M1 Data Science for Business

2019/2020

Project n°1 :

Predict the GDP amount for a given year

Elaborated by :

SOULAIMEN Mohamed

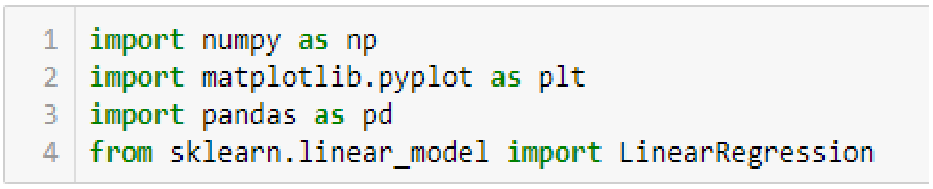
Practical machine learning

soulaimenmohamed@gmail.com

1. Data understanding :

Quality data is fundamental to any data science engagement. To gain actionable insights, the appropriate data must be sourced and cleansed. There are two key stages of Data Understanding : a Data Assessment and Data Exploration.

But before that, we must make sure that our environment is adapted to our project. For this we need to install the following libraries:

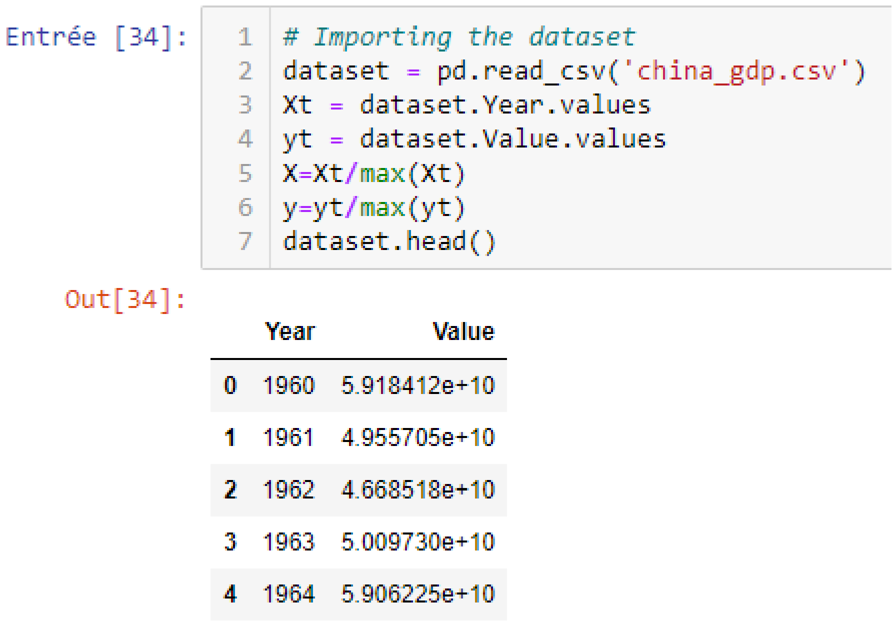


```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression
```

- Data Assessment

The first step in data understanding is a Data Assessment. This should be undertaken before the kick-off of a project as it is an important step to validate its feasibility. This task evaluates what data is available and how it aligns to the business problem.

For that we import our dataset and print the head to make sure that the data was loaded successfuly



```
Entrée [34]:
# Importing the dataset
dataset = pd.read_csv('china_gdp.csv')
Xt = dataset.Year.values
yt = dataset.Value.values
X=Xt/max(Xt)
y=yt/max(yt)
dataset.head()
```

Out[34]:

	Year	Value
0	1960	5.918412e+10
1	1961	4.955705e+10
2	1962	4.668518e+10
3	1963	5.009730e+10
4	1964	5.906225e+10

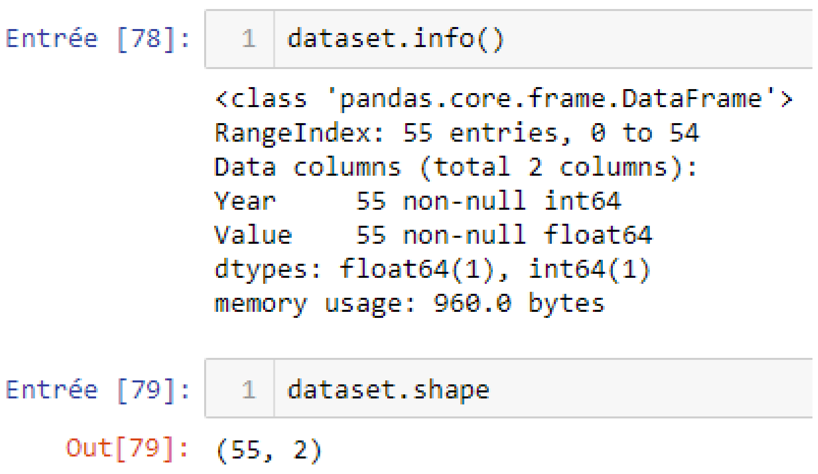
- dataset: the table contains all values in our csv file
- X: the column which contains Year (normalized data)
- y: the column which contains Value (normalized data)

The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges(like year and value).

- Data Exploration

Once you have access to data, you can start Data Exploration. This is a phase for creating meaningful summaries of your data and is particularly important if you are unfamiliar with the data.

To do so we had to write the following code :


```
Entrée [78]: dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55 entries, 0 to 54
Data columns (total 2 columns):
Year      55 non-null int64
Value     55 non-null float64
dtypes: float64(1), int64(1)
memory usage: 960.0 bytes
```

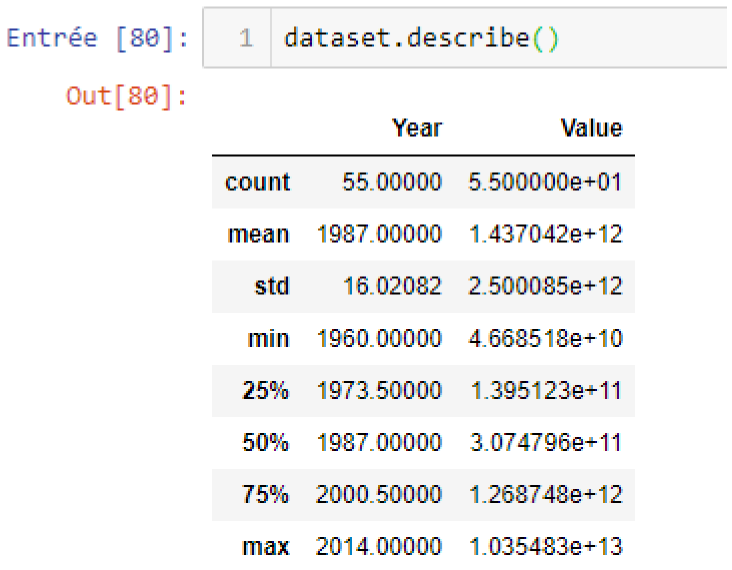
```
Entrée [79]: dataset.shape
```

```
Out[79]: (55, 2)
```

The output of this code show us that our dataset :

- is composed from 2 columns and 55 line
- column 'Year' with datatype int and there is no missing values in this column
- column 'Value with datatype float and there is no missing values in this column

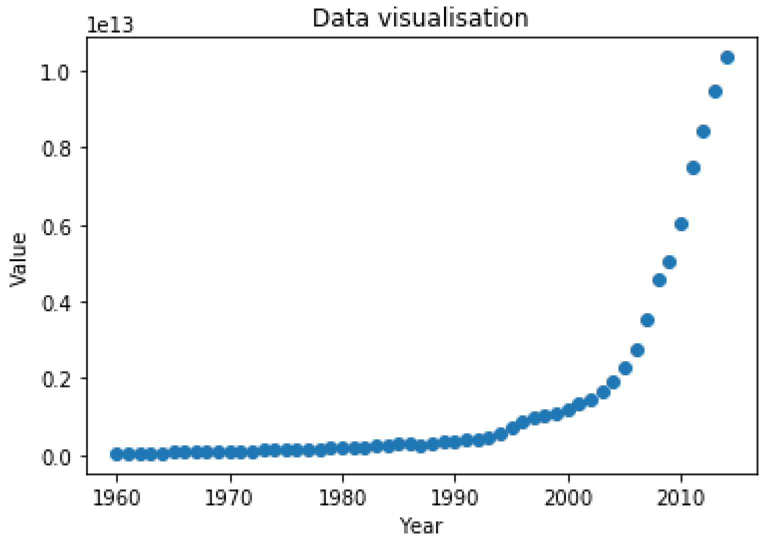
After we made sure that our data is clean we can have a look of the summary of the dataset so we can have a better idea about our data like the min value , the max value ,the average, the percintile etc…


```
Entrée [80]: dataset.describe()
```

```
Out[80]:
             Year         Value
count    55.00000  5.500000e+01
mean   1987.00000  1.437042e+12
std      16.02082  2.500085e+12
min    1960.00000  4.668518e+10
25%    1973.50000  1.395123e+11
50%    1987.00000  3.074796e+11
75%    2000.50000  1.268748e+12
max    2014.00000  1.035483e+13
```

We can now plot our data so we can dive into our analysis and try so visualise some hidden patterns from our data and this is the appropriate code for it :

```
Entrée [123]:
plt.scatter(dataset.Year,dataset.Value)
plt.xlabel("Year")
plt.ylabel("Value")
plt.title("Data visualisation")
plt.show()
```



From the plot we can say that the relation between the dependant and independent variable is non linear. We can also add that our data is well distrubuted in the factoriel plan.

2. Modeling :

In this phase, we are going to chose our machine learning model.

Let’s split our dataset to get training set and testing set
(both X and y values per each set)

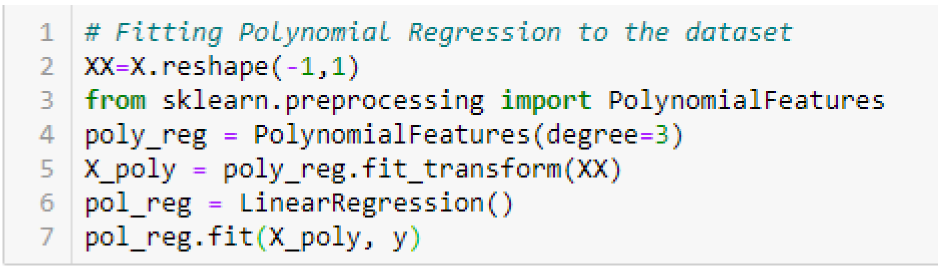
```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

- test_size=0.2 : we will split our dataset (10 observations) into 2 parts (training set, test set) and the ratio of **test set** compare to dataset is 0.2 (2 observations will be put into the **test set**. You can put it 1/5 to get 20% or 0.2, they are the same. We should not let the test set too big; if it's too big, we will lack of data to train.
- train_size: if we use the test_size already, the rest of data will automatically be assigned to train_size.
- random_state: this is the seed for the random number generator.

We already have the train set and test set, now we have to build the Model.
For the model we are going to use a polynomial regression.

- Polynomial provides the best approximation of the relationship between the dependent and independent variable.
- Polynomial regression fits a nonlinear model to the data
- A Broad range of function can be fit under it.
- Polynomial basically fits a wide range of curvature.

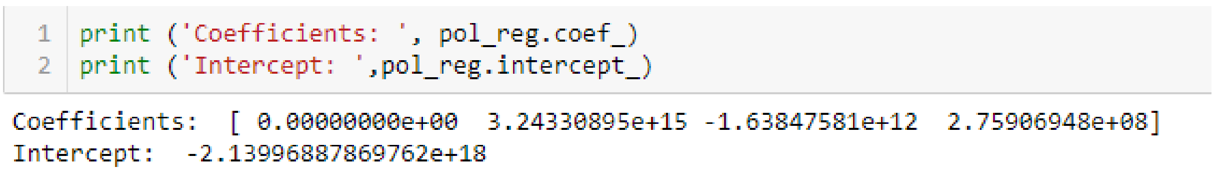
Now, we can deal with it as 'linear regression' problem. Therefore, this polynomial regression is considered to be a special case of traditional multiple linear regression. So, you can use the same mechanism as linear regression to solve such a problems



```
# Fitting Polynomial Regression to the dataset
XX=X.reshape(-1,1)
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=3)
X_poly = poly_reg.fit_transform(XX)
pol_reg = LinearRegression()
pol_reg.fit(X_poly, y)
```

- poly_reg : Generate polynomial and interaction features.
- X_poly : Fit to data, then transform it
- Pol_reg : Performing linear regression with scikit learn

Then we print the Coefficients and the Intercept of our polynomial function to have an idea about it

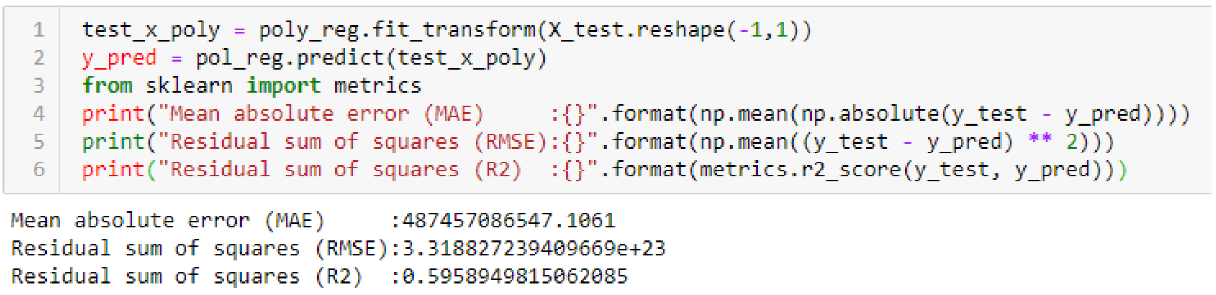


```python
print ('Coefficients: ', pol_reg.coef_)
print ('Intercept: ',pol_reg.intercept_)
```

```
Coefficients:  [ 0.00000000e+00  3.24330895e+15 -1.63847581e+12  2.75906948e+08]
Intercept:  -2.13996887869762e+18
```

3. Evaluation :

Validation and Evaluation of a machine learning Model provides more colour to our hypothesis and helps evaluate different models that would provide better results against our data. These are the metrics that help us evaluate our models



```python
test_x_poly = poly_reg.fit_transform(X_test.reshape(-1,1))
y_pred = pol_reg.predict(test_x_poly)
from sklearn import metrics
print("Mean absolute error (MAE)     :{}".format(np.mean(np.absolute(y_test - y_pred))))
print("Residual sum of squares (RMSE):{}".format(np.mean((y_test - y_pred) ** 2)))
print("Residual sum of squares (R2)  :{}".format(metrics.r2_score(y_test, y_pred)))
```

```
Mean absolute error (MAE)     :487457086547.1061
Residual sum of squares (RMSE):3.318827239409669e+23
Residual sum of squares (R2)  :0.5958949815062085
```

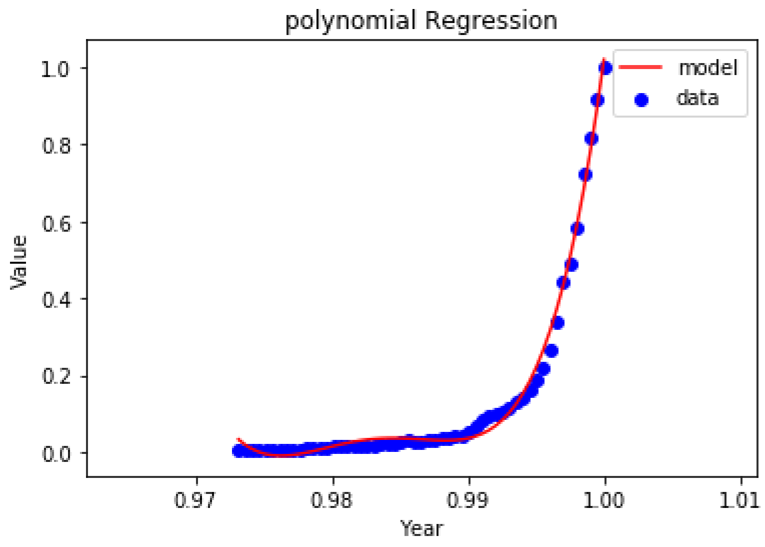
There are three main errors (metrics) used to evaluate models**, Mean absolute error (MAE)**, **Residual sum of squares (RMSE)** and **R2 score**.

- **MAE** : is a measure of difference between two continuous variables
- **RMSE** : used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed.
- **R2 :** regression score function Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse)

4. Plotting the model :

Now we are going to plot our model with our data to see our fit :

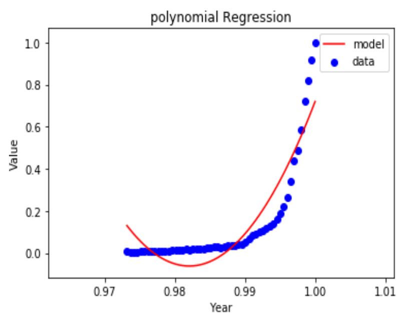
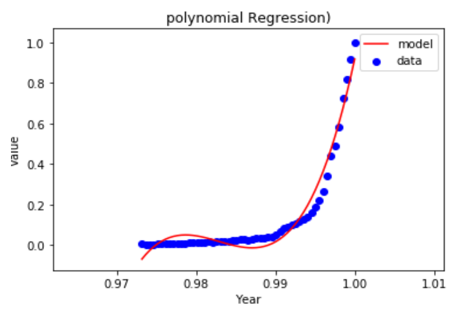
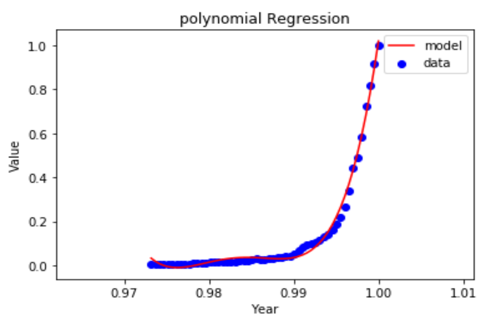
```python
# Visualizing the Polymonial Regression results
plt.scatter(X, y, color='blue',label="data")
plt.plot(XX, pol_reg.predict(X_poly), color='red',label="model")
plt.title('polynomial Regression')
plt.xlabel('Year')
plt.ylabel('Value')
plt.legend(loc='best')
plt.show()
```



From the plot we can say that our generated model pass throught the majority of our dataset

5. Comparing the models :

Finally we are going to compare our different models :

degree	2	3	4
MAE	0.078	0.047	0.016
RMSE	0.009	0.003	0.0005
R2	-0.188	0.595	0.932
Plot			

From this table we can assume that the polynomial model with the degree 4 is the best one because :

MAE(4) < MAE(3) < MAE(2)

RMSE(4) < RMSE(3) < RMSE(2)

R2(1) <R2(2) < R2(3)

→Polynomial (degree=4) :

Coefficients:
[0.
-58406559.52104107
89122479.16439816
-60440021.58002876
15370482.12011258]
Intercept: 14353620.835307961