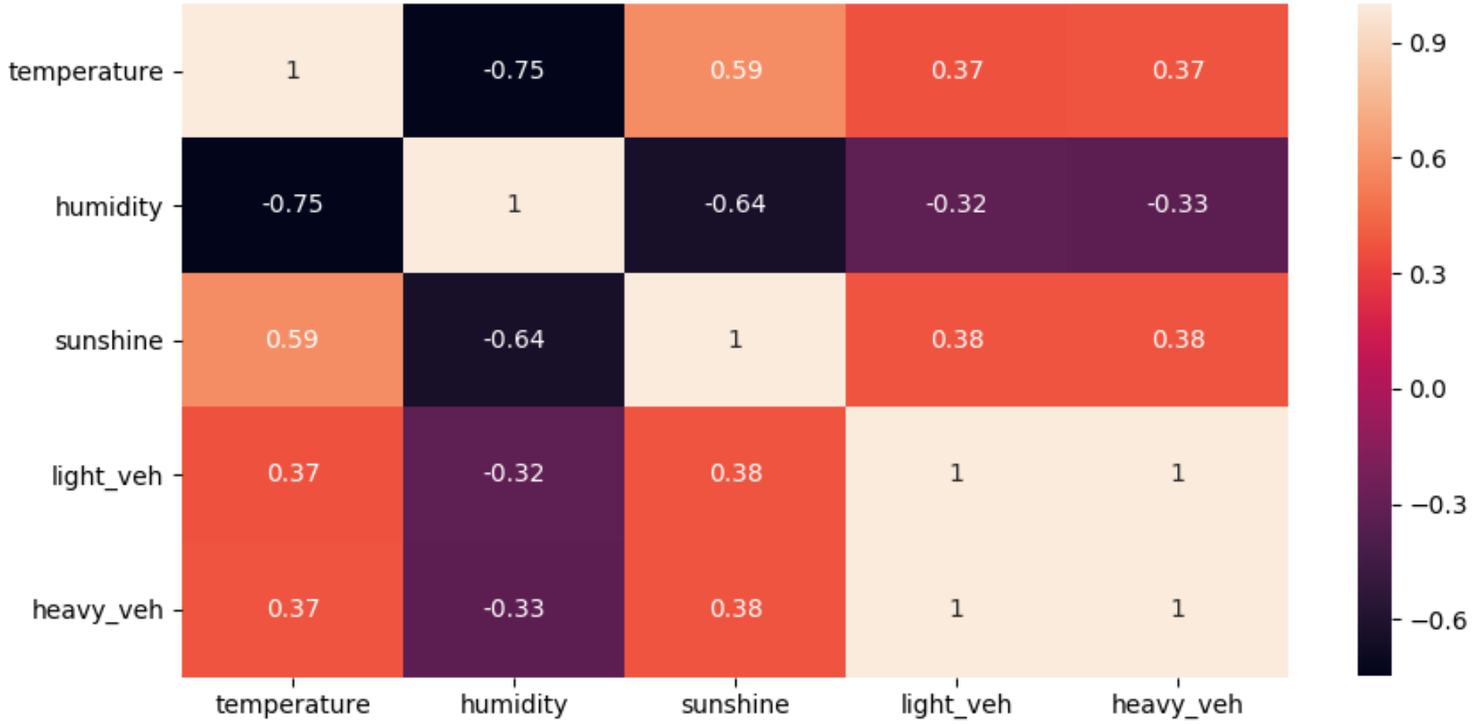
df.corr()

```
print(data.corr())
```

```
             temperature  humidity  sunshine  light_veh  heavy_veh
temperature     1.000000 -0.734430  0.611041   0.401997   0.408936
humidity       -0.734430  1.000000 -0.637761  -0.313952  -0.318198
sunshine        0.611041 -0.637761  1.000000   0.408854   0.409363
light_veh       0.401997 -0.313952  0.408854   1.000000   0.998473
heavy_veh       0.408936 -0.318198  0.409363   0.998473   1.000000
```

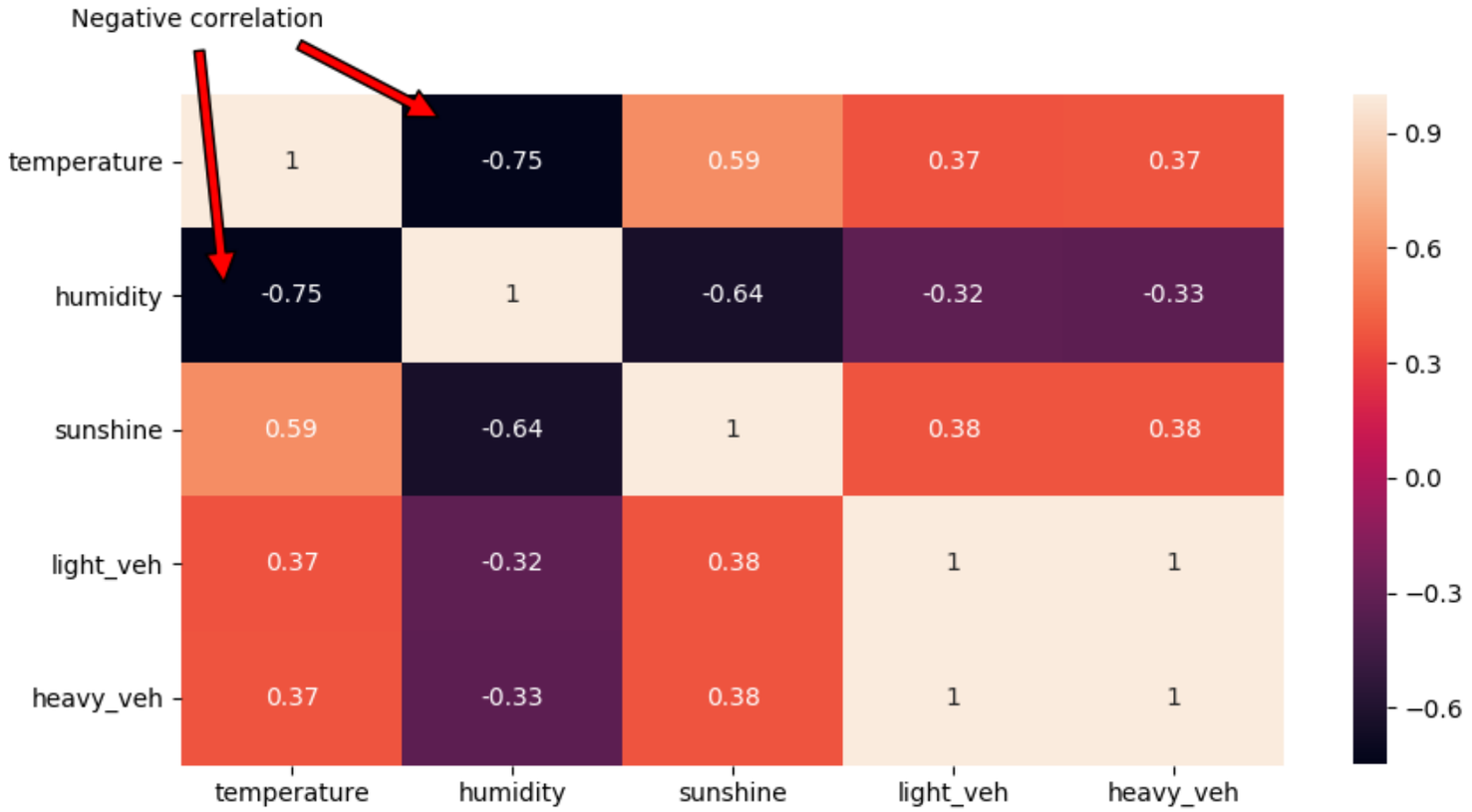
heatmap

```
sns.heatmap(data.corr(), annot=True)
```



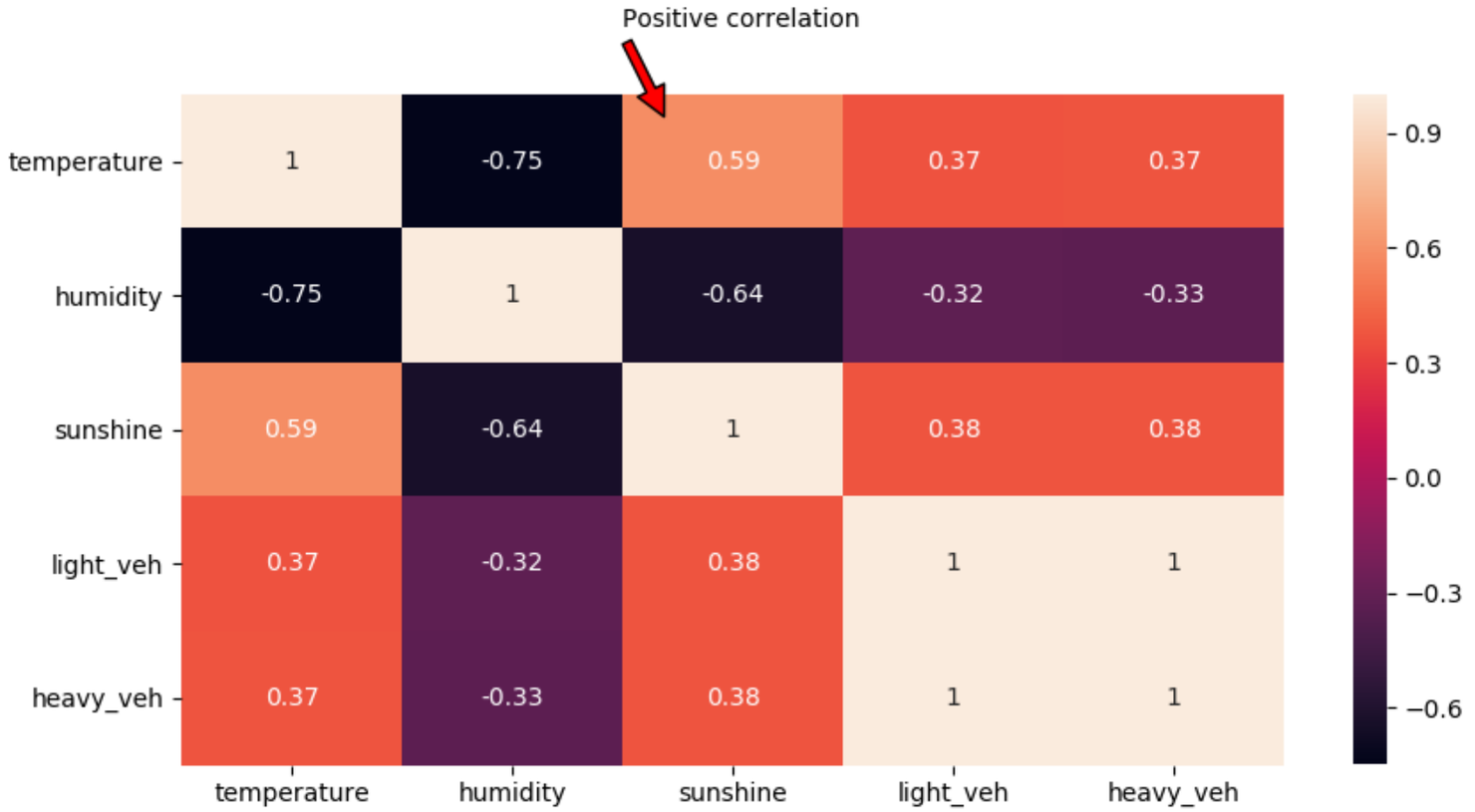
heatmap

```
sns.heatmap(data.corr(), annot=True)
```



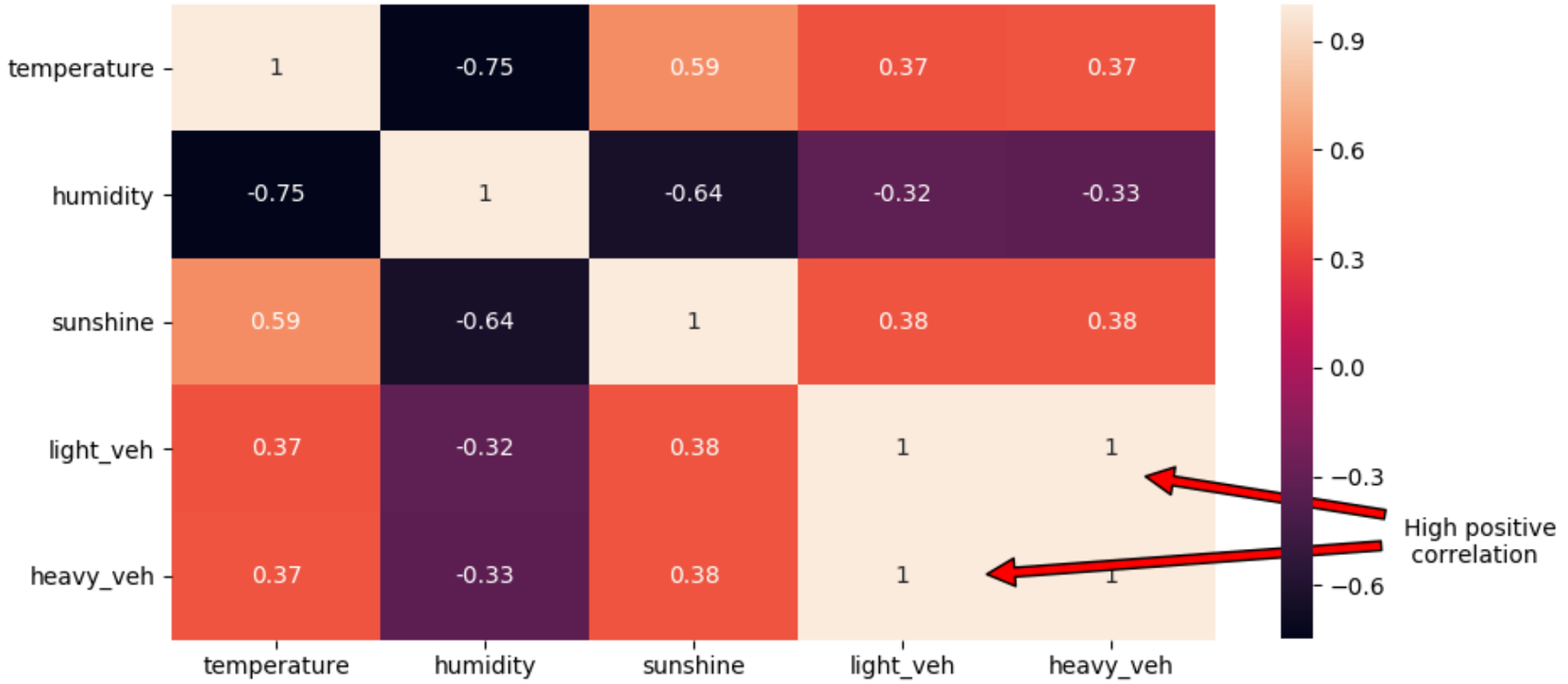
heatmap

```
sns.heatmap(data.corr(), annot=True)
```



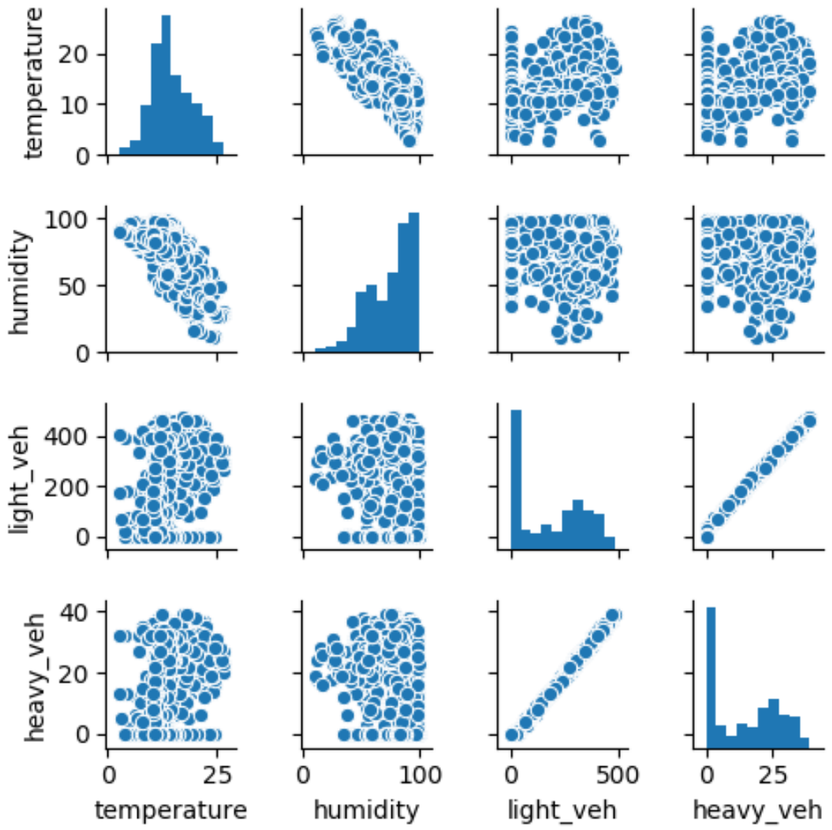
heatmap

```
sns.heatmap(data.corr(), annot=True)
```



Pairplot

```
sns.pairplot(data)
```



Summary

- heatmap
  - Negative correlation
  - Positive correlation
  - Correlation close to 1

Let's practice!

ANALYZING IOT DATA IN PYTHON

Outliers

ANALYZING IOT DATA IN PYTHON

Matthias Voppichler
IT Developer

datacamp

Outliers

Reasons why outliers appear in Datasets:

- Measurement error
- Manipulation
- Extreme Events

Outliers

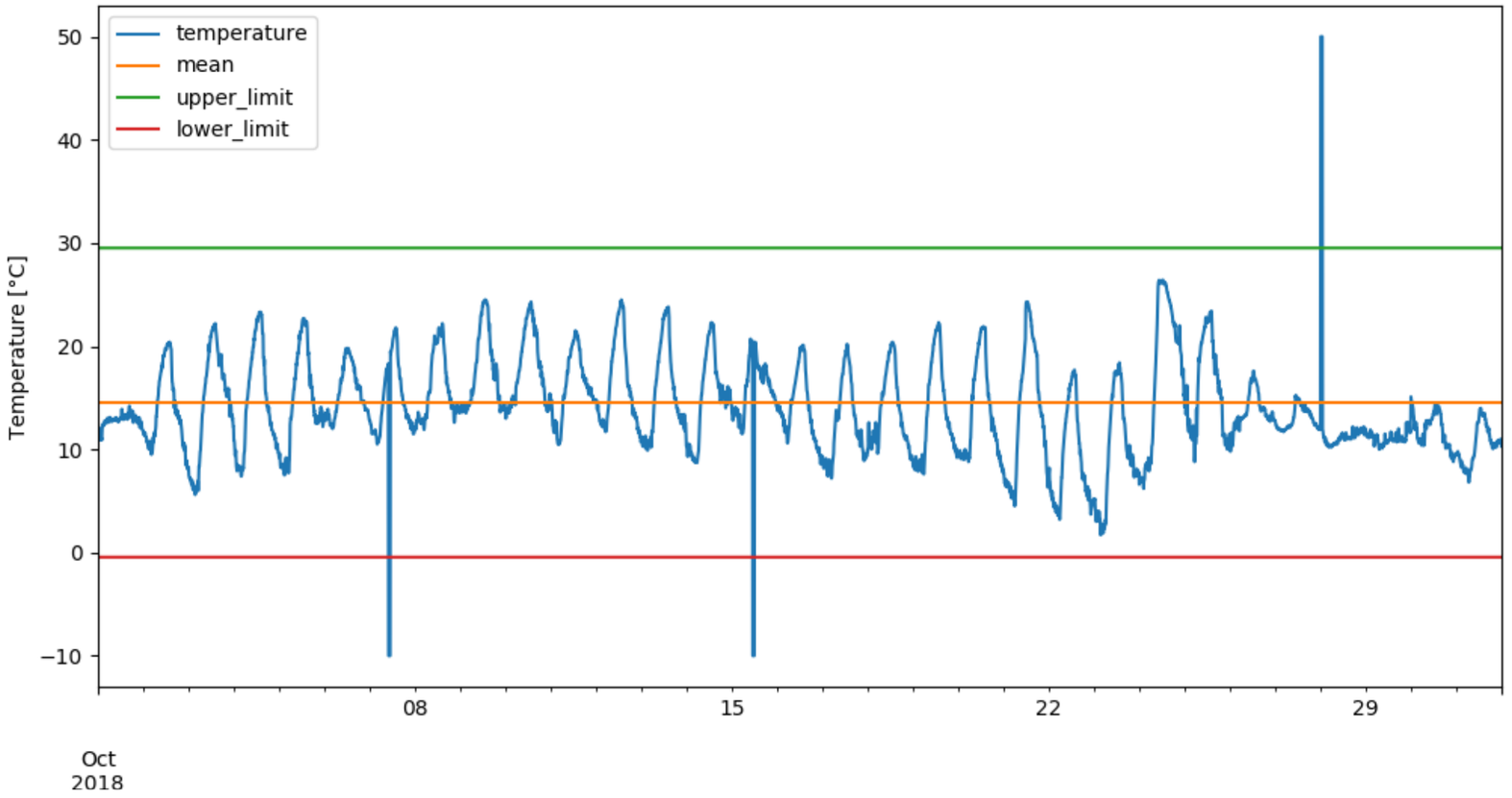
```
temp_mean = data["temperature"].mean()
temp_std = data["temperature"].std()

data["mean"] = temp_mean
data["upper_limit"] = temp_mean + (temp_std * 3)
data["upper_limit"] = temp_mean - (temp_std * 3)
print(data.iloc[0]["upper_limit"])
print(data.iloc[0]["mean"])
print(data.iloc[0]["lower_limit"])
```

```
29.513933116002725
14.5345
-0.444933116002724 56
```

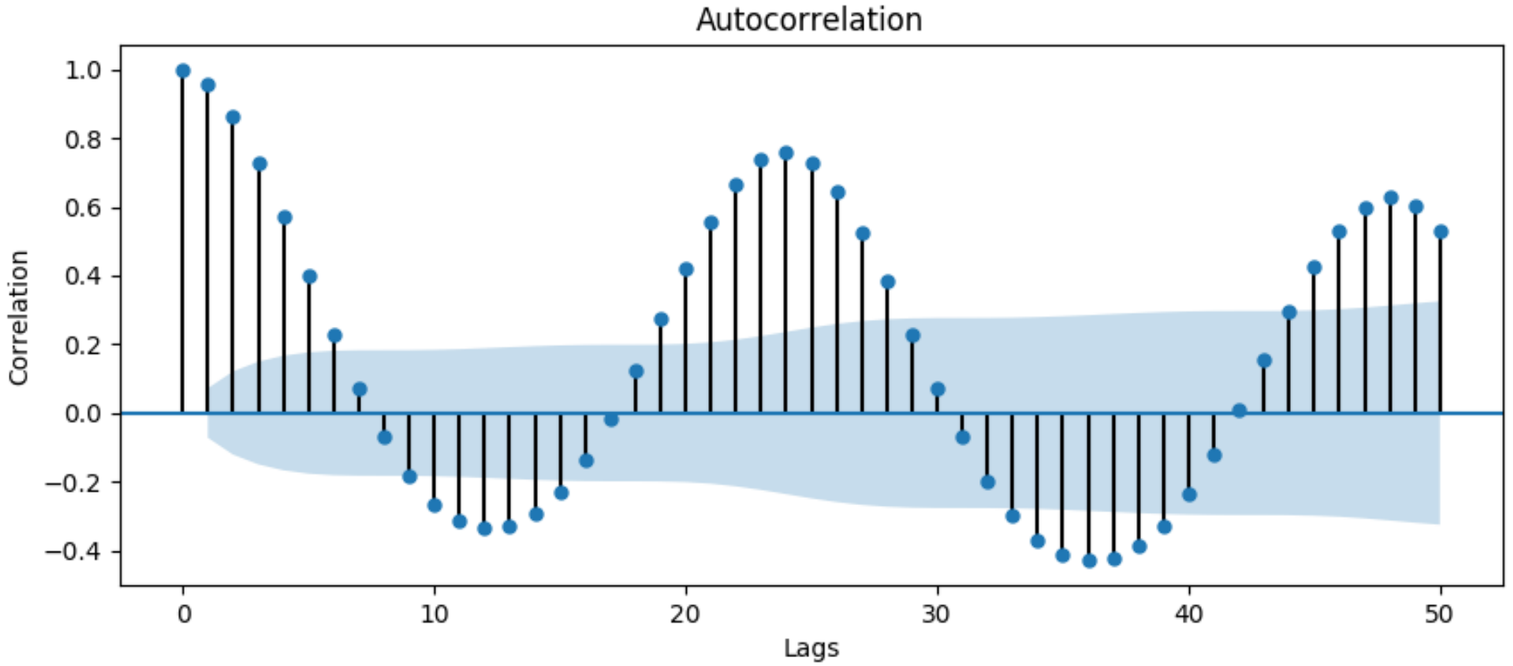
Outlier plot

```
data.plot()
```



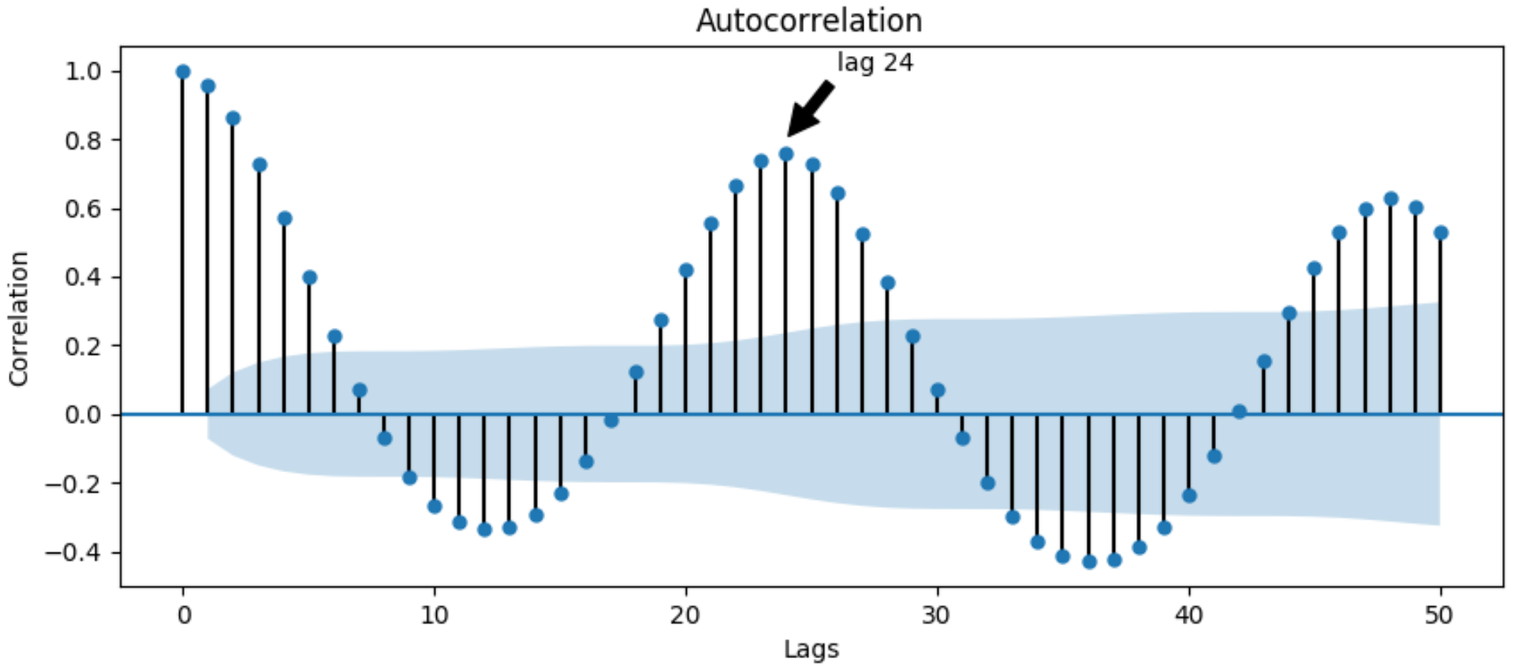
Autocorrelation

```
from statsmodels.graphics import tsaplots
tsaplots.plot_acf(data['temperature'], lags=50)
```



Autocorrelation

```
from statsmodels.graphics import tsaplots
tsaplots.plot_acf(data['temperature'], lags=50)
```



Let's practice!

ANALYZING IOT DATA IN PYTHON

Seasonality and Trends

ANALYZING IOT DATA IN PYTHON

Matthias Voppichler
IT Developer

datacamp

Time series components

- Trend
- Seasonal
- Residual / Noise

```
series[t] = trend[t] + seasonal[t] + residual[t]
```

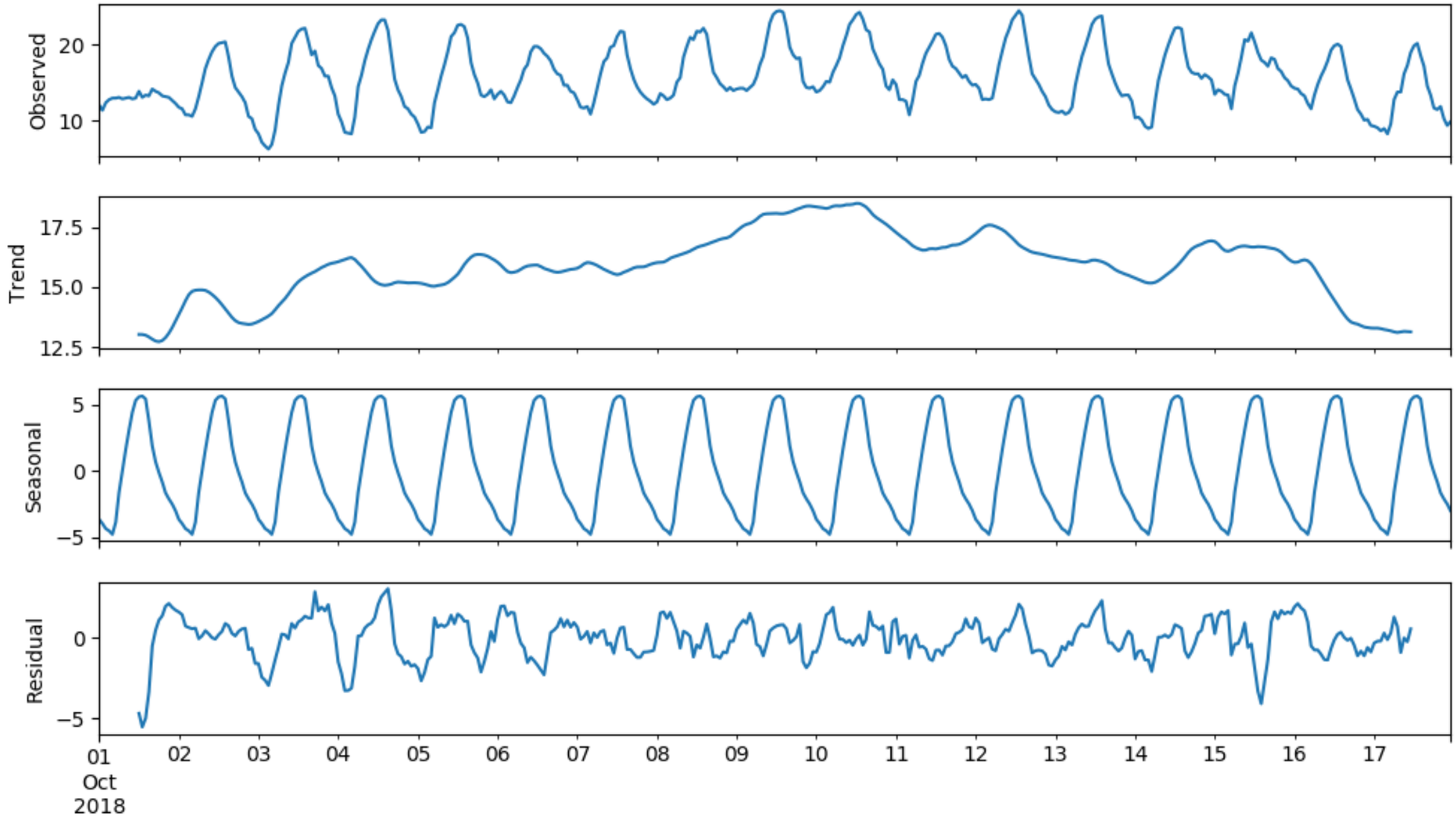
```
20.2 = 14.9 + 4.39 + 0.91
```

Seasonal decompose

```
import statsmodels.api as sm
# Run seasonal decompose
decomp = sm.tsa.seasonal_decompose(data["temperature"])
print(decomp.seasonal.head())
decomp.plot()
```

```
timestamp
2018-10-01 00:00:00   -3.670394
2018-10-01 01:00:00   -3.987451
2018-10-01 02:00:00   -4.372217
2018-10-01 03:00:00   -4.534066
2018-10-01 04:00:00   -4.802165
Freq: H, Name: temperature, dtype: float64
```

Seasonal decompose

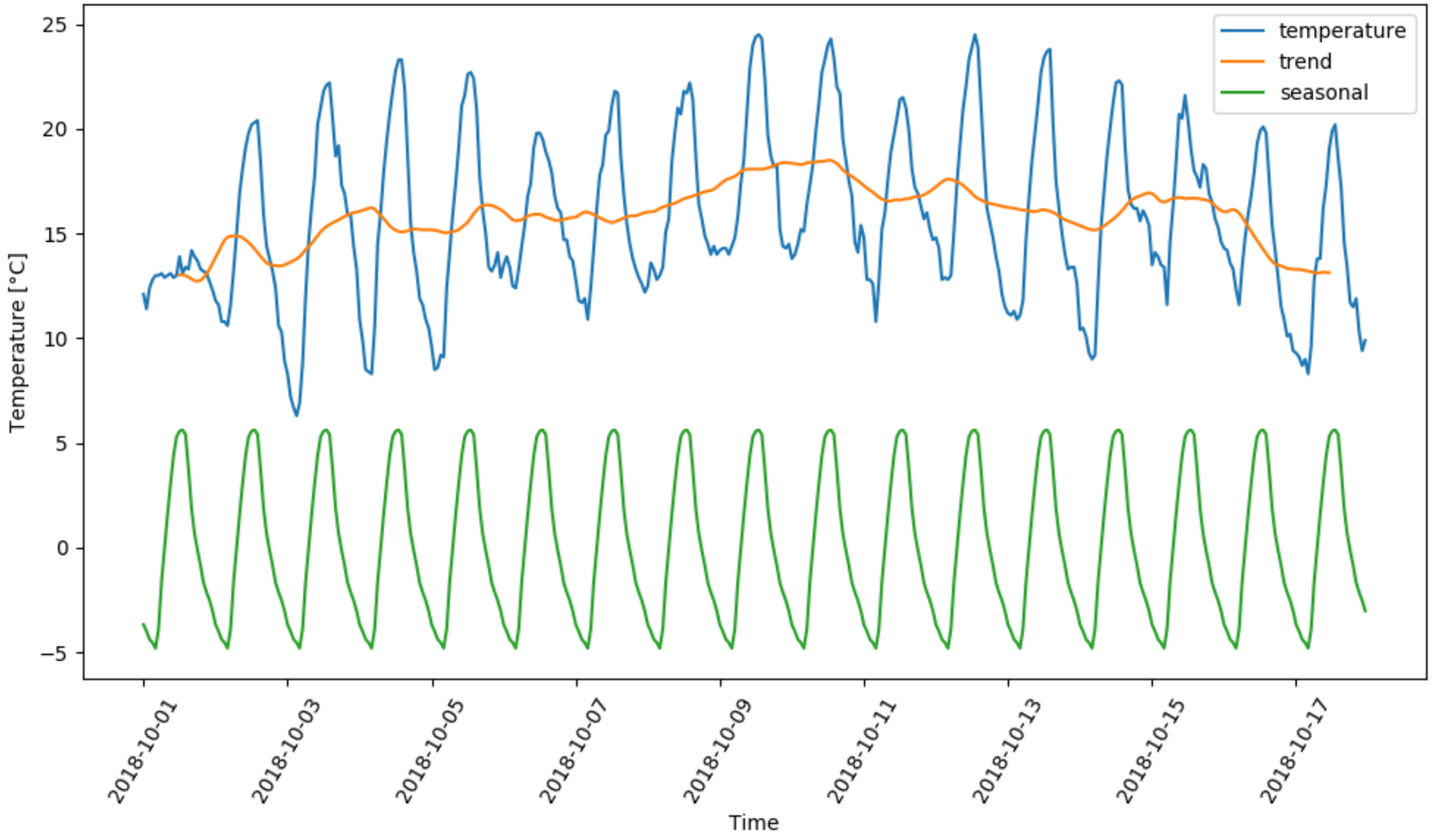


Combined plot

```
# Plot the timeseries
plt.plot(data["temperature"], label="temperature")
decomp = sm.tsa.seasonal_decompose(data["temperature"])
# Plot trend and seasonality
plt.plot(decomp.trend, label="trend")
plt.plot(decomp.seasonal, label="seasonal")


plt.show()
```

Combined plot



Let's practice!

ANALYZING IOT DATA IN PYTHON