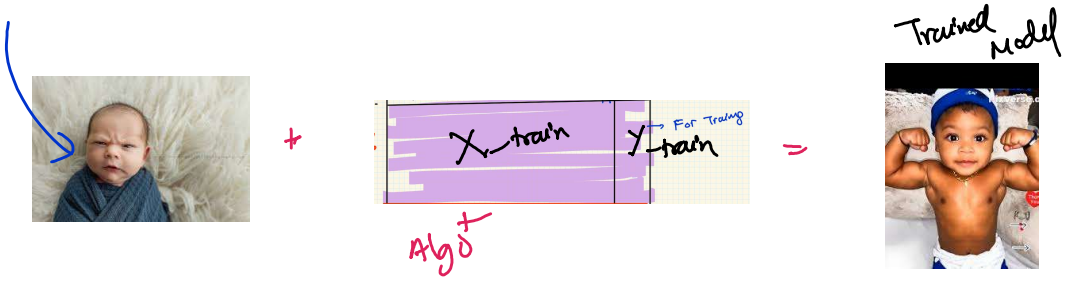
Linear Regression - 1

23 July 2025 20:52

Feature	Regression	Classification
Output Type	Continuous values (e.g., price, temperature)	Discrete labels (e.g., cat/dog, spam/not spam)
Objective	Predict a numerical value	Assign a class/category
Examples	House price prediction, stock price forecasting	Email spam detection, disease diagnosis

Linear Regression	Points (Experience)
Classes	4 [Assumptions, Mathematics]
Interview	<20LPA (Very Imp Algo.)
Characteristics	Simple, Elegant, Explainable
Fields	Finance, Sales, Business (CEOs)
Age	Old, (1920s)



Agenda → I ~~Cars~~ 24 Dataset
→ clean [Prepare]
→ Prepare for further processing

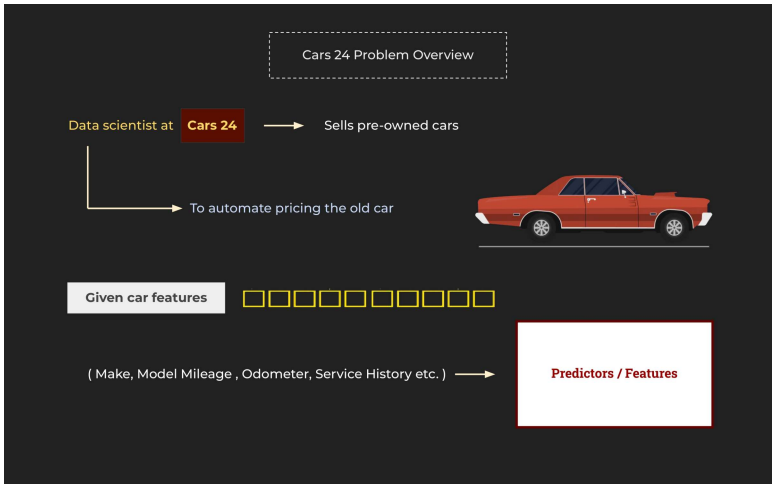
II ML - concepts (cont.) [General]

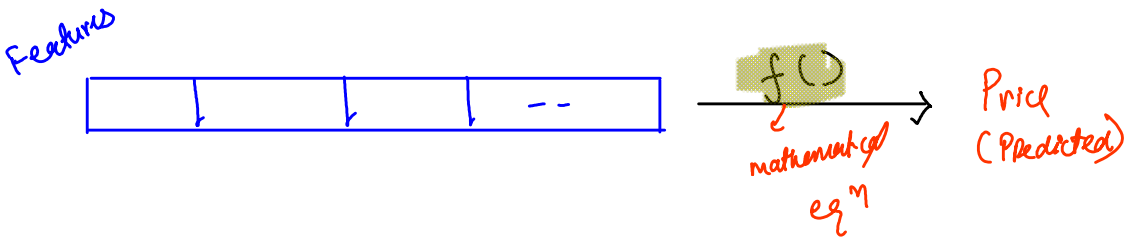
III Linear Regression (Basic Theory)
→ cont. in next classes

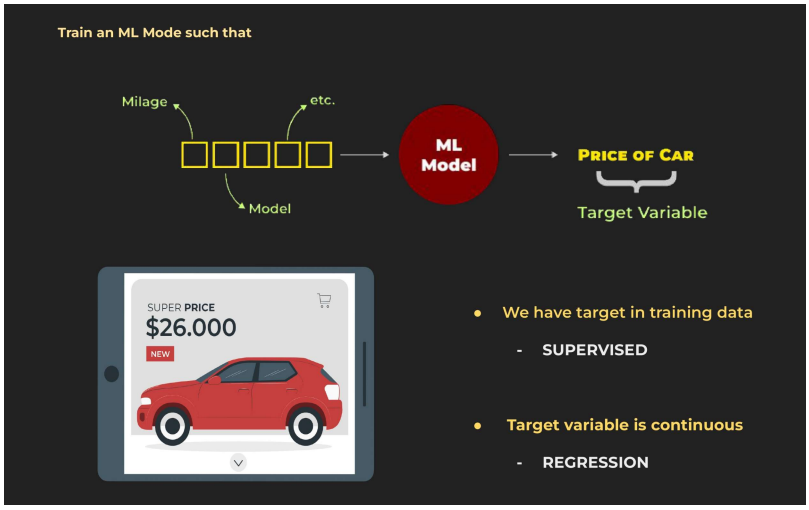
IV [Apply L-R on our data to predict car prices]

Feature	Regression	Classification
Output Type	Continuous values (e.g., price, temperature)	Discrete labels (e.g., cat/dog, spam/not spam)
Objective	Predict a numerical value	Assign a class/category
Examples	House price prediction, stock price forecasting	Email spam detection, disease diagnosis

Binary







Training Data

Features	Price

Target Encoding

Revision - Target Variable Encoding

Make: 41 categories

Using One Hot Encoding will result in a lot of dimensions -> curse of dimensionality

Hence, Target Encoding

Make	Selling Price
Maruti	1.2
Hyundai	5.5
Hyundai	2.15
Hyundai	2.26
Ford	5.70

Target Encoding

Make	Selling Price
1.2	1.2
3.3	5.5
3.3	2.15
3.3	2.26
5.7	5.70

Target encoding replaces the categories with a number representing the average target value associated with each category.

MEAN

Maruti - 1.2

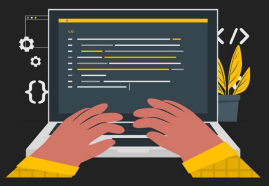
Hyundai - 3.3

Ford - 5.70

Replace

Line of Code

```
df['make'] = df.groupby('make')['selling_price'].transform('mean')
df['model'] = df.groupby('model')['selling_price'].transform('mean')
```



Revision - Scaling the Data

$$X_{Scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Mix - Max Scaler

Km_Driven
10,000
5000
2000

Scaling

Km_Driven
1
0.375
0

$X_{min} = 2000$

$X_{max} = 1000$

Note:

Since all the features are in different ranges, we need to scale the data.

Later we will see why is scaling important

Line of Code

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

df = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
```

Before

	selling_price	km_driven	mileage	engine	max_power	age	make	model	Individual	Trustmark Dealer	Diesel	Electric	LPG	Petrol	Manual	5	>5
0	1.20	120000	19.70	796.0	46.30	11.0	4.684721	1.180000	1	0	0	0	0	1	1	1	0
1	5.50	20000	18.90	1197.0	82.00	7.0	5.458819	4.818750	1	0	0	0	0	1	1	1	0
2	2.15	60000	17.00	1197.0	80.00	13.0	5.458819	3.394000	1	0	0	0	0	1	1	1	0
3	2.26	37000	20.92	998.0	67.10	11.0	4.684721	2.242676	1	0	0	0	0	1	1	1	0
4	5.70	30000	22.77	1498.0	98.59	8.0	5.858258	6.777576	0	0	1	0	0	0	1	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19815	6.50	69480	23.59	1364.0	67.05	6.0	10.532763	7.075000	0	0	1	0	0	0	1	1	0
19816	9.25	18000	17.50	1373.0	91.10	4.0	4.684721	7.128571	0	0	0	0	0	1	1	0	1
19817	4.25	67000	21.14	1498.0	103.52	8.0	7.182097	4.454000	0	0	1	0	0	0	1	1	0
19818	12.25	3800000	16.00	2179.0	140.00	7.0	7.315421	8.096522	0	0	1	0	0	0	1	0	1
19819	12.00	13000	18.00	1497.0	117.60	4.0	5.979902	11.014286	0	0	0	0	0	1	0	1	0

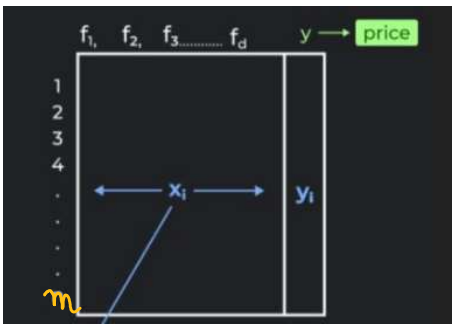
After

selling_price	km_driven	mileage	engine	max_power	age	make	model	Individual	Trustmark Dealer	Diesel	Electric	LPG	Petrol	Manual	5	>5
0.043684	0.031553	0.135345	0.117891	0.066506	0.310345	0.194048	0.041550	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0
0.252397	0.005237	0.128448	0.177281	0.123994	0.172414	0.232517	0.218382	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0
0.089795	0.015764	0.112069	0.177281	0.120773	0.379310	0.232517	0.149143	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0
0.095134	0.009711	0.145862	0.147808	0.100000	0.310345	0.194048	0.093193	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0
0.262104	0.007869	0.161810	0.221860	0.150709	0.206897	0.252367	0.313574	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0

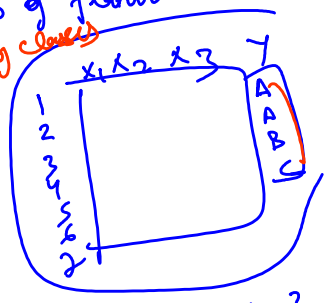
No harm in scaling

Data Notation

Tabular Data

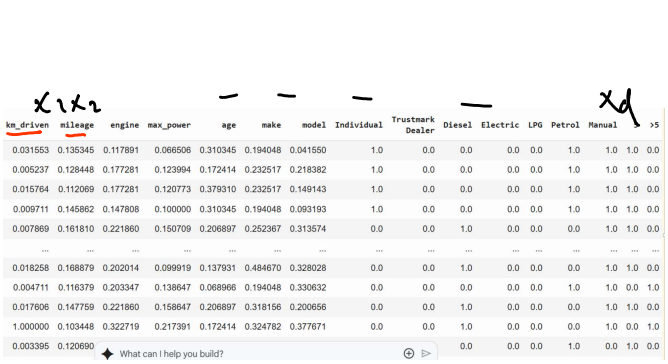


m → No of samples
d → No of features
n → No of classes



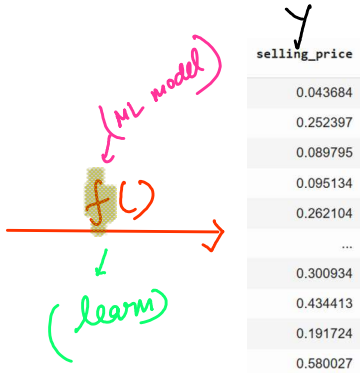
$n = 3$ $m = 7$ $d = 3$

Goal of Machine Learning

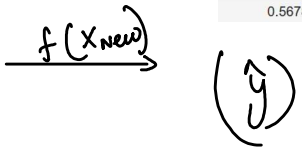


km_driven	mileage	engine	max_power	age	make	model	Individual	Trustmark Dealer	Diesel	Electric	LPG	Petrol	Manual	4	>5
0.031553	0.135345	0.117891	0.066506	0.310345	0.194048	0.041550	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0
0.005237	0.128448	0.177281	0.123994	0.172414	0.232517	0.218382	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0
0.015764	0.112069	0.177281	0.120773	0.379310	0.232517	0.149143	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0
0.009711	0.145862	0.147808	0.100000	0.310345	0.194048	0.093193	1.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0
0.007869	0.161810	0.221860	0.150709	0.206897	0.252367	0.313574	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0.018258	0.168879	0.202014	0.099919	0.137931	0.484670	0.328028	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0
0.004711	0.116379	0.203347	0.138647	0.068966	0.194048	0.330632	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0
0.017606	0.147759	0.221860	0.158647	0.206897	0.318156	0.200656	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0
1.000000	0.103448	0.322719	0.217391	0.172414	0.324782	0.377671	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0
0.003395	0.120690								0.0	0.0	0.0	1.0	0.0	1.0	0.0

What can I help you build?



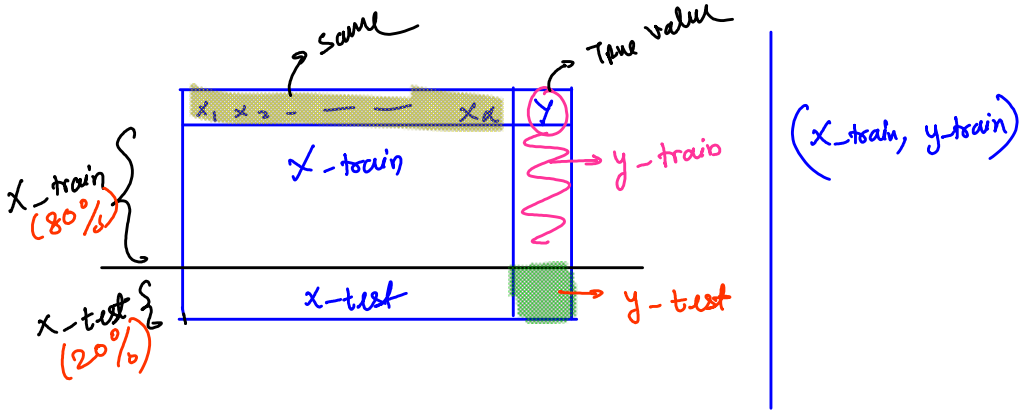
selling_price
0.043684
0.252397
0.089795
0.095134
0.262104
...
0.300934
0.434413
0.191724
0.580027
0.567892

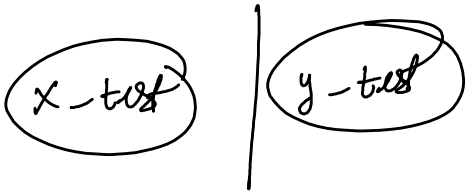


Goal: To generalise and so that we can make prediction on new cars

Train – Test Split

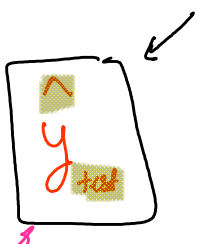
→ we don't use all data for Training





X_test ⇒

→ This has been trained on X_train, y_train



X_{test}	y_{test} (True)	$\hat{y}_{test}$ (given by Model)
① Alto	2 lakh	2.1 lakh
② Maruti	1.5 lakh	1.6 lakh
③ McLaren	22 lakh	21 lakh

→ How will my model is performing

Solution:

Because we don't know the ground truth (y) for New Sample

DO NOT use whole data for TRAINING

Typically,

80%

20%

Use this for training the data

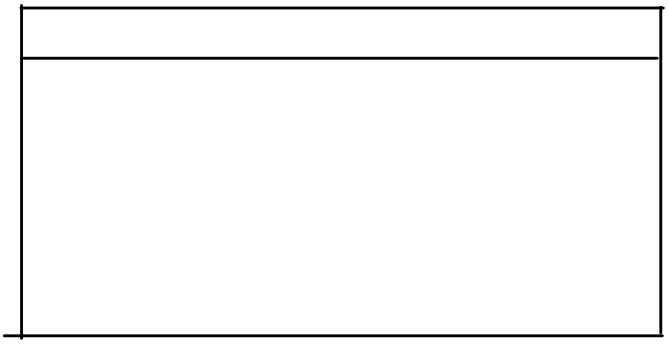
Use this as a proxy for unseen data (substitute)

'y' known

Check if: $\hat{y} \approx y$ → For evaluating model performance

Mean

Train test



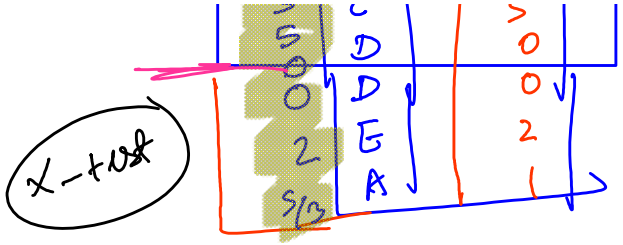
X_{train}

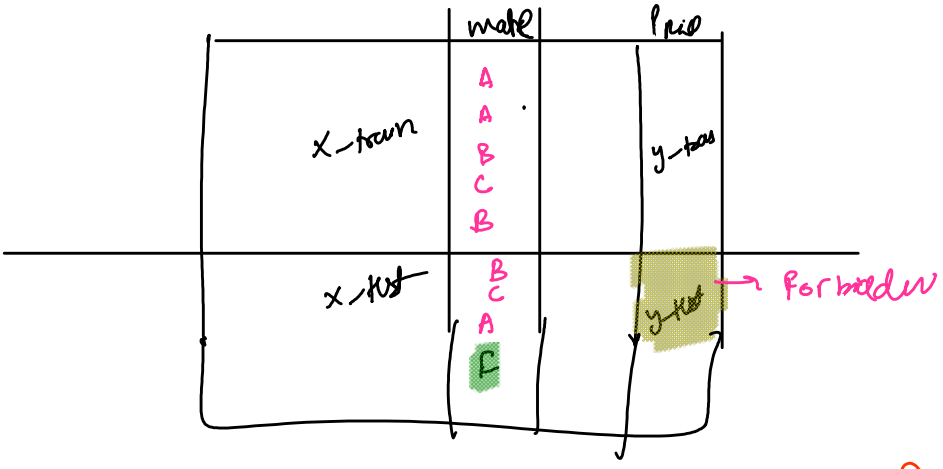
Model	Model	S.P
5/3	A → 2	2
5/3	A	2
3	B	3
5	C	5
0	D	0
0	D	0

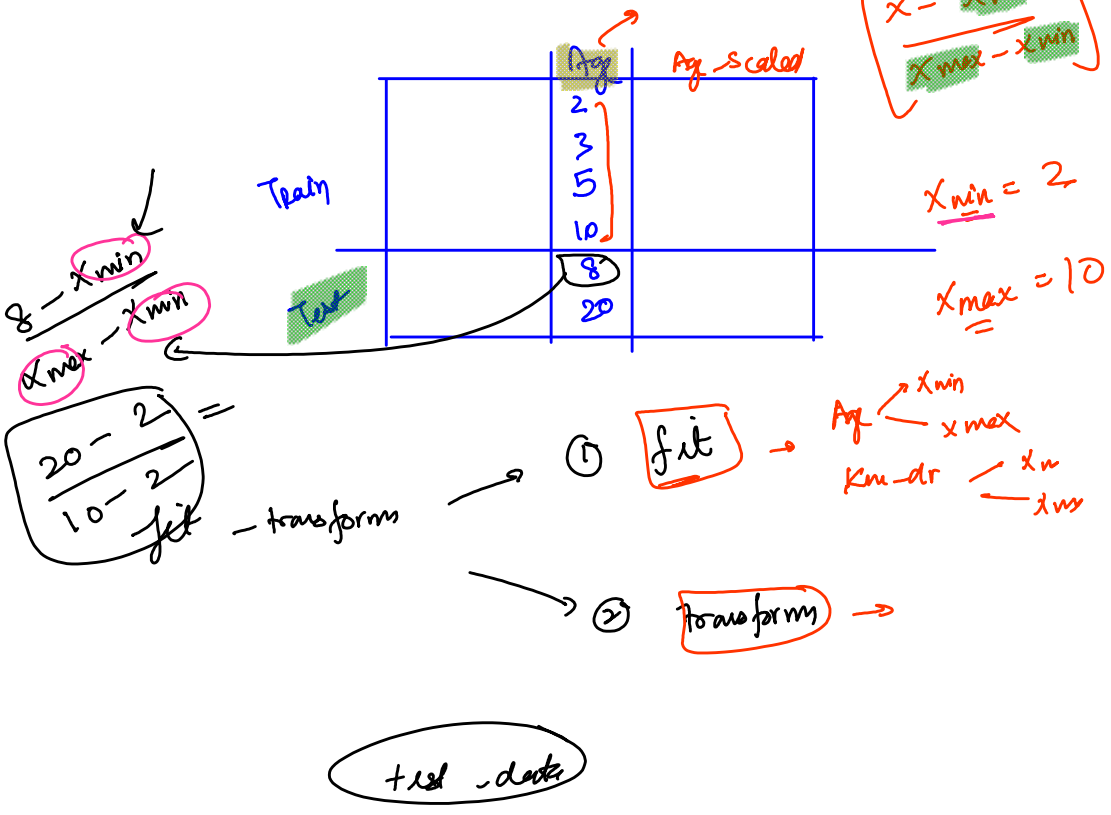
⇒

we have done Target Encoding

A → (2+2+1)/3 ⇒ (5/3)





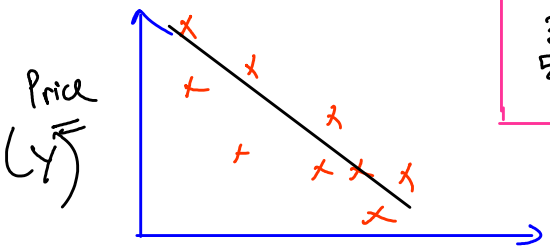


* Linear Regression

X_1	Y
Age	Price
2	10
3	6

2	10
3	6
5	3

① Age



$y = mx + c$ *

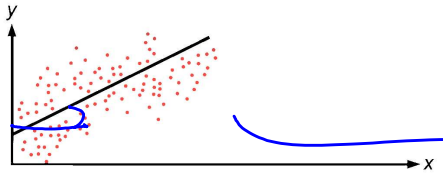
Price = m (Age) + c

Price = w (Age) + c

Geometrically → we are trying to fit a line through our data

① → Uni Variate LR

Linear Regression



Price = fn (Age)

Price = w (Age) + c

Linear → Linear fn

$y = f(x)$

↓

(Linear) $y = mx + c$

$y \neq mx + cx$

y is a function of x

$x^i \xrightarrow{f()} y^i$

$\hat{y}^i = f(x^i)$

f () can be x+2, x^2+3, sin (x), etc

1 predictor LR = Straight line

$y^i = mx + c$

$y^i = (w_1)x + (w_0)$

Geometrically,

Linear Regression is trying to find a line such that most of the points are close to the line

Learning?? → **Estimating best values of <w> vector**

Linear Regression using Sklearn

Univariate

Fitting the model on 1 predictor

```
X1=X[['model']]
X1_train = X_train[['model']]
X1_test = X_test[['model']]

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X1_train, y_train)
```

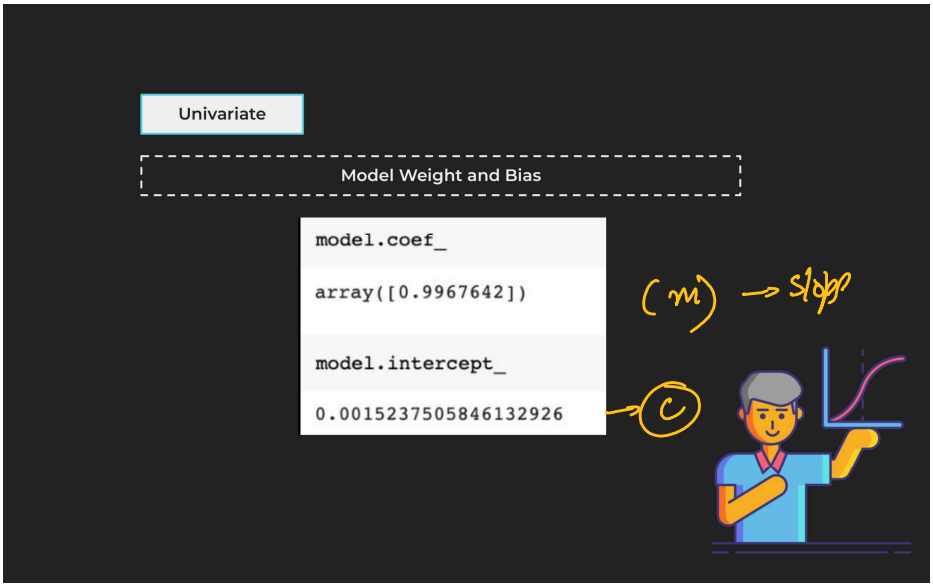
▾ LinearRegression

LinearRegression()

$y = mx + c$

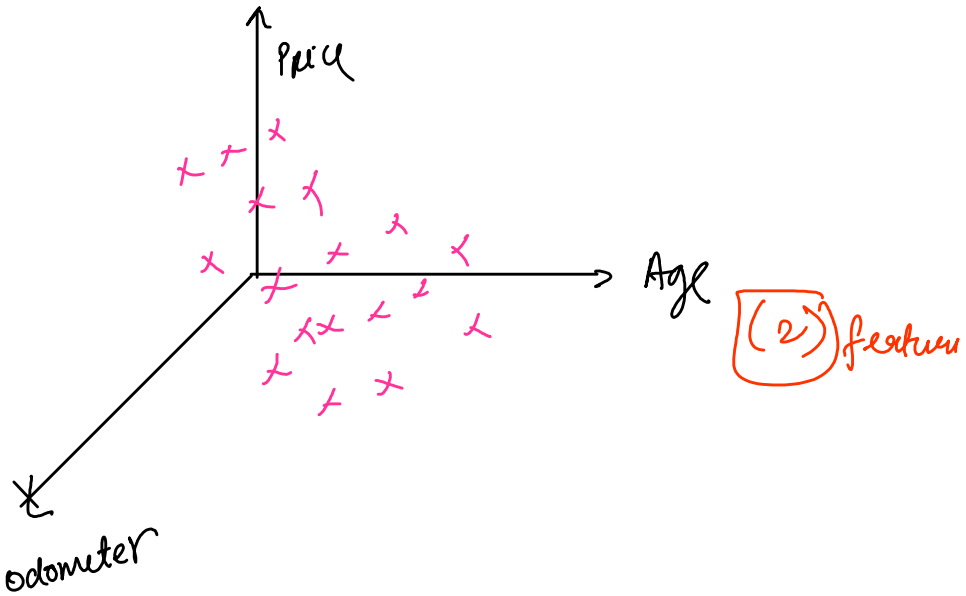
$y = mx + c$ → intercept (Bias)

coeff

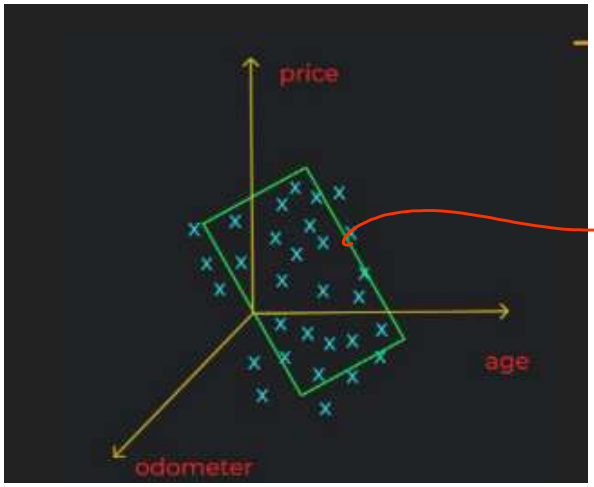


Multiple variables → MLRM

(2)



(2) features

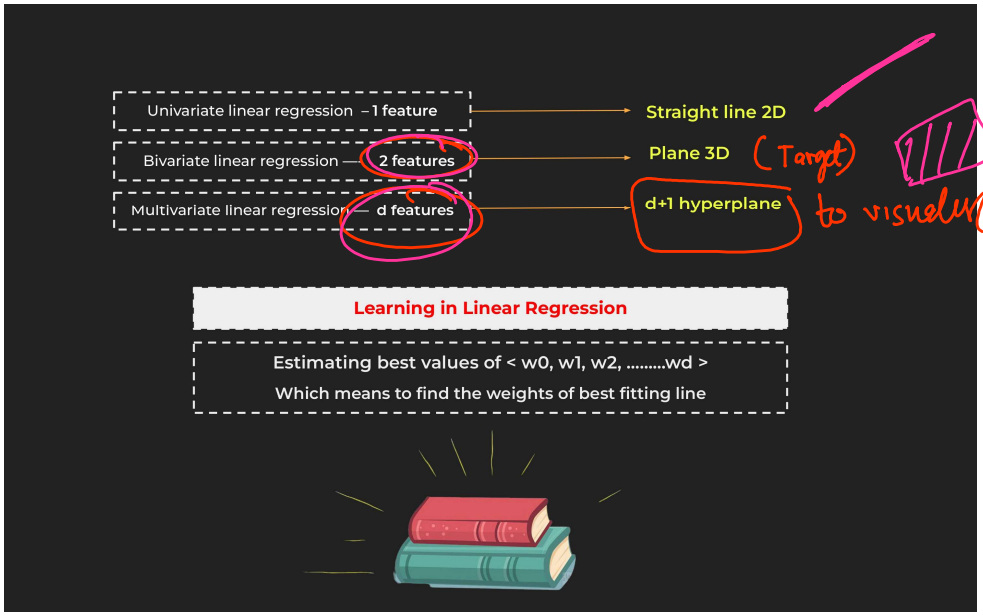


2 variables

Price = w_1 (Age) + w_2 (odometer) + w_0 (Bias)

(w_1: feature; w_2: wt; w_0: wt)

3 features → 4D {Hyperplane}

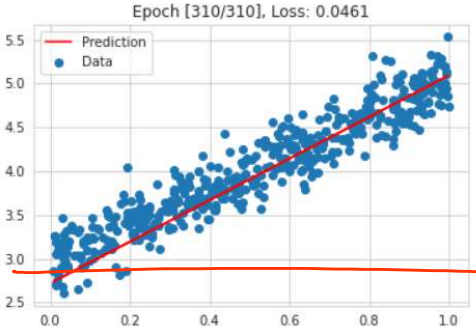


(Target) to visualize

3D

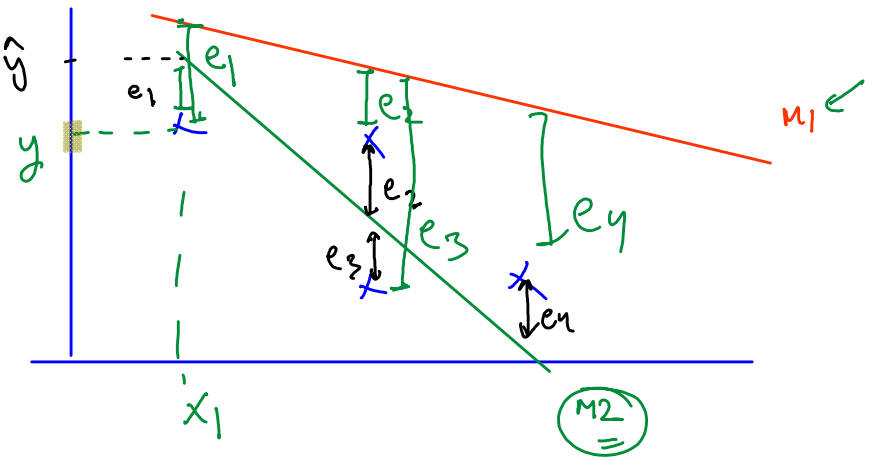
d+1

Machine Learning

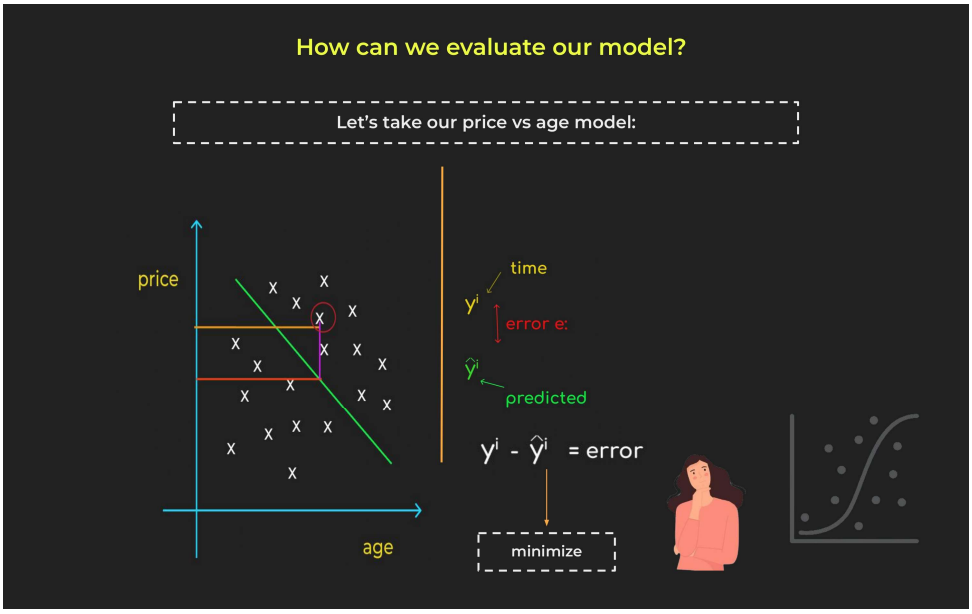


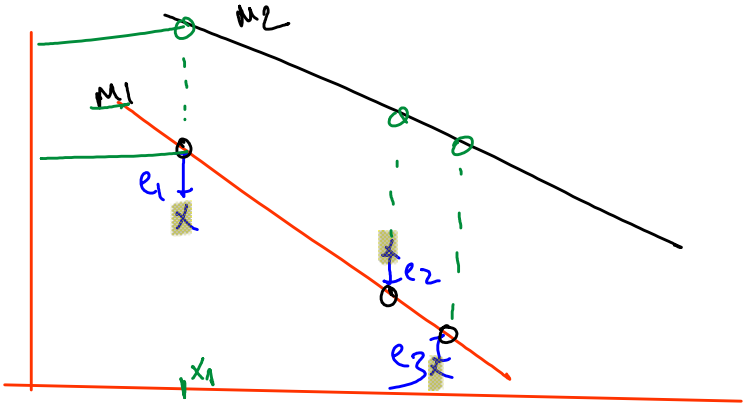
good or Bad?

Evaluation



$e = y - \hat{y}$





Red vs Black

$\sum e_i = e_1 + e_2 + e_3$

$M_1 \rightarrow \sum e_i = 10$

$\sum e_i = e_1 + e_2 + e_3$

good or Bad ?

$M_1 \rightarrow \sum e_i = 10$

$M_2 \rightarrow \sum e_i = 15$

* $e_i = y - \hat{y}$

So, for multiple points..

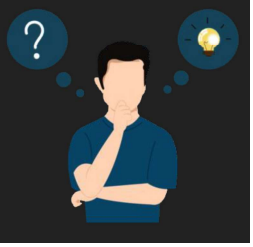
It will be the sum of the error divided by m : $\frac{1}{m}\sum_{i=0}^{m}(\hat{y}^{(i)} - y^{(i)})$

Can we use sum of errors as a metric?? → NO

	Error	Value
1	e1	3
2	e2	4
3	e3	-7

$$\sum_{i=1}^{m} e^i = 0$$

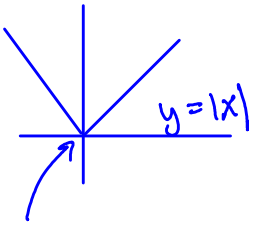
Perfect model?? WRONG!!



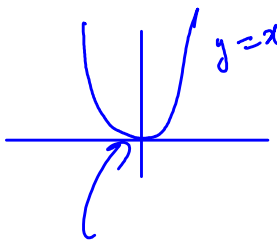
$\sum (y - \hat{y})$

Mod

$\frac{\sum |y - \hat{y}|}{m}$



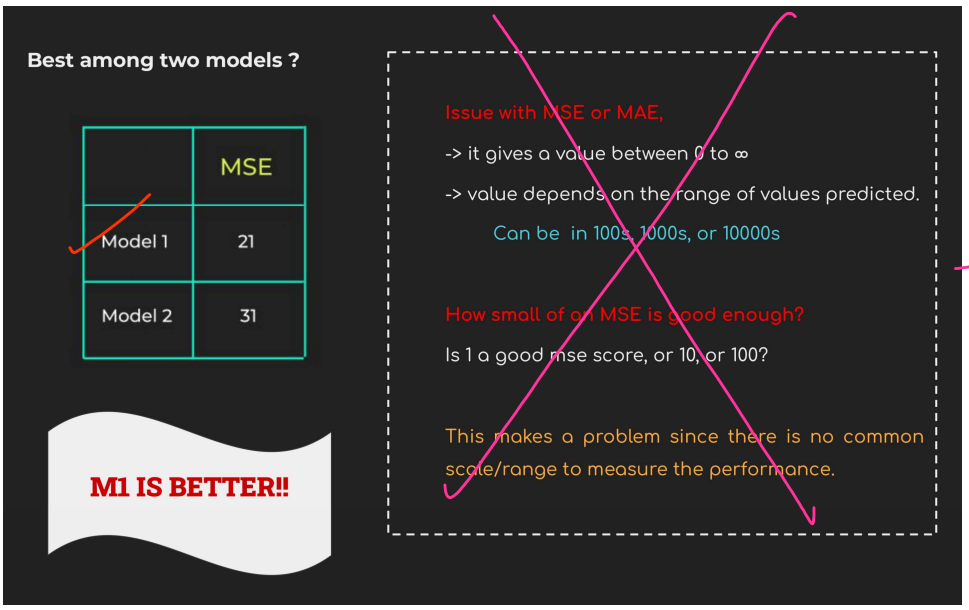
Mean Absolute Error



Square

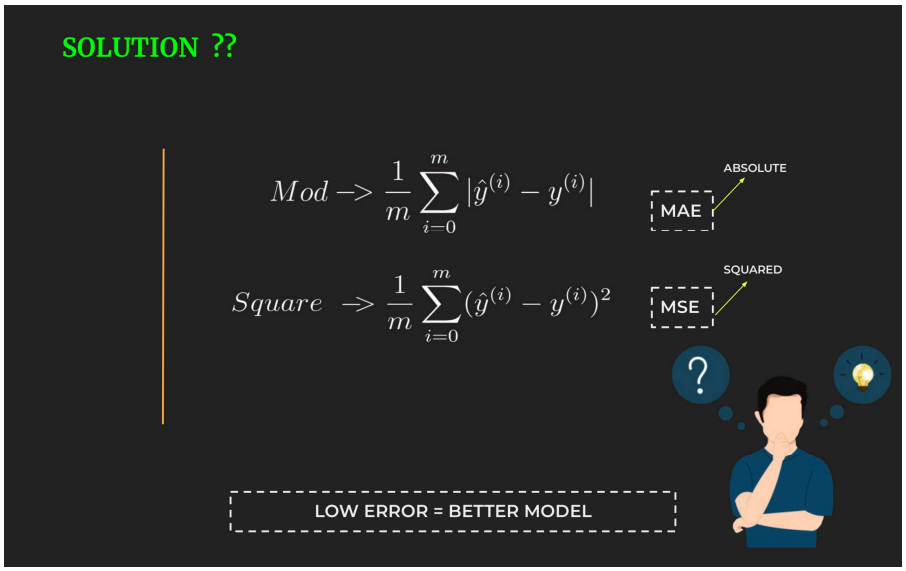
$\frac{\sum |y - \hat{y}|^2}{\sum e_i^2 \quad m}$

Can you compare two models ?



Dont Read

$\rightarrow R^2$



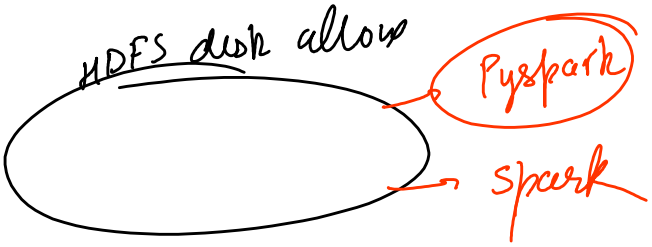
(Lin Reg 2)

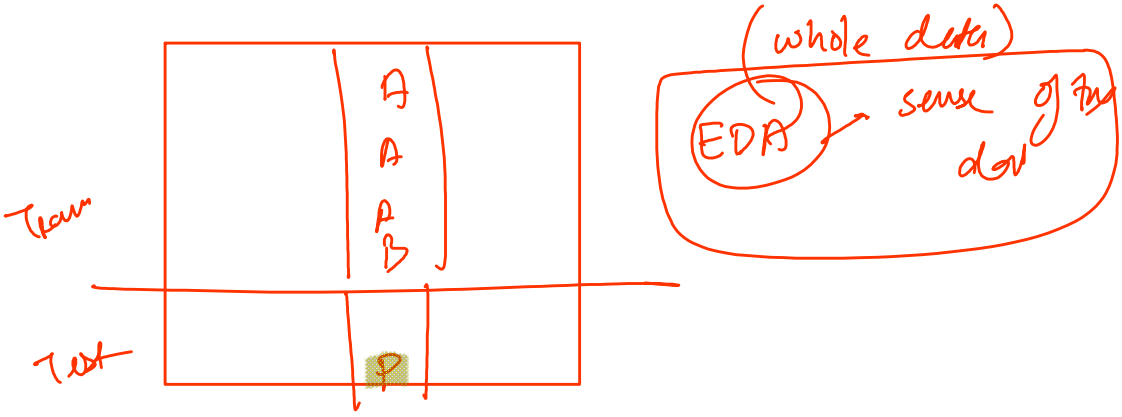
Doubts →

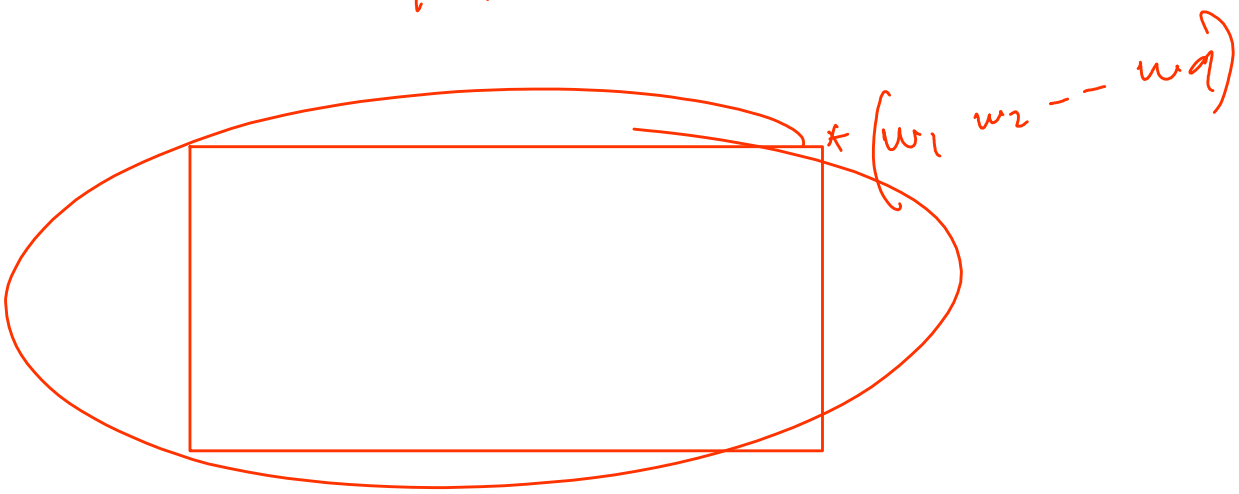
fit → find the parameter

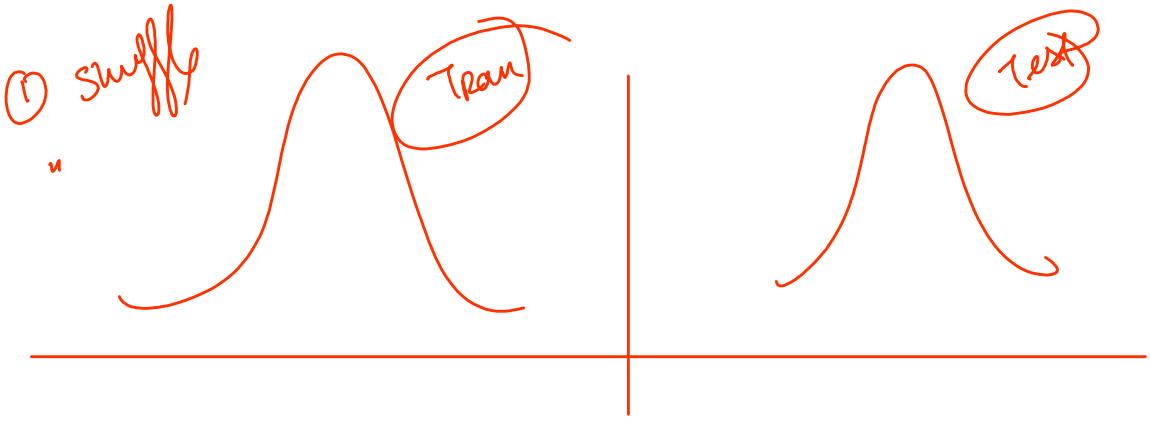
fit → find the parameter

(fit - transform) → find param + transform

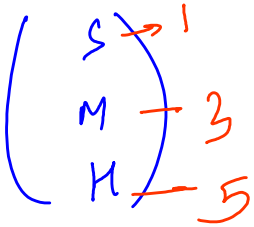








Label encoding → Business Unds → Stress Testing



?.