Introduction To ML HandWritten Notes

- Darshan R M

"Learn, Share, and Collaborate"

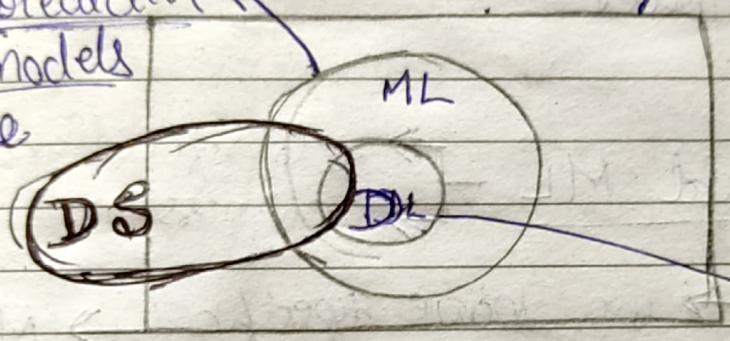
Machine learning

Darshan R.M.

Machine learning - Part-1

Week-13

* AI vs ML vs DL vs DS



AI → Smart application that can perform its own task without any human intervention
eg: self driving Car, Robots

It provides statistical to learn, analyze, visualize and develop predictive models from the data
eg:
Amazon
Netflix
→ Recommendation System

Deep learning - Mimic Human being
[Multi layered neural network]
eg: object detection, image recognition, chatbot, chat GPT, Recommendation System

Mathematics

1. Linear-algebra.
2. Statistics.
3. Probability.
4. Calculus.

* Types of ML

1. Supervised ML.
2. Unsupervised ML.
3. Semisupervised ML.
4. Reinforcement.

1. Supervised ML → classification, Regression

Dataset → we have specific output → Model trained to predict O/P

(independent feature) input

(i) classification

Ex: Play hours	study hours	o/p (dependent feature) Pass/ Fail
8	2	Fail
7	3	Fail
3	6	Pass

No of outputs are fixed.

O/P is Categorical feature → classification

if 2 categories binary feature

(i) Regression

o/p => Continous feature

Ex:

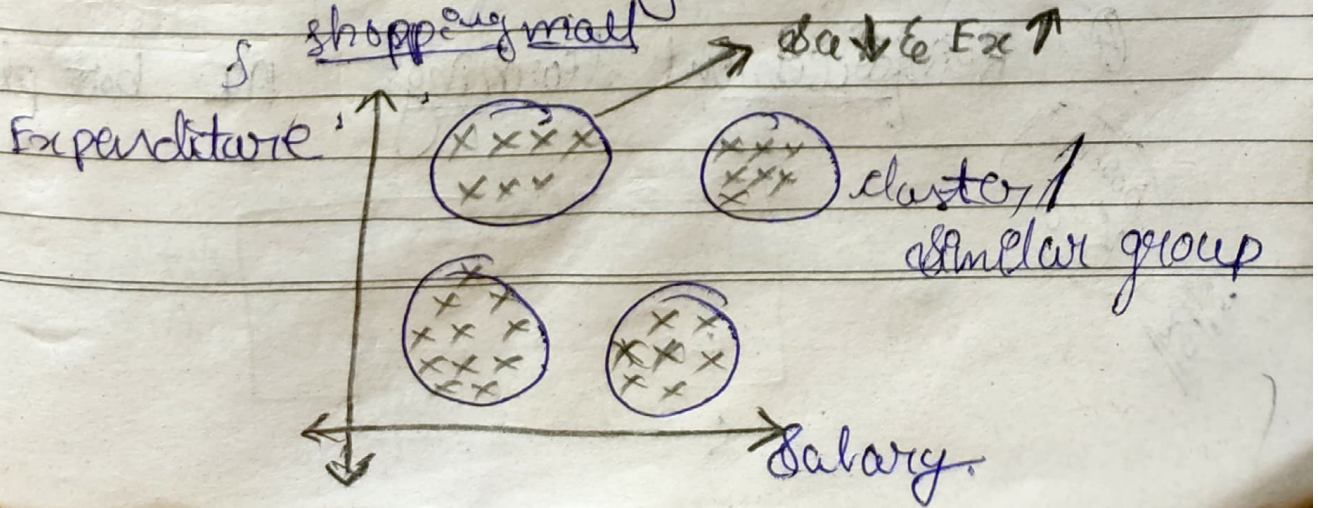
Size of house	No of Rooms	o/p Price → (Continous)

② Unsupervised ML:

Dataset → clusters / similar groups

Ex: Customer segmentation

shopping mall



To launch product → Based on clusters → Target the audience → 20% discount

* In Unsupervised → No app as such → we try to create clusters.

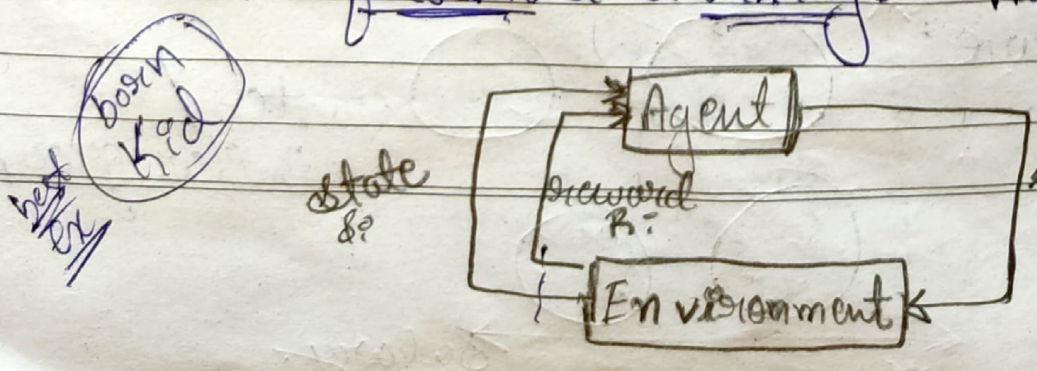
* after we can apply Supervised learning also

③ Semisupervised : Supervised + Unsupervised

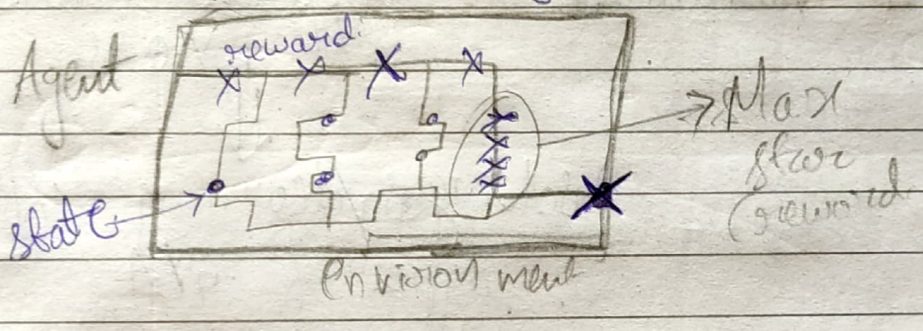
ex: Netflix has so many models based on region.

first it apply clustering → then → supervised to predict output

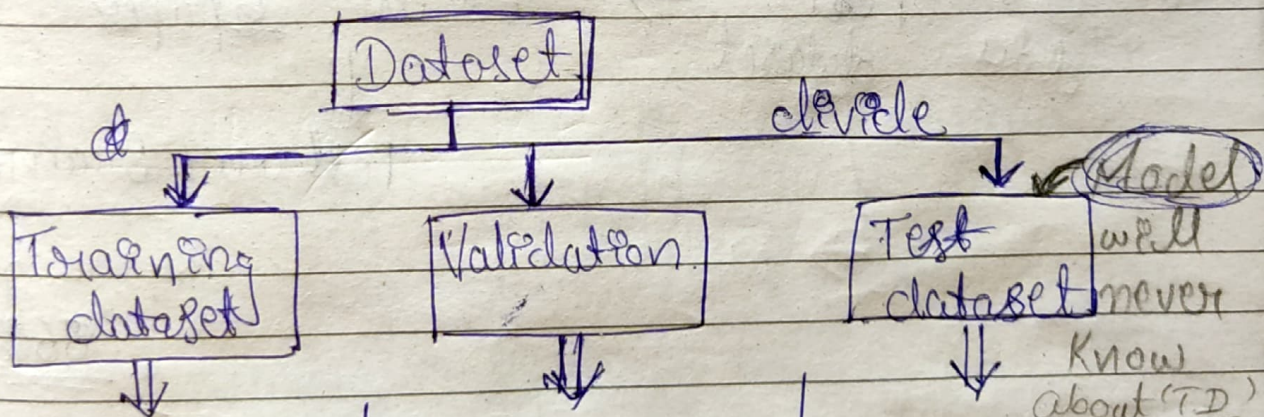
④ Reinforcement learning: AI bot playing



Reinforcement learning is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cummulative reward. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.



* Train, Test and Validation



We will Train the model

Hyperparameter Tunning of the Mode. we check some parameter to get more accuracy

Test the model [how model will work accuracy ??]

Ex:- Exam

Training dataset ⇒ Books

⇓

using Q/A ⇒ Train

Validation ⇒ different books and PYQ's

⇓

solving/learning different types of Q/A ⇒ tuning parameter.

Model should not know abt test data

Test dataset ⇒ Exam Paper

⇓

Test our Brain ⇒

we get "86%"

tendency of model to make systematic errors. / tendency of model to make random errors. /

* Bias, Variance, overfitting and underfitting

*Model performance ⇒ Accuracy ↑↑ → high

① Overfitting and underfitting

Dataset

Ex
Book ⇒ Train → Model is Trained ⇒ Accuracy ↑ → 95%
exam

exam ⇒ Test ⇒ Model is Tested ⇓

Brain is overfitted to Book ⇒ overfitting : Accuracy ↓ ⇒ 65%

⇓

Train: low Bias & [Training accuracy]
↑ accuracy ⇒ low Bais
↓ accuracy ⇒ hig Bais

Test: High Variance [Test accuracy]
↓ accuracy ⇒ high variance
↑ accuracy ⇒ low variance

② Model not properly trained with Train data ↓
Train → accuracy ↓ ⇒ 55% } ↓ underfitting
Test → accuracy ↓ ⇒ 50% } ↓
High Bias & High Variance
(More error)

③ Generalized model

Train ⇒ Accuracy ↑↑ ⇒ 85%
Test ⇒ Accuracy ↑↑ → 86%

⇓

our aim should be. low Bias and low Variance

* Feature engineering
* Handling missing values

Survey in Google form some feild are not falled.

3 mechanism why missing value occur

① Missing completely at Random, MCAR:
- missing is unrelated to both observed and missing data.
- No reason for missing.

② Missing at Random, MAR:
- probability of value being missing depends only on the observed data
- missing value are related to observed data.

③ Missing data not at random, MNAR:

→ prob of missing values depend on the value of missing data itself.

→ missingness is not random.

→ dependent on unobserved / unmeasured factors.

* > df.isnull() gives True if value is null

> df.isnull().sum() column null value Count

>#delete rows or data point

df.dropna() → rowise

axis=0

*disadva : loss of data

df.dropna(axes=1) → Column wise

column

* Imputation missing values

① Mean value imputation. → for normal distribution

> df['new'] = df['age'].fillna(df['age'].mean())

② Median value imputation ⇒ distribution other than normal distribution

```
> df['age_median'] = df['age'].fillna(df['age'].median())
```

③ Mode value imputation ⇒ for Categorical value.

```
> df['embarked'].notna()
> df[ ——— ].unique()   # possible values
> df.dtypes
> mode = df[ df[' — '].notna() ][' — '].mode()
> df.fillna(mode)
```

✱ Handling imbalance dataset

Dataset

classification → o/p in Categories

↓↓

2 outcome then Binary classification

1000 dataset

[900] Yes [100] No

ratio = 9:1 → imbalanced dataset

↓

[Model] → will be baised to Yes category

How to handle imbalance dataset?

→ ① Upsampling.
② Downsampling. (bad)

```
> df['target'].value_counts()
```

```
from sklearn.utils import resample
```

with replacement

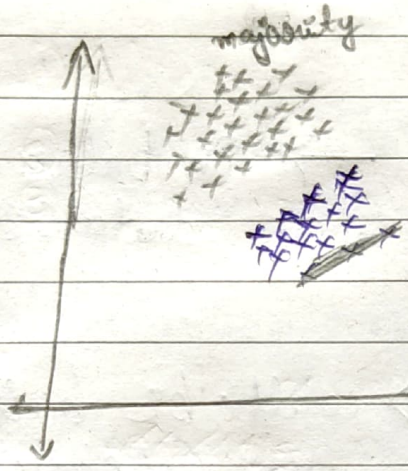
```
df['new'] = resample(df_minority, replace=True,
                     n_samples = len(df_majority),
                     random_state=42)
```

↳ set seed value

```
df.shape
(900,3)
```

* SMOTE

↳ Synthetic minority over sampling Technique



Upsampling

← Recorded similar data to minority

↓

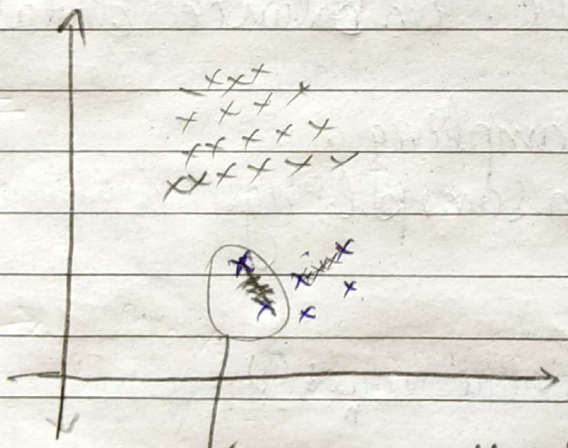
but variance (spread) is not increasing.

↓

overcome

↓

Interpolating



SMOTE will join data points and add points on that line

---

* > from sklearn.datasets import make_classification

↓

used to create binary / multiple class dataset

Concat

df1 df2

↓ ↓

> x, y = make_classification (n_samples = 1000, n_redundant = 0, n_features = 2, n_clusters_per_class = 1, weights = [0.9], random_state = 12)

[imbalanced]

* pip install imb.learn

'over_sampling'

> from imblearn.oversampling import SMOTE

oversample = SMOTE()

x, y = oversample.fit_resample (finaldf ['f1', 'f2'], final_df ['target'])

df1 df2

Concat

plot

> plt.scatter (df['f1'], df['f2'], c = df['target']

* Data interpolation.

↳ estimating unknown values within a dataset based on the known values.

1. linear interpolation.
2. Cubic interpolation. [scipy]
3. Polynomial interpolation.

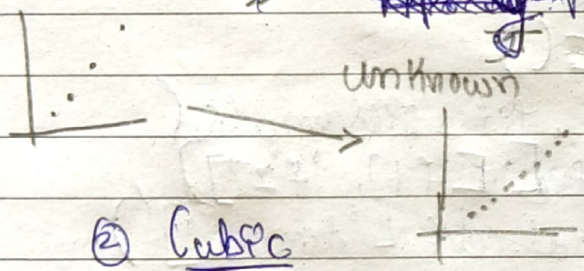
① linear

(1,5) 10 data points

x_new = np.linspace(1,5,10)

~~f~~ = np.interp(x_new, x, y)

↳ known



② Cubic

x =

y = x**3

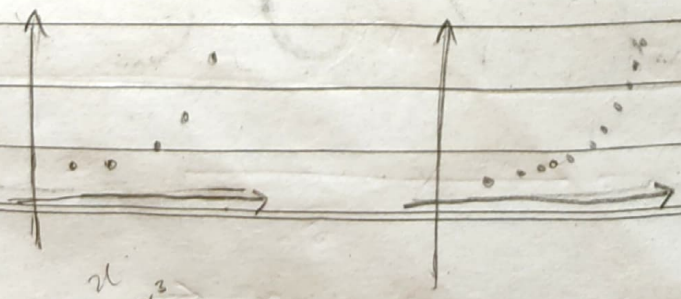
interpolate

from scipy.interpolate import interp1d

> f = interp1d(x, y, kind='cubic')

> x_new = np.linspace(1, 50, 10)

y_interp = f(x_new)



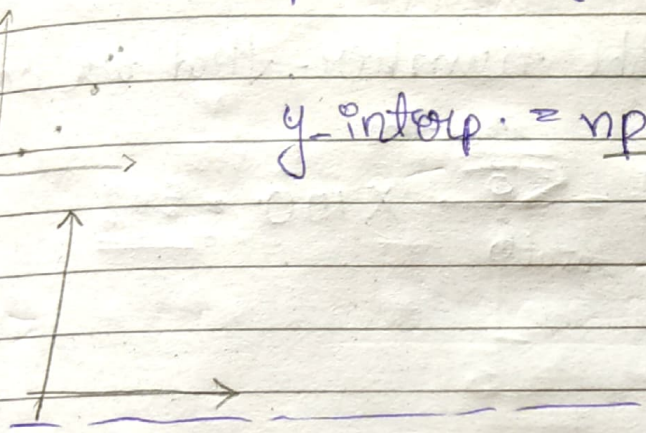
③ Polynomial.

x^2 $y = x^2$ Polynomial

$p = np.polyfit(x, y, 2)$

linspace(1,5,10)

$y\_interp = np.polyval(p, x\_new)$

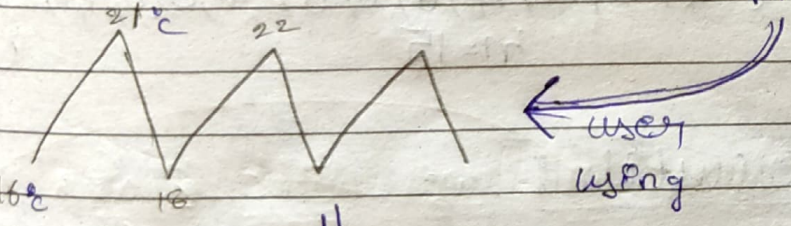


* Use case : (Krish) [panisonic]

AI model → provides → steep profile

↓

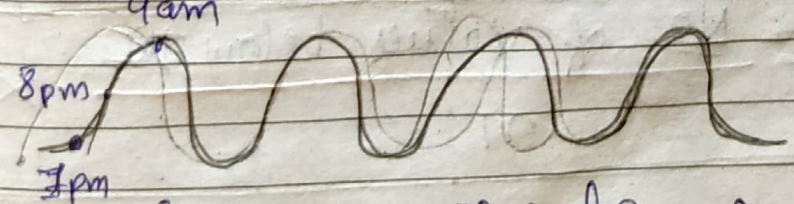
AC's → previous data.



↓

Electricity

lost of electricity waste as sudden pumping



So we create a sine wave instead of bump up

* Percentiles and Quartiles

(*) percentage: 1, 2, 3, 4, 5, 6

* what % of the number that are odd

$$\Rightarrow \%\text{ of the odd} = \frac{\text{fraction } 3}{\text{total } 6} \times 100 = 50\%$$

* percentiles

defn: A percentile is a value below which a certain percentage of observations (or) data points lie.

sorted values $\quad 1 \quad 2 \quad 3 \quad 4 \quad 5$

$x = \{2, 3, 3, 4, 6, 6, 6, 7, 8, 8, 9, 9, 10, 11, 12\}$

$n = 15$

75 percentile: we will find a value below which 75% of entire distribution lies.

① Percentile Rank of 10. $[x = 10]$

$$\text{percentile} = \frac{\text{No of values below } x}{n} \times 100$$

$$= \frac{12}{15} \times 100 = 80 \text{ percentile rank}$$

→ indicates that 80% of entire distribution falls below the value 10.

② What value exhist at percentile 25.

$$\Rightarrow \text{Value} = \frac{\text{Percentile}}{100} \times (n+1)$$

$n = 15$

$$= \frac{25}{100} \times 16 \rightarrow \text{total no of values.}$$

$$= 4 \rightarrow \text{index value}$$

if 4.2 / 4.5 / 4.7

then average $\left(\frac{4^{th} + 5^{th}}{2}\right)$

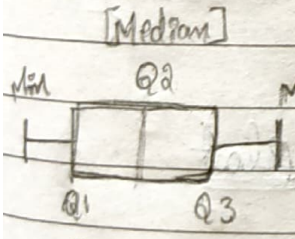
Value at 4^{th} value i.e. 4

* Quartiles

Q1 → 25 percentile

Q2 → 50 percentile [Median]

Q3 → 75 percentile



Boxplot

Ex° Ecommerce

Products	delivery time	location
—	2	—
—	3	—
—	1	—

Product Manager → some product takes for delivery → Task assigned To

90% ⟸ percentile ⟸ analyse the data

[gives product taking many days] [8 products]

↓

$x = 10$ for 90 percentile.

So. we have to get the details of the product that takes more than 10 days for delivery

---

* 5 number summary and Box plot

↳ To find outliers.

(i) Minimum
(ii) ~~First~~ Quartile (25 ~~Percentile~~ percentile): Q1
(iii) Second Quartile (50 percentile): Q2 (median)
(iv) Third Quartile (75 percentile): Q3
(v) Maximum

Removing the outliers

Q1 Q3 Q3

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

$x = \{1, 2, 2, 2, \boxed{3}, 3, 4, 5, 5, \boxed{5}, 6, 6, 6, 6, \boxed{7}, 8, 8, 9, 29\}$

↳ this 1 value is very big in dataset.

$n = 19$

[lower Fence ⟷ Higher Fence]

if any element
< lower fence
> higher fence
are considered as 'outlier'

IQR ⇒ Inter Quartile range ~~⇒ Q3 − Q1~~

lower Fence = Q1 − 1.5 (IQR)

Higher Fence = Q3 + 1.5 (IQR)

Q1 = 25 percentile = $\frac{25}{100} \times (19 + 1)$

$= 5$

↳ 5th value

$\boxed{Q1 = 3}$

Q3 = 75 percentile = $\frac{75}{100} \times (19 + 1)$

$= 15$

↳ 15th value

$\boxed{Q3 = 7}$

$IQR = Q3 - Q1 = 7 - 3 = 4$

$\therefore$ lower fence $= Q1 - 1.5(IQR)$
$= 3 - 1.5 * 4 = 3 - 6$
$= -3$

Upper fence $= Q3 + 1.5(IQR)$
$= 7 + 1.5 \times 4 = 7 + 6$
$= 13$

$[-3, 13]$

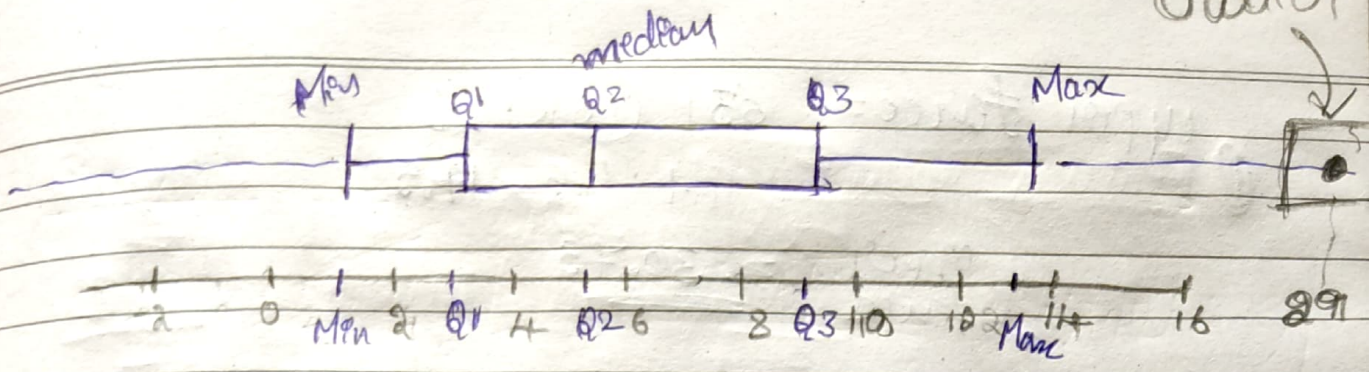
if any points come out of these range then they are called outliers.

Outliers in our dataset = 29 (as $29 > 13$)

(*) 5 number summary [Box plot] (excluding outlier)

Minimum = 1
$Q1 = 3$
(median) $Q2 = 5$
$Q3 = 7$
Maximum = 9

$19/2 = 9.5 \approx 10^{th}$ element



Assignment

① $Y = \{-13, -12, -5, -6, 3, 4, 5, 6, 7, 7, 8, 10, 11, 55\}$

$n = 14$

$\Rightarrow$

* $Q1 = 25^{th}$ percentile $= \frac{25}{100} \times (14+1)$

$= 3.75$

$Q1 = \frac{(3^{rd} + 4^{th})}{2} = \frac{(-5) + (-6)}{2} = \frac{-11}{2}$

$Q1 = -5.5$

* $Q3 = 75$ percentile $= \frac{75}{100} \times (14+1)$

$= 11.25$

$Q3 = \frac{11^{th} + 12^{th}}{2} = \frac{8+10}{2} = 9$

$IQR = Q3 - Q1 = 9 - (-5.5)$

$= 14.5$

lower fence $= Q1 - 1.5(IQR)$

$= -5.5 - 1.5(14.5)$

lower fence $= -27.25$

upper fence = Q3 + (IQR) 1.5

= 9 + 1.5 × 14.5

upper fence = 30.75

[-27.25, 30.75]

∴ % outliers = 55

Box plot =

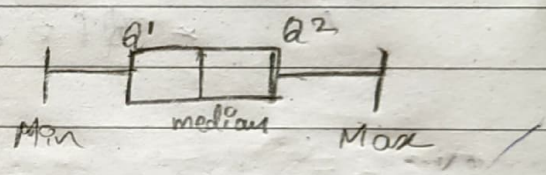
Min = -13

Q1 = -5.5

Q2 = 5.5

Q3 = 9

Max = 31



Handling outliers

min, Q1, med, Q3, max = np.quantile (list, [0, 0.25, 0.5, 0.75, 1.0

> plt.boxplot (list)

> sns.boxplot (list)

Feature selection

- Filter method.
- Wrapper method.
- Embedded methods.

Study after few ML algo

Feature extraction

def: Feature extraction is a process of selecting and extracting the most important features from raw data.

ML application → 1000 features

disadvantage

using all features may lead to
↑ in development time,
↓ in accuracy / model performance

Call features not required

most important features

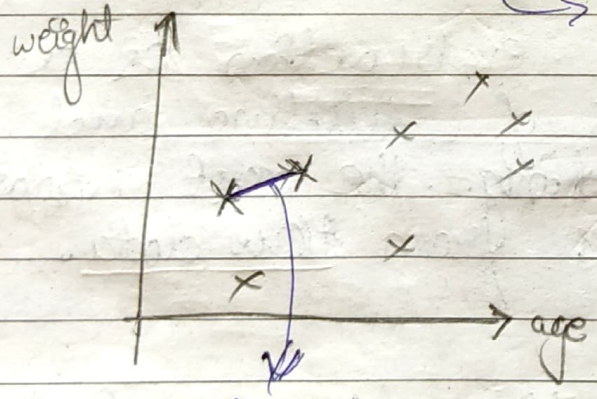
Train ML models.

① Feature Scaling:

	I/P			O/P
eg:	Age	weight	height	BMI
	32	70	140 cm	—
	28	75	160 cm	—
	35	80	155 cm	—
	(years)	(kg)	(cms)	

all features are calculated in different units

* There are some ML algo (KNN) which works on distance matrix.



distance calculation
as values big calculation will take time.

↓

So we will scale the data in some small range

ML technique

(i) Standardisation → [-3, +3]

Z-score $Z = \frac{x_i - \bar{x}}{\sigma}$

data convert $\left\{ \frac{32-\bar{x}}{\sigma}, \frac{28-\bar{x}}{\sigma}, \frac{35-\bar{x}}{\sigma} \right\}$

Standard normal distribution

SND → $\mu = 0, \sigma = 1$

deep learning - image data

(ii) Normalization [Min-Max Scaler]

↳ transform value b/w 0 and 1.
[0, 1]

Age
24
25
32
45
50
58

Transformation →

Age' [Min Max Scaler]

→ [0, 1] values

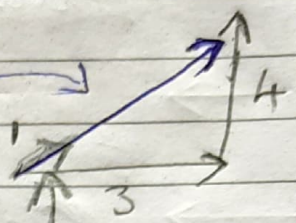
0.
0.

$$X_{scaled} = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

(iii) Unit vector

↳ Magnitude of 1

$\vec{x} = (3, 4)$



Magnetude

A/c to Pythagoras theorem,

$\|\vec{x}\| = \sqrt{3^2 + 4^2}$

$\sqrt{25}$

$= 5$

$\hat{\mu} = \frac{\vec{x}}{\|\vec{x}\|}$

Unit vector, $\hat{\mu} = \left(\frac{3}{\|\vec{x}\|}, \frac{4}{\|\vec{x}\|}\right)$

unit vector $\hat{\mu} = \frac{3}{5}, \frac{4}{5}$

$\|\hat{\mu}\| = \sqrt{\left(\frac{3}{5}\right)^2 + \left(\frac{4}{5}\right)^2}$

$= \sqrt{\frac{9}{25} + \frac{16}{25}}$

$= \sqrt{\frac{25}{25}} = \sqrt{1}$

Magnitude of unit vector is always = 1

$\|\hat{\mu}\| = 1$

② Feature selection:

↳ we just pick the most important features.

500 features —take→ Top 10 features

⇓ then

ML model Train.

To pick top 10 features..

(i) Filter method:

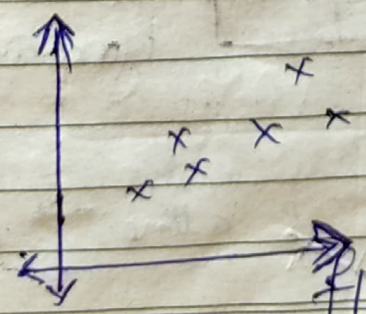
Ex: Correlation.

Vo Vo Imp (ii) Embedded method:

(iii) Wrapper method.

③ PCA [principal-Component analysis] part of unsupervised learning

(f1 f2) o/p → 2 Dimension [2D] point



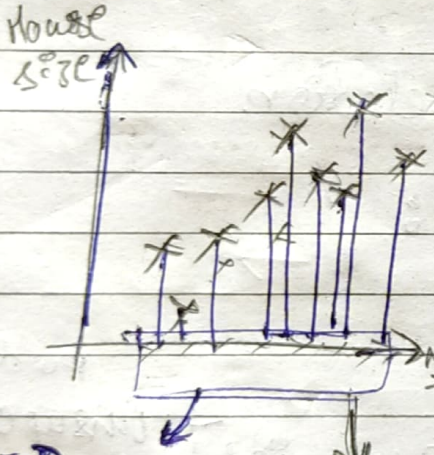
2D —Convert using PCA→ 1D ⇒ Here, there will be some loss of data

Ex: Dataset → independent feature

No of rooms	House size	Price
—	—	—
—	—	—

Feature selection.

select top k feature



→ ignoring House size entirely

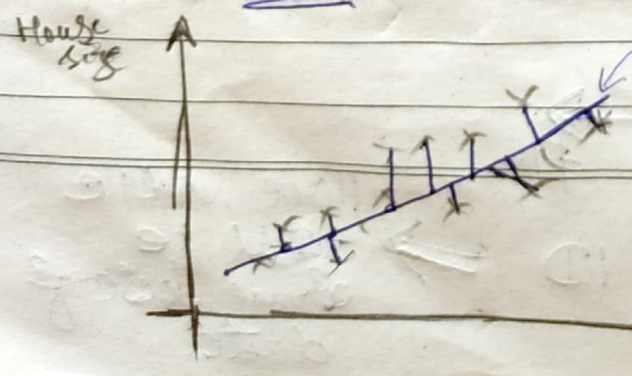
1-D

Taking only No of rooms

* There will "loss of information"

But, in PCA

Here, we are not ignoring a feature Completely that is advantage



creats a line → points on line are 1 D features

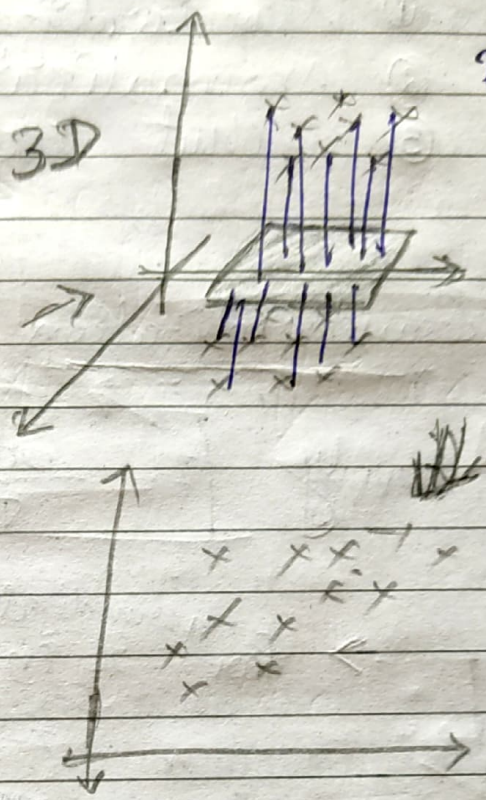
→ only "some loss of infor"

1000 D → 2D [Can be done]

↓

We cannot points ~~on~~ ~~4~~ D only

3D → 2D



* Higher dimension → to → lower dimension

* Feature Scaling

① Standardization
② Normalization - Min Max Scaler
③ Unit Vector

① Standardization

apply $\left[Z = \frac{x - \mu}{\sigma} \right]$

np.mean()
np.std()

> for normalized x in list(df[ ]):
    Z-score = (x - mean) / std
    normalized.append(Z_score)

* from sklearn.preprocessing import StandardScaler
  scaler = StandardScaler()
  scaler.fit(df[['data-bru']]) → calculate required values
  scaler.transform(df[[ ]]) → transform the feature
  (or)
  scaler.fit_transform(df[[ ]]) → for the data

② Normalization - MinMaxScaler

> from sklearn.preprocessing import MinMaxScaler

> min_max = MinMaxScaler()

> minmax.fit_transform(df[[-,-]])

for any new data > min_max.transform(df[[-,-]])

③ Unit vector

> from sklearn.preprocessing import normalize

> normalize(df[[-,-]])

Data Encoding

① Nominal / one hot encoding.
② Label and ordinal Encoding.
③ Target guided ordinal Encoding.

why? eg: Exp | Degree | Salary

B.E
PHD
Master

Our model cannot understand this values but only numerical value can be understood

Categorical variables → Data encoding → Numerical value

Model can perform Computation

our model cannot understand categorical data

↓

Categorical data → is converted into → Numerical data.

Numerical data ⇑ our ml model can perform some computation.

one of the technique to convert it is data encoding

datasize (>5) we use

① Nominal encoding / OHE (OneHot Encoder) encoding

⇒ one hot encoding (OHE) [Red green Blue]

represented as a "binary vector"

ex: Colors

Categorical: Red, green, Blue ⇒

	Red	green	Blue
Red	1	0	0
green	0	1	0
Blue	0	0	1

* disadvantage

① If we have 1000 features then 1000 new features are created.

② Sparse matrix ⇒ overfitting [low Bias & high variance]

Scikit learn

> ~~Import~~ from sklearn.preprocessing import OneHotEncoder

> encoder = OneHotEncoder()

encoded =
> encoder.fit_transform(df[['color']]).toarray()

alphabet	blue	green	red
0	0	0	1

>

Encoded_df = pd.DataFrame(encoded, columns = encoder.get_feature_names_out())

New data >
encoder.transform([['blue']]).toarray()

=> [[1.0 0.0]]

> pd.concat([df, encoded])

② label (or) ordinal encoding :

↳ assigning a unique label to each category in the variable.

may be based on according alphabet (or) frequency

>> from sklearn.preprocessing import LabelEncoder

> encoder = LabelEncoder()

> encoded = encoder.fit_transform(df['...'])

> encoder.transform([['red'],['green'],['blue']])

different values for 1 feature

[[...],[...],[...]]

[[...,...,...]]

multiple values for 1 feature.

---

③ Ordinal encoding:

↳ encode data that have intrinsic order (or) ranking.

> from sklearn.preprocessing import OrdinalEncoder

> ord_Encoder = OrdinalEncoder(categories = [['small', 'medium', 'large']])

tell the order of ranking to be given

"large is good" so highest rank

>

ord_Encoder.fit_transform(df['size'])

>

④ map categorical to numerical ← Convert into dictionary ← ③ ② take mean of each group ← ① group categorical value

④ Target guided ordinal encoding :

→ Encode based on their relationship with the target variable.

→ useful when : large number of unique categories.

→ Replace Categories in Categorical variable by → numerical value based on mean/median of target variable

→ Create : monotonic relationship.
Improve : predictive power.

```
> mp = df.groupby(['city'])['price'].mean().to_dict()
```

based on, feature

⇒

city	
london	150.0
newyork	190.0

```
> df['city_encod'] = df['city'].map(mp)
>
```

measures relationship b/w 2 variables

coefficient value tells the strength and direction of relationship

* Covariance and Correlation

→ imp EDA & Feature engineering.

① Covariance

degree of association

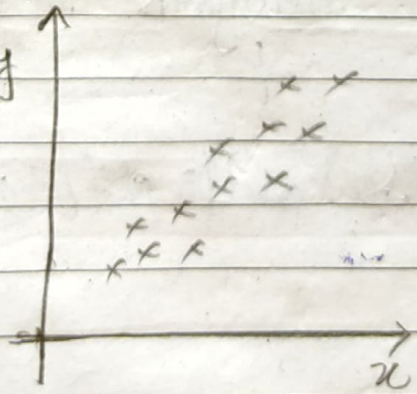
x	y
2	3
4	5
6	7
8	9

{relationship between x and y}

x ↑ y ↑
x ↑ y ↓
x ↓ y ↑
x ↓ y ↓

size location → Price

I/p [independent] O/p

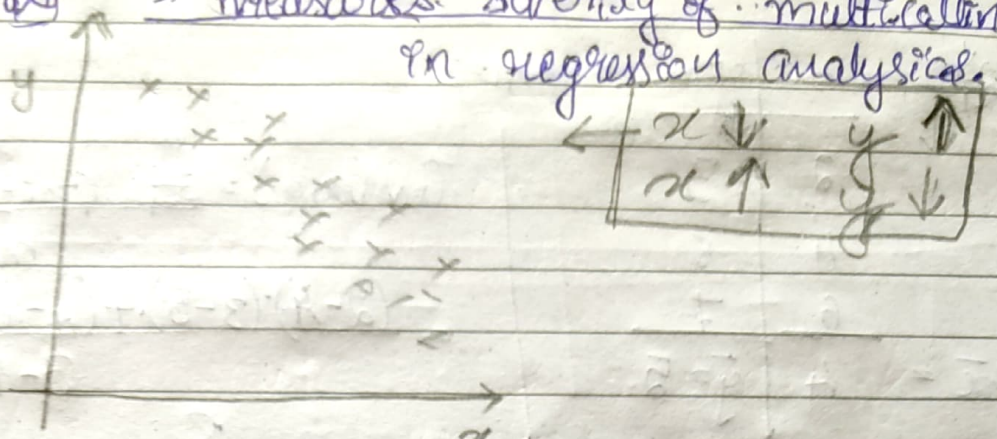


x ↑ y ↑
x ↓ y ↓

VIF → Variance inflation factor. (↑ → ↑ in level of multicollinearity)

VIF = 1 ⇒ no multicollinearity

[0, ∞] → measures severity of multicollinearity in regression analysis.



$$\text{Covariance}(x,y) \quad (\text{or}) \quad \text{Cov}(x,y) = \sum_{i=1}^{n} \frac{(x_i - \bar{x})(y_i - \bar{y})}{(n-1)}$$

$$\text{Variance}(x) \quad \text{Var}(x), \sigma^2 = \sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n-1}$$

⇓

Variance tells the spread of data.

$$\text{Variance}(x) = \text{Covariance}(x, x)$$

↓

how x is related to itself.

$x \uparrow$ $y \uparrow$
$x \downarrow$ $y \downarrow$ ⇒ +ve cov ⇒ tells that x and y are positively correlated.

$x \uparrow$ $y \downarrow$
$x \downarrow$ $y \uparrow$ ⇒ -ve cov

Q)

x	y
2	3
4	5
6	7
$\bar{x}=4$	$\bar{y}=5$

$n=3$

$$Cov(x,y) = \frac{\sum_{i=1}^{n} (x_i-\bar{x})(y_i-\bar{y})}{n-1}$$

$$= \frac{(2-4)(3-5)+(4-4)(5-5)+(6-4)(7-5)}{3-1}$$

$$= \frac{8}{2}$$

$Cov(x,y) = 4$ [+ve]

⇓

We can tell that x and y are positively correlated.

(2)

Advantage

① Relationship b/w x and y

↓

may be +ve/-ve

Disadvantage

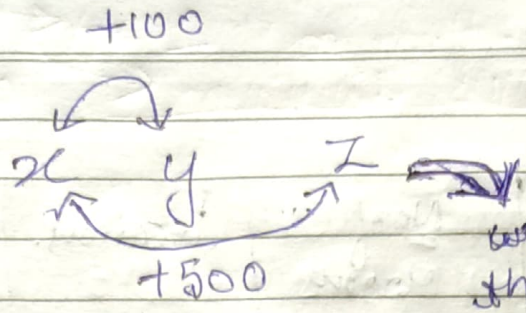
① we don't have a limit/range for the values of covariance.

↓

To overcome [magnitude matter]

we have spearman correlation and pearson correlation coefficient

(range → [-1, +1])



we cannot conclude that x is highly correlated to y than z

7-5

② Pearson Correlation coefficient [-1 to +1]

$$\rho_{(x,y)} = \frac{cov(x,y)}{\sigma_x \times \sigma_y}$$

standard deviation

Range ⇒ [-1, +1]

* The more the value towards +ve the more positively correlated they are.

* The more the value towards -ve the more negatively correlated they are. [x & y]

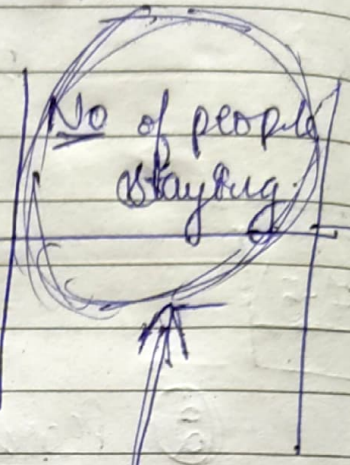
Use Case:

Dataset: 1000 features [Md models]

(Taking all feature and training is not so important.)

Ex:

Haunted	Size of the house	No of rooms	location	No of people staying
—	—	—	—	



To predict price
No of people staying will not help

not correlated to price

So, we do Feature selection
where we select only features which are "highly correlated" to ("price".) output.
and
less correlated, can be dropped. ← Feature.

(3) Spearman rank correlation: → better than Pearson
↳ we consider rank

$$r_s = \frac{Cov(R(x), R(y))}{\sigma_{R(x)}\ \sigma_{R(y)}}$$

op Price	x	y	R(x)	R(y)
-	1	2	5	5
	3	4	4	4
	5	6	3	3
	7	8	2	1
	0	7	6	2
	8	1	1	6

highest

rank 1 → assigned to → highest score value

* python → Covariance and Correlation

> df.cov()

> df.corr(method = 'pearson')

> df.corr(method = 'spearman')

diagonal elements will always be 1.