Polynomial Regression ⇒ Relevant for non linear dataset

Comparison view betn Regressions (Equations) ↓↓

Simple linear Regression

$$\hat{y} = b_0 + b_1 x_1$$

multiple linear Regression

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + \ldots\ldots + b_n x_n$$

(we talk about coefficients)

Polynomial linear Regression

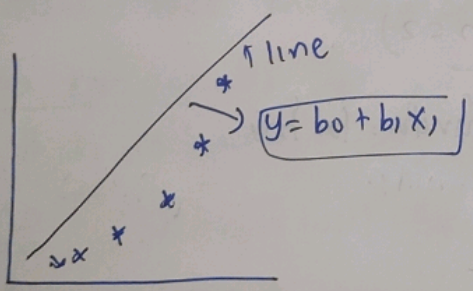
$$\hat{y} = b_0 + b_1 x_1 + b_2 x_1^2 + b_3 x_1^3 + \ldots\ldots + b_n x_1^n$$

for a same independent variable we use multiple powers

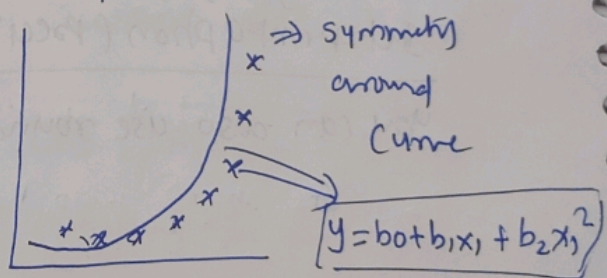
~~Power of independent go on increasing in polynomial linear regression~~

~~⇒ for a same~~

⇒ Sometime data not form symmetry around line but can for about some curve Ex:



Parabolic curve



⇒ Curves can be achieved using polynomial equations

⇒ It all dependent on the behaviour of dataset

⇒ It depends on condition you have & on top of it you will choose to select model.

⇒ In model we ~~have to~~ model try to find value of $b_0 \ldots b_n$ and so that we can fill in value of x & find value of (y)

Imp ⇒ polynomial linear regression is a special case in multiple linear regression

check code

If for some dataset we can't do splitting which can lead to disfunction of model.

#How to build polynomial regression model Code

so as we know

$$y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + \dots + b_n x_n$$

↑ for MLRM

now in polynomial we can convert polynomial regression model to multiple linear regression model such that

$x_1 = x_1$, $x_2 = x_1^2$, $x_3 = x_1^3$, $x_n = x_1^n$

So as it seems we have to create a new matrix of feature to convert a dataset which has only one feature to multiple linear regression using polynomial Powers of same independent feature

Create new matrix of feature

X_Poly = (new matrix of feature) so code

from sklearn.preprocessing import PolynomialFeatures → class

#Polynomial feature class will help us to convert normal matrix of feature to polynomial matrix of feature

Code

poly_reg = PolynomialFeature(degree = n)

we have to choose n

here we have to choose degree. that is n

(see change in result with change in n afterwards)

now fit_transform of matrix of feature (original)

X_Poly = Polyreg.fit_transform(X)

that is now we have new matrix of feature & now just train it like multiple linear regression model

as we increase degree performance increases

⇒ Polynomial goes performance increases with degree

(*) ⇒ heighr degres will reduce performance

Tricks for heighr resolution & smoother curves ⇒ not much useful.

Predicting a new result using build model

In this we are going pass own argument and check or predict output for inputted values

⊙ for both simple linear regression & polynomial

⇒ lin_reg.predict([[6.5]]).

↑ 2Darray

~~poly_reg.predict([[6.5]]).~~

Imp

We use polynomial feature to modify MOF and train it like MLRM

such like that you can predict

⇒ you have to enter as many feature, which you provided at time of build

(*) Polynomial takes argument on basis of Degree model

plr.predict(polyRegression) ⇒ output

↑

object of polynomial feature

polyRegression = PolynomialFeature(degree = 2)