Linear regression 5

11 December 2025 10:32

one hot encoding ⇒ Converting categorical variable to numeric

Car	origin
Tesla	USA
Ford	USA
Hyundai	ASIA
BMW	Europe

one hot encod. →

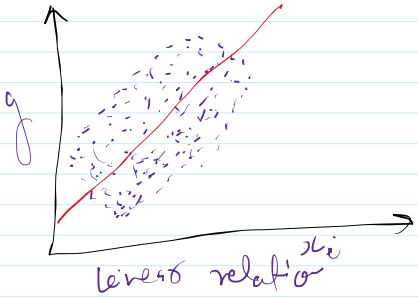
Car	origin_USA	origin_ASIA	origin_Europe
Tesla	1	0	0
Ford	1	0	0
Hyundai	0	1	0
BMW	0	0	1

only two col. is enough
the row where we have 0 and 0 can be interpreted as Europe

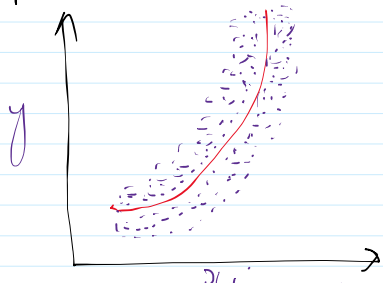
Assumption :

for any data linear regression model works when these assumption hold true

(1) Assumption of linearity → i.e. Relationship of any independent variable and dependent variable should be linear



linear relation



non-linear relation

(2) Features are not multi-colinear :-

(1) independent variables do not depend on each other

How to check Treat one independent variable as y and others as x

Calculate R^2 and VIF

$$VIF = \frac{1}{1-R^2}$$

if VIF > 10 very high multi colinearity

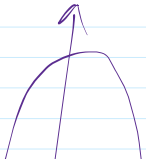
5 < VIF < 10 high multi colinearity

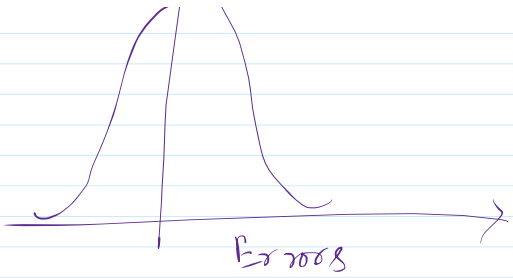
VIF < 5 Feature can be used

(1) Do the same for each independent variable

VIF ⇒ Variance inflation factor

(3) Errors are normally distributed :-

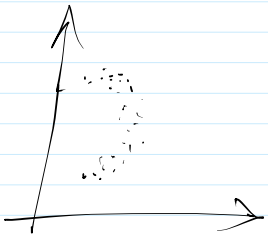




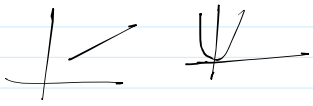
④ Target Vs. Error does not show any pattern

⑤ Predicted Vs. Error shows linear relationship

Polynomial Regression :-



Data is not linear



in equation polynomial regression tries x^2, x^2, x^4 etc. instead of x

example $x_1^2 + x_2^2 + x_3^2 + x_4^4 + x$

But how do we know what is good x^2 or x^3 or x^4 ?

① start with simple linear equation and calculate R^2 if it is very small e.g. 0.5

```
From sklearn.preprocessing import polynomialfeatures
Poly = PolynomialFeatures(degree=1)
X_poly = poly.fit_transform(X)
```

degree = 1 ⇒ linear reg.

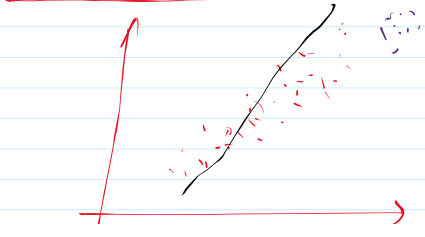
X =	degree = 1	degree = 2	degree = 3
1	1 1	1 1 1	1 1 1 1
2	1 2	1 2 4	1 2 4 8
3	1 3	1 3 9	1 3 9 27
4	1 4	1 4 16	1 4 16 64
5	1 5	1 5 25	1 5 25 125

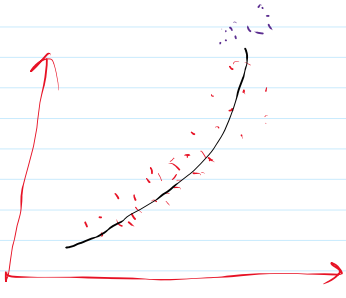
```
model = linear_regression()
model.fit(X_poly)
```

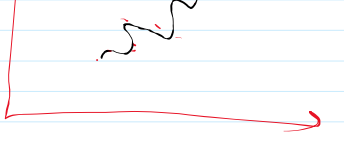
Data ⇒ LR ⇒ R^2 ⇒ Degree 2 ⇒ R^2 ⇒ Degree 3 ⇒ R^3

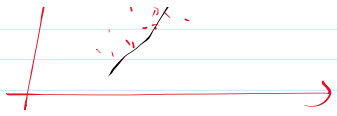
Overfit and underfit

Bias Vs. Variance :-





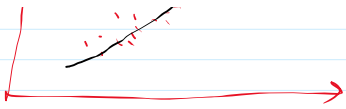




Model 1

Training Error is very high

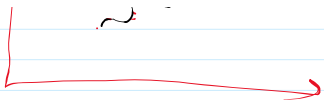
Model changing upon adding new data ⇒ low



Model 2

Training error is low

Model changing upon adding new data ⇒ medium



Model 3

Training error is super low

Model changing ⇒ very high.

Memorized the data

We prefer Model 2 ⇒ Why Model 1 is under fit

Model 2 is over fit

on test data Model 1 and 2 will perform bad.

in overfitted model for 2 data points which are very close prediction will be very far apart while in good model predictions would be quite close as well

overfitted model performs very well in testing but performs very bad in testing data

underfitted model performs very bad on both training and testing data

for good model or best fit model both training and testing errors will be low

Polynomial leads to overfit

$$y = c + m_1 x_1 + m_2 x^2 + m_3 x^3 + m_4 x^4 + m_5 x^5 + m_6 x^6 + m_7 x^7 \ldots$$

if we keep m_i very low and keep it decreasing as power goes up it will solve overfitting problem caused by high power value.

This is call regularization

Regularization:

normally Loss = MSE

in regularization

loss = MSE + m^2 for $y = mx + c$

loss = MSE + $(3)^2$ $y = 3x + 7$

Regularization adds a penality m^2 while calculating loss which make loss very high model tries to reduce the loss again while keeping m very low which is desirable to avoid overfitting caused by high order of power in polynomial regression making m low leads towards underfitting

$$MSE = \frac{\Sigma(y - (mx + c))^2}{n} + \Sigma(m_i^2)$$

$$y = m_1 x_1 + m_2 x_2 + m_3 x_3 + m_4 x_4 + m_5 x_5 \ldots$$

Algo will try to find a combination whose m is small and MSE is also small

$$MSE = \Sigma(y - (mx + c))^2 + \lambda \Sigma(m_i^2)$$

$$MSE = \frac{\Sigma(y-(mx+c))^2}{n} + \lambda \Sigma(m_i^2)$$

$\lambda \Rightarrow$ Super high will lead to underfit as it will lead model to choose m value very low

$\lambda \Rightarrow$ Super low will lead to overfit as it will lead model to choose m value very high

if training R^2 > testing R^2 + 5 %. then model is overfitting