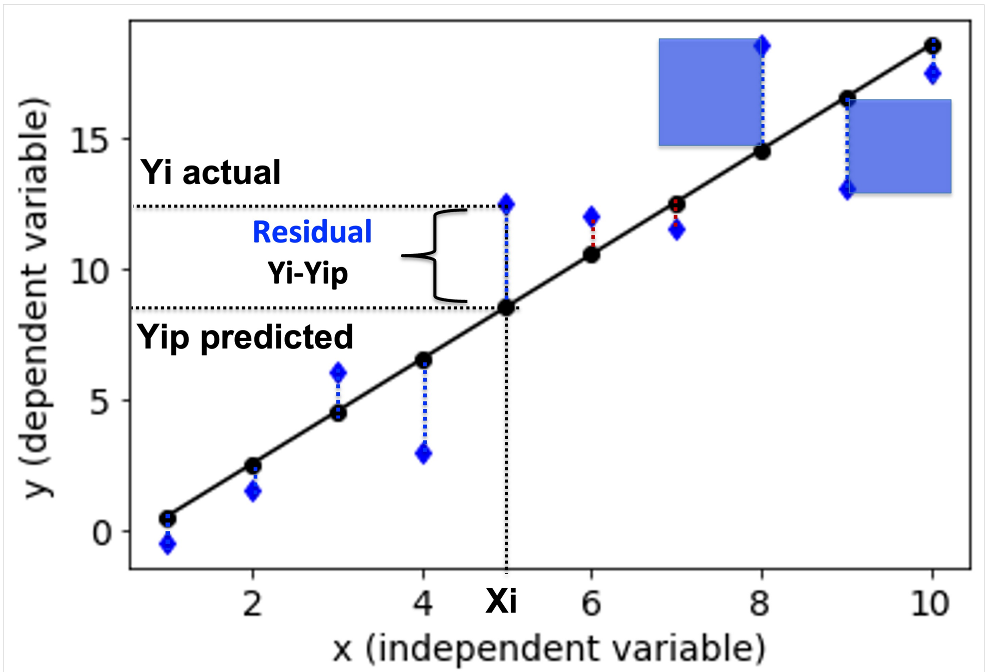
Least-Squares method

The Least Square method is the process of finding a regression line or best-fitted line for the observed data by minimizing **the sum of the squares of the residuals** between the data the fitting model



Sum of squared residuals (SSR) is basically the sum the blue squares in the above figure. Mathematically it is described as:

$$SSR = \sum_i (y_i - y_{ip})^2$$

Linear models - Coefficient of determination (R^2)

The coefficient of determination **R^2** is popularly used to determine the quality of the fit.
It is defined as:

$$R^2 = 1 - \frac{\sum_i \left(y_i - y_{ip}\right)^2}{\sum_i (y_i - \bar{y})^2}$$

Where:
y_i – the ith y data
y_{ip} – the ith predicted y value from the model
$\bar{y}$ – the average of the y data

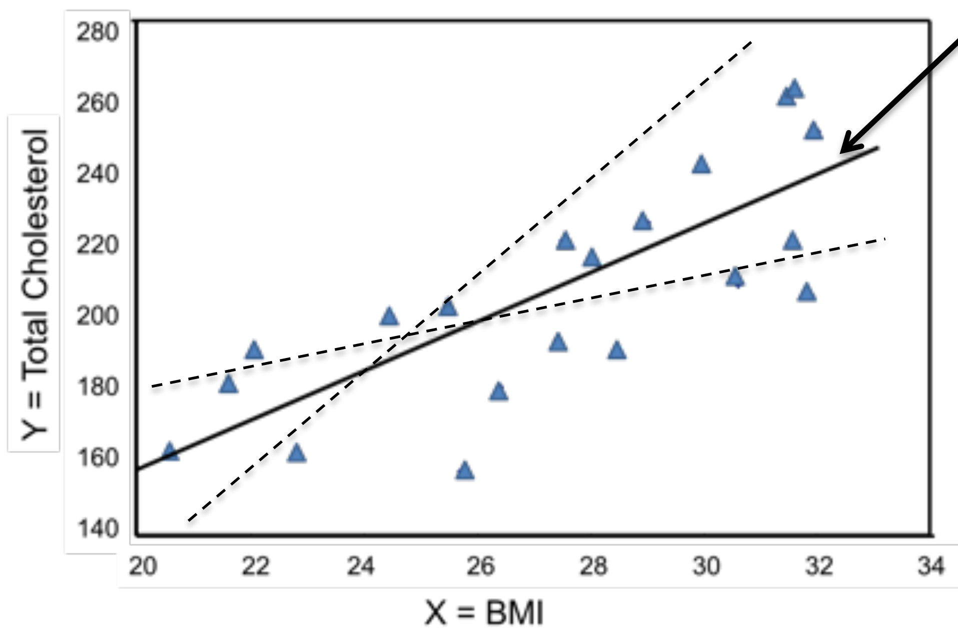
If R^2 close to 1 the model is a good approximation to the data compared to the mean of the data, but be careful not to overfit data

If R^2 close to 0 the model is poor, as the two deviations will be comparable

Example

Below is an example of a linear regression analysis, showing the dependence of the total cholesterol (dependent variable) on the Y-axis vs body mass index (independent variable) on the X-axis:

Model: Linear equation:
y = a+b x (a and b are the fitting parameters)



Fitted Model
y = 28.07 +6.49 x

R^2=0.67

Fitted parameters
a=28.07
b=6.49

Least Square Regression method finds the line that minimizes the differences between observed and predicted values of the outcome.

Linear models

Linear models are linear functions, i.e., linear in their parameters, β0, β1 etc.

Polynomials are linear models

degree 1 – linear function $y = β0\ x + β1$

degree 2 – quadratic function $y=β0\ x^2 + β1\ x + β2$

degree 3 – cubic function $y = β0\ x^3 + β1\ x^2 + β2\ x + β3$
and so on..

Fit Linear models with NumPy

We will use two NumPy functions:

- **np.polyfit** is a function that performs the linear fit based on the **Least-Squares method** and returns the fitted parameters of a polynomial

- **np.poly1** it constructs a polynomial function from the parameters returned by polyfit. It constructs and returns the fitted model function.

```
coeff_array=np.polyfit(xarray, yarray, deg)
```

xarray, yarray - 1D arrays
deg: integer number specify the degree of the polynomial
coeff_array: 1D array containing the numerical values of fitted parameters: β0, β1, etc.

```
model_func = np.poly1d(coeff_array) #fitted model function
yp=model_func(xarray) #returns predicted yp values from the model
```

You can use directly both in one line of code:

```
model_func = np.poly1d(np.polyfit(xarray, yarray, deg))
```

Example – Fit a linear model with NumPy

step1 - Look at the raw data and choose the model

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt


#create data
x_data = np.arange(1,25)
y_data = x +6 * np.random.random(x.size)**2-1


#plot the raw data
fig, ax= plt.subplots()
ax.plot(x_data,y_data,'Dr',label='data')
ax.set_xlabel('x')
ax.set_ylabel('y')


#choose the model
#the model is y = β0* x + β1
```

Example – Fit a linear model with NumPy

step2 - Perform the fit and construct the fitting model function

```
#the model is y = β0* x + β1
coeff=np.polyfit(x, y, 1)
mymodel = np.poly1d(coeff)


print(mymodel.c)  #access the coefficients (fitted parameters)
[0.95374719 0.97329153] #[β0,  β1]

print(mymodel.order) #the order of the polynomial, its degree
1


#Create 1D array of predicted values yp for each element of x.
yp = mymodel(x)

#add the fitting model to the same Axes as the row data plot
ax.plot(x,yp,'b',label='model')
```

Example – Fit a linear model with NumPy

step3 - Calculating the R^2

$$R^2 = 1 - \frac{\sum_i \left(y_i - y_{ip}\right)^2}{\sum_i (y_i - \bar{y})^2} = 1 - \frac{SSR}{SST}$$

```
#You can calculate the R2 by applying the formula
SSR = ((y_data - yp)**2).sum()
SST = ((y_data - y_data.mean())**2).sum()

R2 = 1-(SSR/SST)
print(R2)
0.966040933792211
```

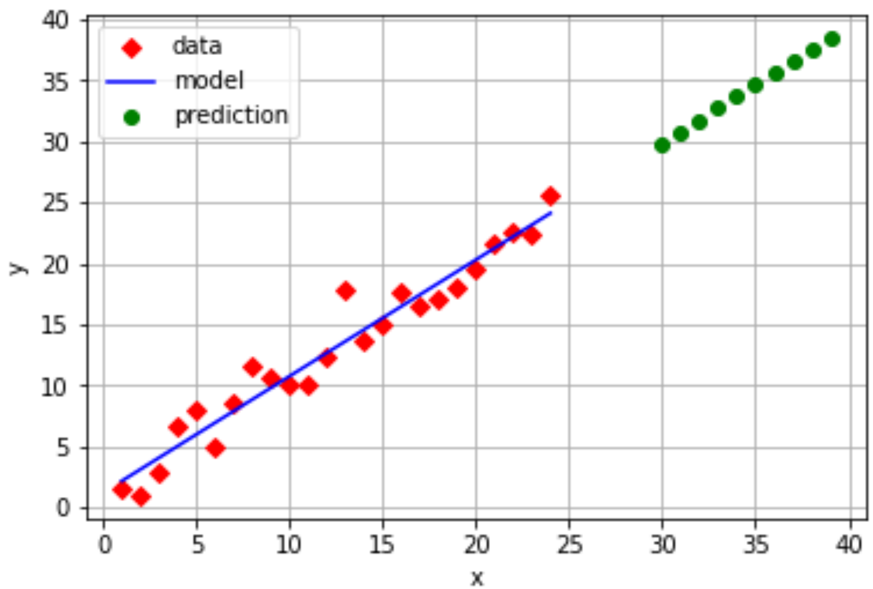
```
#or you can import the r2_score function from scikit-learn, the
Python machine Learning library

from sklearn.metrics import r2_score
print(r2_score(y,yp))
0.966040933792211
```

Example – Fit a linear model with NumPy

step4 - Make Predictions

If the model is good, you can use it to make predictions.

```
# predict y values for x values 30-40
xpred=np.arange(30,40)
ypred=mymodel(xpred)
ax.plot(xpred,ypred,'go',label='prediction')
ax.legend()
ax.grid()
plt.show()
```



To install **scikit-learn module :**

if you use Anaconda - type at the IPython Console or bash terminal

```
conda install scikit-learn
```

If you use standard Python3 IDLE:

```
pip3 install scikit-learn
```

To import it use:

```
import sklearn
```

The scikit-learn is a Python machine leaning library https://scikit-learn.org/stable/
You can also perform linear regression with scikit-learn or with SciPy.