# Polynomial regression

The example belows uses a temperature-energy dataset in order to illustrate how to perform a non linear regression.

Workflow:

1. Preparation
   - Extract the dataset from the _power_demand_vs*temperature.csv*
   - Explore the dataset and check for missing values
   - Plot the distribution
   - Divide the dataset into train and test
   - Create an evaluation function
2. First experiment
   - Create a linear model
   - Train the model on X_train and y_train
   - Evaluate the model on X_test and y_test
   - Visualize the prediction of the model
3. Second experiment
   - Create a polynomial regression model with degree 2
   - Train the model on X_train and y_train
   - Evaluate the model on X_test and y_test
   - Visualize the prediction of the model
4. Third experiment
   - repeat the steps done in the second experiment but with degree 3
5. Third experiment
   - repeat the steps done in the second experiment but with degree 4
6. Compare the evaluation of each model

In [ ]:
```
# Code source: Filippo Orazi
# License: BSD 3 clause
```

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

random_state = 42 # this will be used to guarantee the repeatability of the experiment
```

# Dataset preparation

## Load the dataset from a `.csv` file

This cell allows full compatibility between execution in Google Colab and in local

In [ ]:

```
try:
  import google.colab.files
  IN_COLAB = True
except:
  IN_COLAB = False
# from google.colab import files
if IN_COLAB:
    uploaded = files.upload()
```

The file must be available in the same directory, or uploaded in the Colab environment in the execution of the previous cell

Set the date column as index

In [ ]:

```
```

Out[ ]:

| | demand | temp |
|---|---|---|
| **date** | | |
| **2015-01-01** | 1.736065 | 1.7 |
| **2015-01-02** | 1.831672 | 2.2 |
| **2015-01-03** | 1.714934 | 14.4 |
| **2015-01-04** | 1.628577 | 15.6 |
| **2015-01-05** | 2.045394 | 0.0 |

## Explore the dataset and check for missing values

In [ ]:

Out[ ]:

| | demand | temp |
|---|---|---|
| **count** | 1096.000000 | 1096.000000 |
| **mean** | 1.831796 | 16.927737 |
| **std** | 0.329434 | 10.791581 |
| **min** | 1.316033 | -15.000000 |
| **25%** | 1.581654 | 8.900000 |
| **50%** | 1.731479 | 18.900000 |
| **75%** | 2.024869 | 26.100000 |
| **max** | 2.804025 | 33.900000 |

In [ ]:

```
The dataframe has 0 invalid rows
```

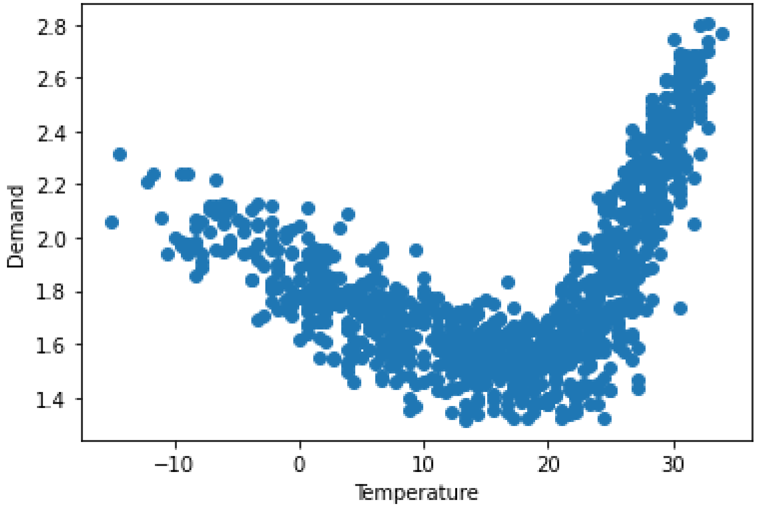
## Create X and y

In [ ]:

```
X has shape(1096, 1)
Y has shape(1096,)
```

## Plot the distribution

In [ ]:




## Divide the dataset in train and test splits

In [ ]:

```
Training set and test set have 767 and 329 elements respectively
```

## Create an evaluation function to compute, print and return the metrics: rmse r2 f-statistic and p-value

In [ ]:

```
# Computation of F-statistic and p-value for the regression
# http://facweb.cs.depaul.edu/sjost/csc423/documents/f-test-reg.htm
```

# First experiment

Create a linear model

Train the model on X_train and y_train

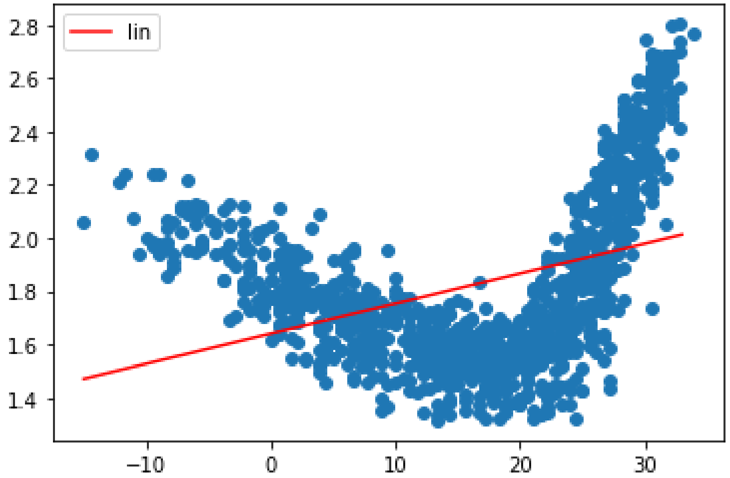
Evaluate the model on X_test and y_test

In [ ]:

```
Mean squared error:     0.10016
r2 score:               0.1803
f-statistic:            53.273
p-value:                2.2197e-12
```

### Visualize the prediction of the model

In [ ]:




# Second experiment - Polynomial regression

We can clearly see that the linear regression model cannot really approximate the data distribution.

We can now try with a non linear regression model:

1. Use the sklearn fucntion *PolynomialFeature* to create a new array of features. Set *degree=2* and _include*bias=False*

2. Train a Linear regression model with the new features
3. Evaluate the model
4. Visualize the predicted values of the model

### Polynomial degree = 2

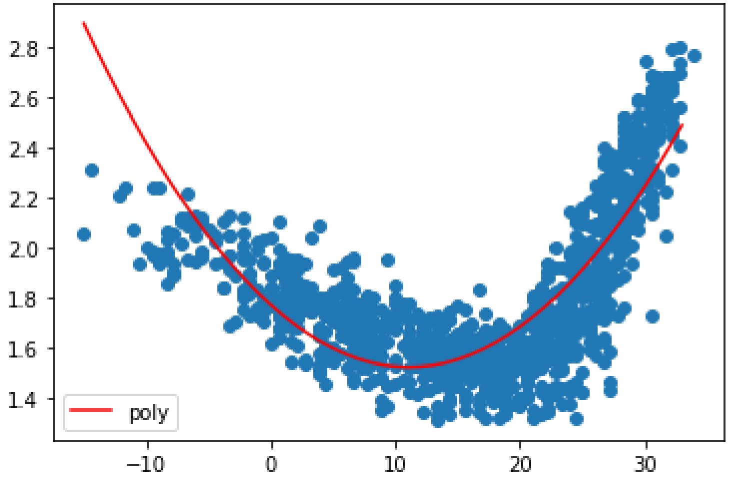
In [ ]:

Out[ ]:
```
LinearRegression()
```

In [ ]:

```
Mean squared error:     0.033456
r2 score:               0.72619
f-statistic:            384.89
p-value:                1.1102e-16
```

In [ ]:




## Third experiment

### Polynomial degree = 3

In [ ]:

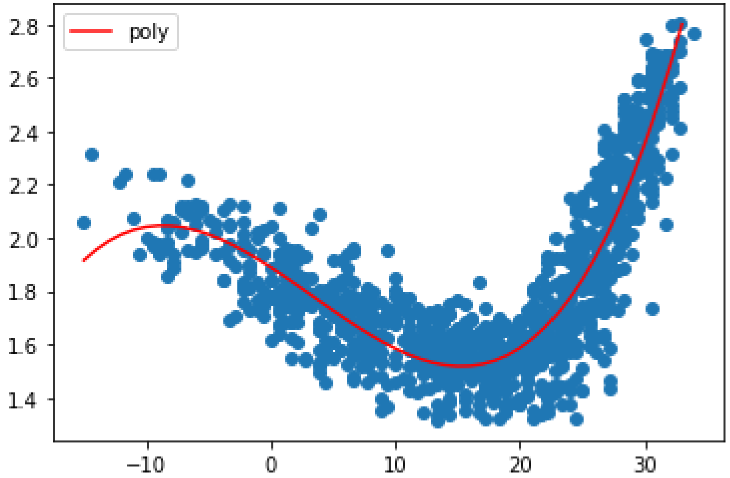
```
Polynomial degree = 3
LinearRegression()
```

Out[ ]:

In [ ]:

```
Mean squared error:     0.021749
r2 score:               0.822
f-statistic:            502.32
p-value:                1.1102e-16
```

In [ ]:




# Fourth experiment

## Polynomial degree = 4

In [ ]:

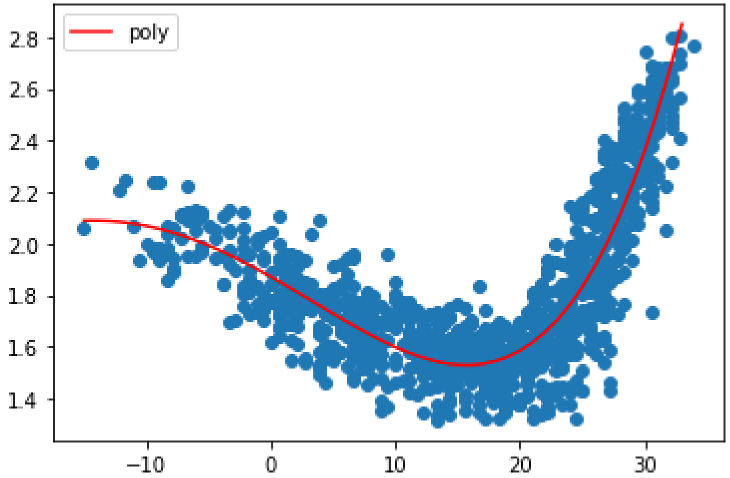
```
Polynomial degree = 4
LinearRegression()
```

Out[ ]:

In [ ]:

```
Mean squared error:     0.021334
r2 score:               0.8254
f-statistic:            390.05
p-value:                1.1102e-16
```

In [ ]:




# Compare the performance of the four models

In [ ]:

Out[ ]:

| | linear | polynomial d = 2 | polynomial d = 3 | polynomial d = 4 |
|---|---|---|---|---|
| **rmse** | 1.001591e-01 | 3.345625e-02 | 2.174942e-02 | 2.133387e-02 |
| **r2** | -5.366169e+00 | 5.756325e-01 | 7.843318e-01 | 7.923317e-01 |
| **f-statistic** | 5.327309e+01 | 3.848865e+02 | 5.023183e+02 | 3.900454e+02 |
| **p-value** | 2.219669e-12 | 1.110223e-16 | 1.110223e-16 | 1.110223e-16 |

In [ ]: