Data

Data given in form of csv/excel

Data is in the format of files (.jpg, .png, .txt)

↓

making use of paths

Extract the data then convert into dataframe

↓

Creating new features from available features which are required for predicting output

↓

Treating missing values & outliers

↓

EDA part

↓

Identify target variable & required features as input

↓

Splitting test & train

↓

Cleaning to be done in this process

↓

Splitting & rescaling both numerical & categorical to numerical data

↓

Then Model training & testing

Only fit_transform

↳ Only transform

↓

predictions

↓

Mode Save

↓

Web API

Errors

* In cmd Import Error i.e., builder.py is not found
  upgrade pro

* Unicode error during path without r before it
* Sparse matrix error - update sklearn version

Use Cases:

- Google Translation i.e, Language Translation
- Voice Recognition (Voice to Text)
- Image inside text to be recognized

Doubts:

1. How we split the data into two parts because to train a model we have to learn entire data that was given.
2. What is necessary to learn One hot encoding though it is possible to do label encoding for nominal data.
3. How data cleaning is done on the target variable?

= Web API to extract data from the websites

= Parameter vs Hyperparameter

&

Parameter vs Argument.

↓ the variable defined within the paranthesis during function definition

↳ the value that is passed to a function when it is called.

= What is the need of Adjusted R square already we has R square?

+ R^2 gives same score even after introducing garbage feature
↳ not good metric (It always ↑ when you add new feature).

* Adj R^2 is better in this case

= Jeffrey Hinter - Father of ~~AI~~ ML/DL

↓

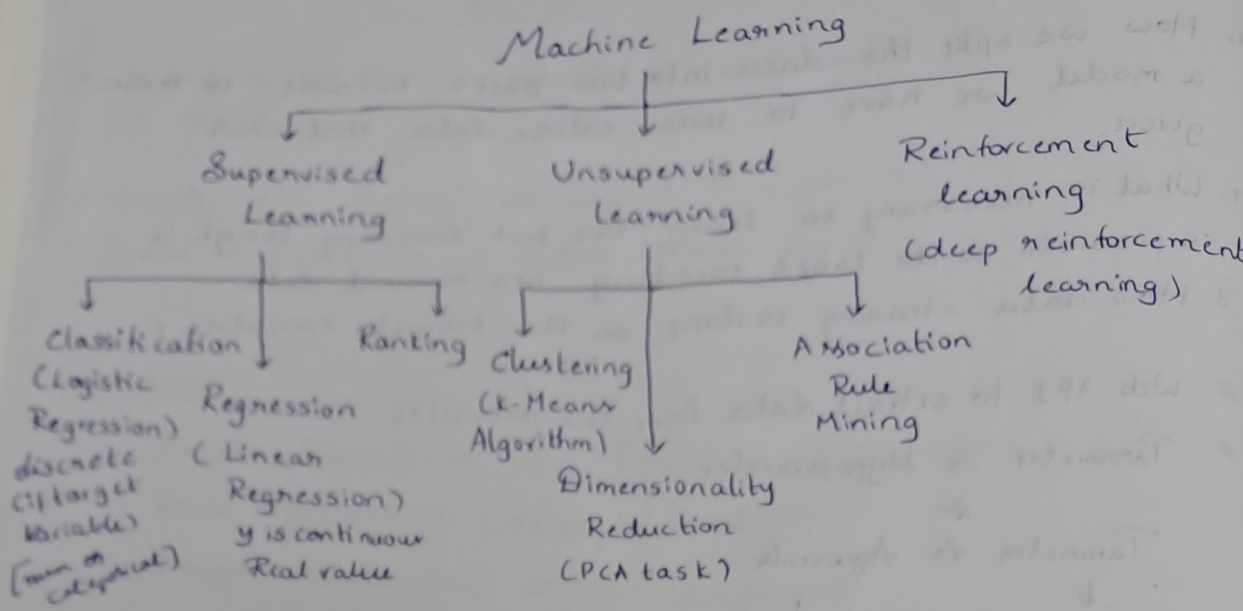
Geoffrey Hinton

- Father of AI - John McCarthy.

= A sparse vector is a vector that has a large number of zeroes so it takes unwanted space to store these zeroes.

Sparse	vs	dense
A sparse array contains mostly zeroes & few non zero entries		A dense array contains mostly non-zeroes.

dense vectors also named as embeddings.



Data + Algorithm → Model

Model → (predicted)

→ The value which is going to predict is called target variable (tv).

For Example:

In term deposit data the tv is status i.e., Approval or rejected and tv is categorical so we use Classification (logistic regression)

- In house pricing data the tv is price i.e., numerical so we use Regression (linear Regression)

Eg:

	Height	Weight	Gender
	155	50	Female
	180	90	Male
	175	85	Male
	145	50	Female
	185	90	Male
	150	55	Female
Queries Q_1	145	55	?
Q_2	180	85	?

Q) How do you train your computer to do this prediction?

Approach No - 1

Step 1 : Store the data

Step 2 : Perform a lookup

Approach No - 2

Male	Female
Height : [175, 185]	Height : [145, 155]
Weight : [85, 90]	Weight : [50, 55]

{ Th_H : 160
Th_W : 65 }

Code :

```
if Q['Height'] <= Th_H and Q['Weight'] <= Th_W :
    print('Female')
else :
    print('Male')
```

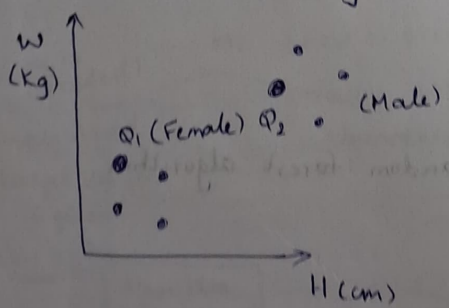
⇒ Here it is human learning / analysis because machine learning means machine itself create some conditions & predict the outcomes.

⇒ Features / attributes nothing but columns.

Algorithms :

* Distance based Algorithm

K-NN (K - Nearest Neighbour)

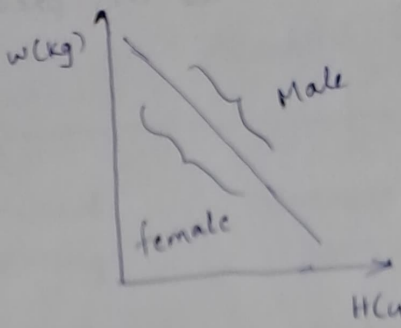


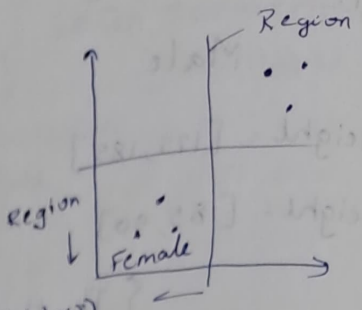
* Boundary based Algorithm

(Logistic Regression)

It is possible to draw a line that separates male & female.

* Support vector machines (Non linear separation)



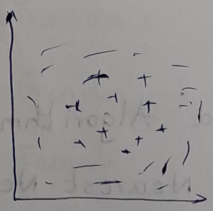


* Rule Based Algorithm
  (Decision tree Algorithm)
  Approach 2 - Automated

* Probabilistic Approach P(F | (Hei, Wei))
  Naive Bayes Algorithm P(M | (He, We))

* Ensemble Approach (collection of Algorithm)
  - Random Forest Algorithm
    - Stacking
    - BAGGing (Bootstrap Aggregating)
  - GDA Boost
  - GBDT
  - xGBoost
  - Voting Ensembles
  - Cascading

* Deep learning Based Algorithm ← Artificial Neural Networks
- If the data created in non linear structure then it uses deep learning



* In Ensemble Approach

Parrallel Models
- * Voting Ensemble
- * Stacking
- * BAGGing
  - ↳ Random forest algorithm

Sequence Models (consequative steps)
- * Cascading
- * Boosting
  - → ADA Boost
  - → GBDT
  - → XGBoost

Algorithms

Classification	Regression	Clustering	Dimensionality Reduction
• KNN	• KNN	• K Means (Hard clustering Algorithm)	
• Logistic	• linear	• kMeanstt (Hard clustering Algorithm)	
• System Vector class Support	• SV	• Heirarchial clustering (Hard clustering Algorithm)	
• DT class	• DT	• DBscan	
• NB class	• prob of linear Reg (MLE)	• GMM Algo (Soft clustering)	
• Voting Ensembles	• Voting Ensembles	• EM Algorithm (Soft clustering)	
• stacking	• Stacking	• Gaussian Mixture Model Expectation Maximization (Soft clustering)	
• Random forest (RF)	• RF		
• Cascading	: Cascading		
• ADA Boost, GBDT, XGBoost	• ADA Boost, GBDT, XGBoost		
• ANN class	• ANN Reg		

MLE - Maximum likelyhood Estimation

MAP - Maximum A-Posteriori

Type of Problems:

① Given the height, predict Weight

② Given SL, SW, PL and PW

③ Given an email, o/p if the email is Spam or not

④ Given a text review of product. Predict the rating

⇒ Height (x) → system / model → Weight ($\hat{y}$)

ML (how to predict which Algorithm is used)

- (data is labelled) → Supervised (if target variable is given) — if data consists of target variable
- (data is not labelled) → Unsupervised learning

* Data → Algorithm → Model

Actual process

Data (either numerical/ categorical/ something) → Preprocessing Technique → Data (Numerical) → Algorithm → Model

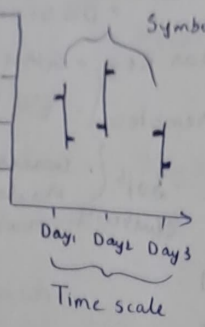
Clustering - grouping similar data or images.

∴ Understanding of Quiz Questions

Curse of dimensionality

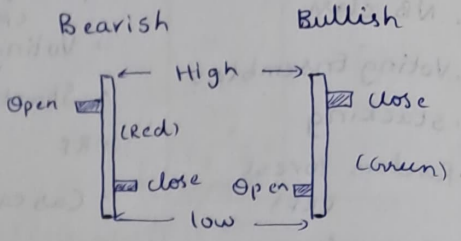
Edit url of linkedin

① Stock marketing : $40 $30 $20 $10



- Technical Analysis
  - uses historical price of stocks to predict future price
- Qualitative Analysis Price
  - external factors like scale company's profile, market situation, political & economic factors

Symbol Anatomy :



Here open means opening time pri
close - closing time price

② Spam data in emails are found. If it is labelled email then we use Supervised learning. So ∴ the tasks we use are
- Logistic Regression
- SV machines
- Random forest classifiers.

③ forecast water utilization for different input data

④ Pricing of house based on features like bedroom, sqft area, location (latitude, longitude), floors, condition of house.

- For all these data at first we must know the appropriate selection of features for our outcomes.

Curse of dimensionality :

It refers to the set of problems that arise when working with high dimensional data. The dimension of dataset corresponds to no of attributes / features that exist in dataset.
>100

Data drift : changes in the data due to ~~dimensionality~~ seasonality, changes in consumer preferences, addition of new products.

Prefect library

MLflow library

Data Preprocessing and [ML Process] Lifecycle:

Data → [Preprocessing] → X - transformed, y → [Algorithm] → Model
X, y — numerical

Data Cleaning:
- @ Missing value detection & Treatment
- ⓑ Outliers detection & Treatment

→ Categorical Columns (predefined & certain categories)

Eg: Crop type, Soil type, Gender, Nationality columns

- @ One hot encoding | Dummy Encoding
- ⓑ Label encoding

} help to convert categorical col to numerical col

Categorical
- Nominal (No order) — Eg: Gender: M, F, M, F, F
  * Use One hot encoding
- Ordinal (Order) — Eg: Grade: Ex, Avg, Good, Ex, Avg
  * Label Encoding

→ Text data (It should be any length & of our choice)

ML
- @ BOW (Bag of Word)
- ⓑ TF - IDF (Term frequency - Inverse Document frequency)

DL
- ⓒ Word 2 vec (Google) (2013)
- ⓓ Fast text
- ⓔ Glove (Global vector Encoding)
- 2017 ⓕ ELMo (Embeddings from language Model)
- 2018 ⓖ BERT (Bidirectional Encoder Representation from Transformers)

SORT

→ Image Data

- @ Flattening } ML
- ⓑ SIFT (Scale Invarient future Transformation)
- ⓒ CNN [extract features

→ Numerical Data
- (a) Normalization
- (b) Standardization

→ Audio Data
- (a) Mel-Scaled filter bank
- (b) Mel-frequency Cepstrum Coefficiency (MFCC)
- (c) Fourier Transformations

① Identify target variables
② Identify features (x - collection of input features)
③ Determine type (supervised / unsupervised)
④ Determine task → $y \in \{\text{'spam'}, \text{'ham'}\}$ → classification
(↑ belongs to; ↑ Discrete values)
$y \in \mathbb{R}$ (Real no/value → Regression in 1-D)

Example:

X	Y
Col-1	Col-2
1	1
3	9
0	0
2	4
5	25

class vs label
in a column spam & ham are diff classes.
2 or more

quadratic means square of one variable

Col-1 = 5 → Model → 25 for Col_2

Col_2 = Col_1 * Col_1

Multi label classification: (multi output classes)

Movie Summary → Comedy: T, Horror: F, Drama: T ...

* Multiple target variables. If y has more than 1 target variable.

Binary Classification:

→ $y \in \{\text{'spam'}, \text{'ham'}\}$ (1, 0). If y has only two categories

Multiclass Classification:

→ $y \in \{\text{'setosa'}, \text{'versicolor'}, \text{'virginica'}\}$. If y has more than 2 categories.

* Data preprocessing is also done it to the new query point

model development

x_{qv} → [Data Preprocessing] → x_{qv} - transformed → [Model] → $\hat{y}_{qv}$

Email: Create a spam detection for email data

email-text → 🧍 → spam/ham

Step:

① target → Email-label ∈ {'spam', 'ham'}

Data →

email	label
- - - -	spam/ham
- - - -	
- - - -	

Type → Super learning algo can be applied

Task → Classification

Dataset X (text), y (label) → [data preprocessing] → x - transformed by y (num)

Q) How do you use this model?

email → Model → label

email x_{qv} (text) → Data Preprocessing → x_{qv} - transformed (num) → Model → $\hat{y}$

* SME - ~~Support~~ Subject matter expert

* feature Engineering - deriving new features from existed

⇒

Alexa or Siri, Google Assistant ← Speech

Automatic Speech Recognition

[ASR → Text → NLP Techniques]

[data preprocessing → Data (num) → Model] → Text (outcome)

Alexa, Youtube subtitles.

Data → data Cleaning → Data transformation → Data transformed

Num
Cat } → ⓐ Missing values ⓐ Num → Normalization/ Standardization
ⓑ Outliers
↳ Remove noise & abnormalities from data
ⓑ Cate → OHE LE
↳ Non numerical to numerical data

Text → ⓐ Special char Removal
ⓑ lower case Conversion
ⓒ Stopword Removal
ⓓ Stemming / lemmatization
All are required

ⓐ BOW ⓔ ELMO
ⓑ TF-IDF ⓕ BERT
ⓒ W2V
ⓓ fast-text
ⓔ Glove

Vectorization means Convert Non numerical to numerical

Why these techniques? (for categorical features)
To convert categorical into numerical features

Why? (for numerical columns)
Rescaling because for each feature it has some own scale which is different for everyone / every feature
Eg: Height, Sqft, Weight, age. So that all numerical features follow same scale.

⇒ x - transformed, y → Algorithm → Model

Regression
$y \in \mathbb{R}$

Classification
$y \in \{1, 0\}$ → binary classification
$y \in \{0, 1, 2, 3\}$ - Multi class "

Data Transformation
- Numerical → Normalization
→ Standardization

① Normalization (Min-max scaling) → [0 - 1]

Eg:

H (→ [140-190] cm)	W (→ [40-90] kg)	G

→

H' (→ [0-1])	W' (→ [0-1])	G

for H (col): $\frac{Obs_i - min}{max - min}$

for W (col): $\frac{Obs_{i(W)} - min}{max - min}$

for H (col): $\frac{140 - 140}{190 - 140} = 0$ (min)

$\frac{190 - 140}{190 - 140} = 1$ (max)

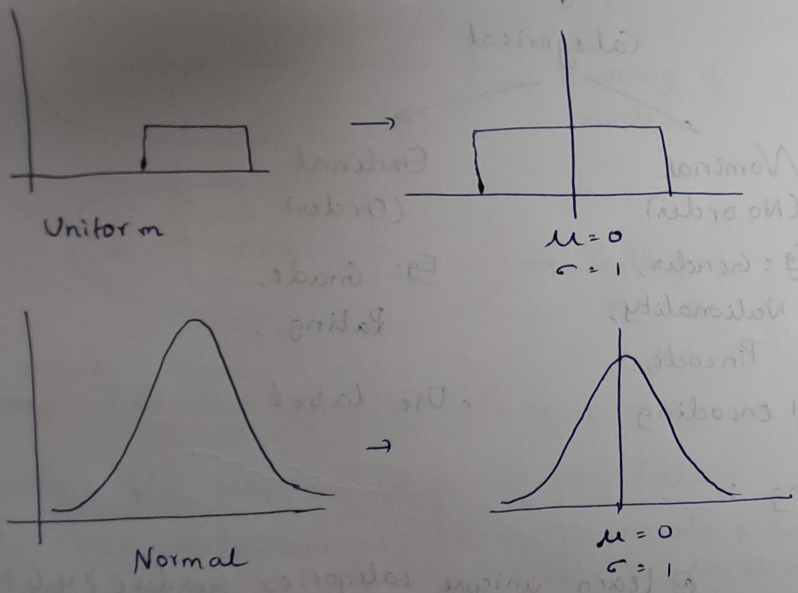
* It brings all the values to scale [0 - 1]

② Standardization (z-transformation)

→ $\mu = 0$

→ $\sigma = 1$

Robust Scaler



Z score → how many standard deviation away is the observation from the μ

or

how far away is observation from mean w.r.t std deviation

$$2\text{-score} = \frac{Obs_i - \mu}{\sigma}$$

* dipricated - not applicable for future versions just throw errors.

* DNS - Domain name server
* Spread - how far data points away from mean

σ (for sample)

$$s = \sqrt{\frac{\Sigma (x - \bar{x})^2}{n-1}}$$

σ (for population)

$$\sigma = \sqrt{\frac{\Sigma (x - \mu)^2}{N}}$$

→ $H \xrightarrow{\text{Standardization}} H' = \frac{obs_i - \mu_H}{\sigma_H}$

- Here, outliers are going to corrupt / effect the data

→ Conduct same technique either standardization / Normalization for all features in a dataset

⟹ Categorical Columns

- Categorical
  - Nominal (No order)
    - Eg: Gender, Nationality, Pincode, ...
    - Use OH encoding
  - Ordinal (Order)
    - Eg: Grade, Rating, ...
    - Use label

One Hot Encoding :-

① Gender

Gender
Male
Male
female
Male
female

→

ⓐ learn unique categories Gender ∈ {Male, female}

ⓑ Apply Onehot Encoding (drop the first column) because if we dlt Gender_M then Encoding for male - 0, for female - 1 (unique right)

① Curse of dimensionality ↓

② Multi collinearity blw features

Gender_M	Gender_F
1	0
1	0
0	1
1	0
0	1

Male → 1 0
female → 0 1

Q: What is encoding for Gender = Male?

Ans 1

Gender

Male
female
Male
Male
female

→ ⓐ Identify unique categories
Gender ∈ {'Male', 'female'}
ⓑ label categories
{'Male' : 1
'female' : 2}
ⓒ Apply label Encoding

→

1
2
1
1
2

Grades

Excellent
Good
Avg
Good
Avg
Excellent

ⓐ Unique Grades ∈ {'Excellent', 'Good', 'Avg'}
ⓑ {'Excellent' : 3,
'Avg' : 1,
'Good' : 2}
3 > 2 > 1
ⓒ Apply

→

3
2
1
2
1
3

Algorithm → Model → Training of model

x_q → model → $\hat{y}_q$ → Testing of model

Machine Learning Project Life Cycle :

① Define the problem & Check the data

i.e., Price prediction → identify all inputs & target variables
x → sqft, age, locality
y → Price

Spam prediction
x → email
y → label (spam, ham)

Task

Regression

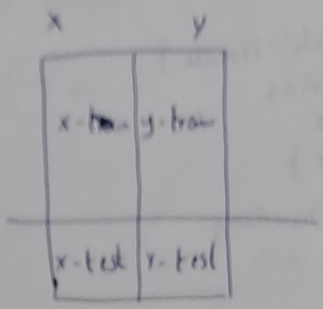
Algorithm	Evaluation matrix
KNN	1. Mean squared error
linear	2. " absolute "
SV	3. Root mean squared error
DT	4. R^2 (coefficient of determinant)
	5. Adjusted R^2

Classification

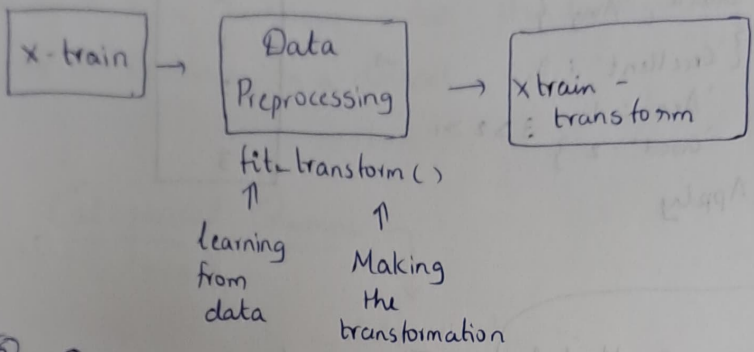
Algorithm	Evaluation Matrix
KNN class	1. Accuracy
logistic	2. Confusion matrix
SV	3. Precision & Recall
DT	4. F1-score
	5. ROC-AUC Curve
	6. Log loss.

② Identify input & Output

③ Split the data for training & testing

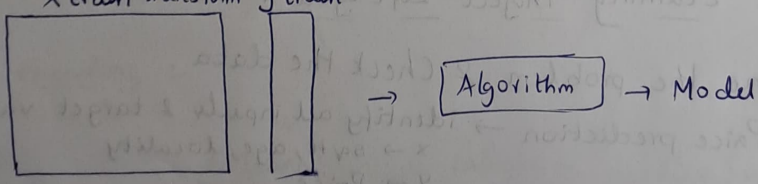


④ Data Preprocessing on x-axis train

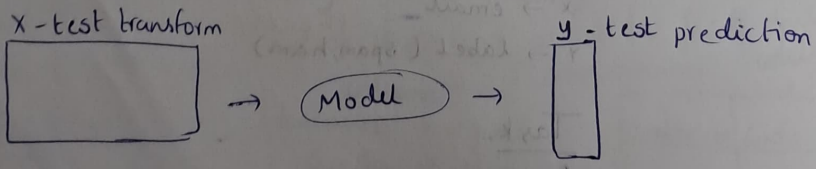


⑤ Data preprocessing on x-test

⑥ Apply ML Algorithm to learn from training data



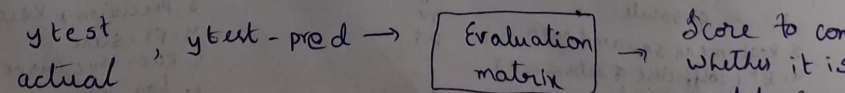
⑦ Testing your model on unseen data



Here we already have y-test so just compare it with model y-test predictions.

ⓐ Predict on unseen data

ⓑ Evaluation of your prediction



* Version should be 1.1 or above

Coding part :

» print (sklearn.__version__) → class

» from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier() [object creation]

model.fit (x_train_transformed, y_train)

Data

y_test_pred = model.predict(x_test transformed)

predictions

from sklearn.metrics import accuracy_score

print (accuracy_score (y_test, y_test_pred))

ⓐ for splitting the data into train and test

» import sklearn (!pip install -U scikit-learn)

from sklearn.model_selection import train_test_split → here data is randomly shuffled.

x_train, x_test, y_train, y_test = train_test_split (x_data_name, y_data_name, train_size = 0.8/0.7, random_state = 100)

ⓑ Separating Categorical and Numerical Columns

categorical :

» x_train_cat = x_train.select_dtypes (include = ['object'])

numerical :

» x_train_num = x_train.select_dtypes (include = ['int64', 'float64'])

ⓒ Scaling the numerical features

» from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

x_train_num_rescaled = pd.DataFrame (scaler.fit_transform (x_train_num), columns = x_train_num.columns, index = x_train_num.index)

x_train_num_rescaled

≠ Based on the Mean Absolute error which value is less means best model i.e, Random forest Regression

≠ Maximum time for training took by Random forest Regression Algorithm.

Another technique to rescale the numerical features i.e.,

④ Normalization

» from sklearn.preprocessing import Normalizer

(or)

» from sklearn.preprocessing import MinMaxScaler

---

fit_transform() means here fit represents learner from the data i.e., μ & σ for every feature for transforming using Standardization

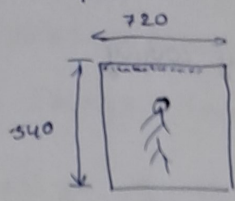
⑤ Applying OneHotEncoding for Categorical Columns

» from sklearn.preprocessing import OneHotEncoder

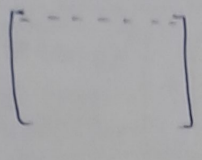
encoder_ = OneHotEncoder(drop='first', min_frequency=3000, (merged i.e., <3000)
sparse=False)

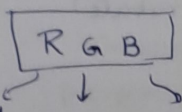
Image Preprocessing and Computer vision:

import pil → Python imaging library



Here along W → 720 pixels
along H → 540 pixels
1 pixel = 1 Numerical value

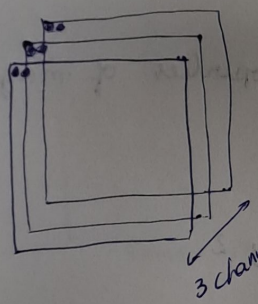




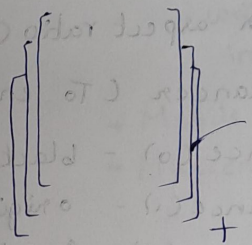
Red (0-255) Green (0-255) Blue (0-255) intensities

White – R 255, G 255, B 255

Black – R 0, G 0, B 0

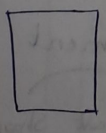


Matrix



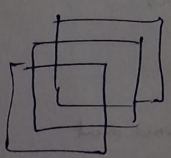
→ Every colour is formed with the intensities of R G B colours

* Aspect ratio = $\frac{W}{h}$ → No of pixels in width
→ No. of pixels in height



→ Black & white (Grey scale) → No need of multiple channels

→ No. of pixels = 64
Aspect ratio = 1



→ Coloured image (e.g. RGB) → No. of pixels = 64 * 3

» from matplotlib import image
(read the image & displays the o/p. of image)

* The image is stored in python as numpy array

Image Manipulation with PIL:

① Rotate
② Crop
③ Flipping — flip left to right, flip top to bottom, flip transpose (also done with rotate)
④ Resize

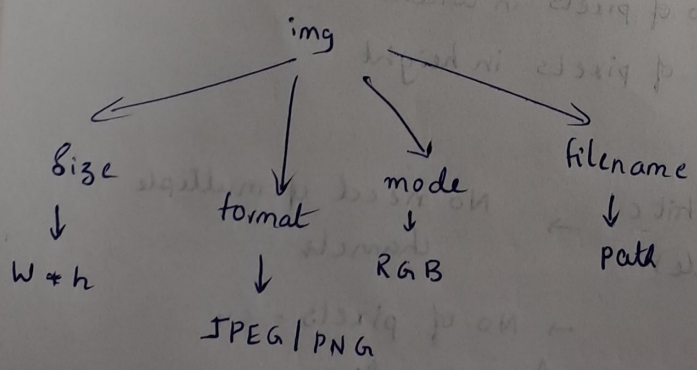
* from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont

→ Resizing is not automatic but thumbnail itself takes the size as per aspect ratio (W/H)

→ Image Enhancer (To enhance / ↑ the properties of image)
enhance(0) - black & white
enhance(1) - original
enhance(2) - ↑ intensity of colour by 2
① Colon
② Contrast

*img.resize((w, h))
*img.thumbnail((w, h))
↳ forcefully keeps the original images Aspect ratio.

Properties of image



* Aspect ratio = img.size[0] / img.size[1]

→ ImageEnhancement
Color, Bright ness, Contrast, Sharpness

→ Image Filters
→ Blur
→ Smooth
→ Edge Enhancement
→ Emboss
→

Reducing the Dimensionality :

1. Reducing the Channel → Convert to Grayscale

```
plt.imshow(img.convert('1'))
gray_image = image.convert('1')
```

2. Resize of the image

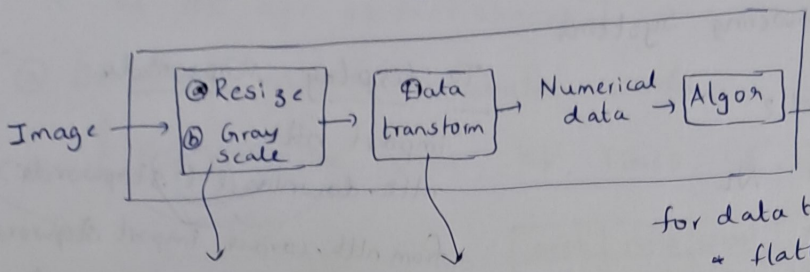


Image pre-processing steps

- reduce dimensionality
- reduce the noise (DL)

```
np.array(img)
```

for data transforming (image data) to numerical

- flatten
- CNN
- SIFT (old technique) + flattening
- AlexNet (CNN Network)
- GoogleNet (CNN Network) (Inception module)
- ResNet (CNN)
- Vision Transformer.
- snd file (to write what is spoken in the video)
- Mobile Net (CNN)
- Efficient Net (CNN)

* [ ] Row vector

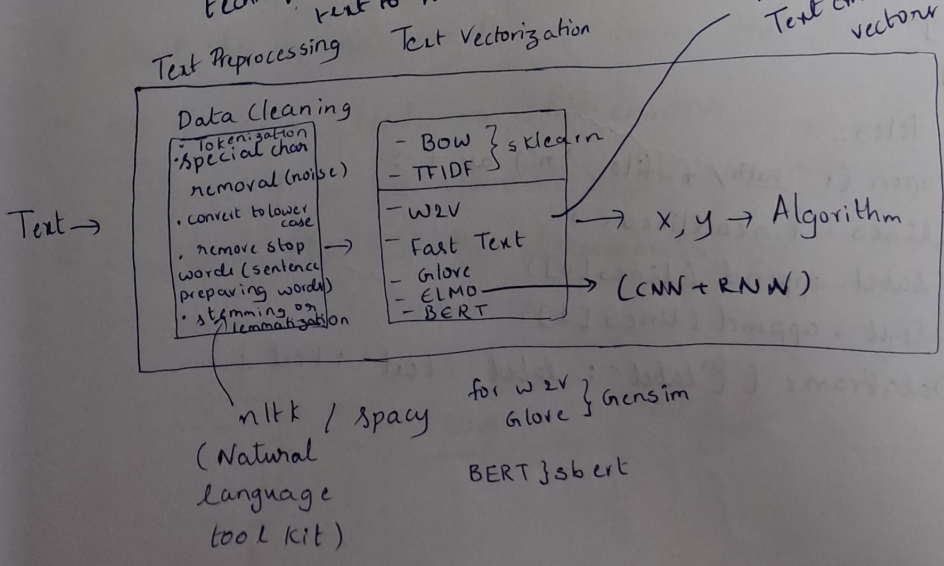
* [ ] Column vector

→ To clean the outliers we make use of log transformation

Text data :→ techniques to transform text to numerical

→ RNN / LSTM / GRU

→ Transformers (SOTA)



nltk / spacy (Natural language tool kit)

for W2V Glove } Gensim

BERT } sbert

1. Spell checker, keyword search etc
2. Sentiment analysis, Spam classification
3. Machine Translation → language translation
4. Chatbots / Dialog Systems
5. Questions Answering Systems

Why NLP is hard?

- Ambiguity in NL
  same meaning
  different names
- Complexity of representation

To display stopwords:

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop = set(stopwords.words('english'))
print(stop)
```

- S = 'Hii Sri Kanya'
  S.split()
  - ['Hii', 'Sri', 'Kanya'] } word level token
  - ['Hii Sri Kanya'] } sentence level token
  - ['H', 'i', 'i', ' ', 's'...] } character level token

Quiz:

```
import os
print(os.listdir('text'))
files_ = os.listdir(r'path')
label = []
text = []
for file in files_:
    with open(r'path/' + file, 'r') as f:
        lines = f.readlines()
        label.append(lines[0])
        text.append(lines[1])
df = pd.DataFrame({'label': label, 'text': text})
```

path → foldername

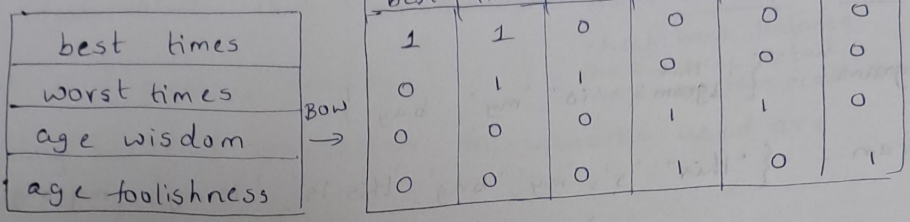
Vectorization:

Bag of Word:

['it was the Best of Times $', 'It Was The worst of times.', 'IT was a the age of wisdom', 'it was thE age of foolishness'] — After data cleaning → ['best times', 'worst times', 'age wisdom', 'age foolishness']

ⓐ Learn the unique vocabulary words

{'it', 'was', 'the', 'Best', 'of', Times', '$', 'It', ...}



best	times	worst	age	wisdom	foolish
1	1	0	0	0	0
0	1	1	0	0	0
0	0	0	1	1	0
0	0	0	1	0	1

* corpus: collection of Documents

~~Lotte~~ Document = A single data

D – Document
T – Term
M – Matrix

ⓑ Arrange in alphabetical order i.e.,

age best foolish times wisdom worst

No. of columns = No. of unique vocabulary words in Corpus

Most common value = 0

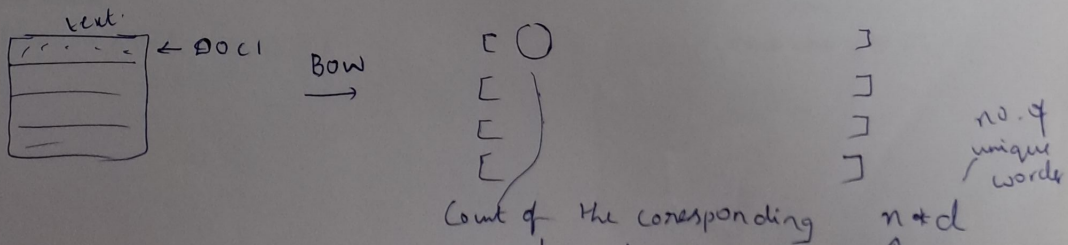
During the process of BOW:

ⓐ Learn all the unique words from the Corpus

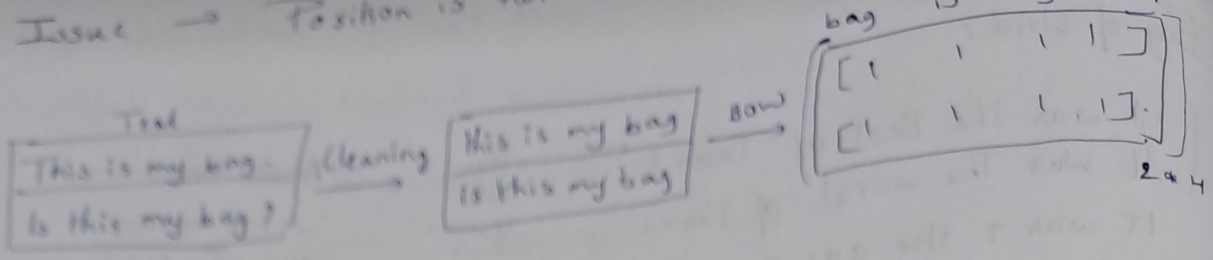
ⓑ Sort the unique words & form the features

* No. of col's / features / dimensions
  - No. of unique words in the corpus.

ⓒ Transformation → All the documents gets transformed to numerical vectors.



Issue → Position is not Preserved



Try to somehow include the positional information

n-gram approach:

1 gram - { 'this' (1gram), 'is', 'my', 'bag' }

2 gram - { 'this', 'is', 'my', 'bag', 'this is', 'is my', 'my bag', 'is this', 'this my' }

Bag of words

from sklearn.feature_extraction.text import CountVectorizer

Steps:

'csr' → Compressed sparsed raw format

① Identify target variable

Classification Regression

② Split the data into train & test

③ Apply text cleaning & transform on the train data

④ Apply cleaning on test data.

⑤ Build the model using train data

x_train_dtm ⟶ Model →

⑥ Use the model for pred on unseen data

x_test_dtm → Model → y_test_pred

⑦ Evaluate the performance of Model

In Chatbots (RASA)

① Intent Classification is a Classification problem. [intention]

intent ∈ {'greet', 'good bye', ...}, check_bank_balance, check_course_detail, check_weather, call_somebody

For creating backend application, frameworkr used are

- Flask
- Django
- Fast API

} to create backend server. → Python frameworkr

Server responds to the request given by any person

CDN - Content delivery network

- Spring
- Spring boot

} - Java frameworkr

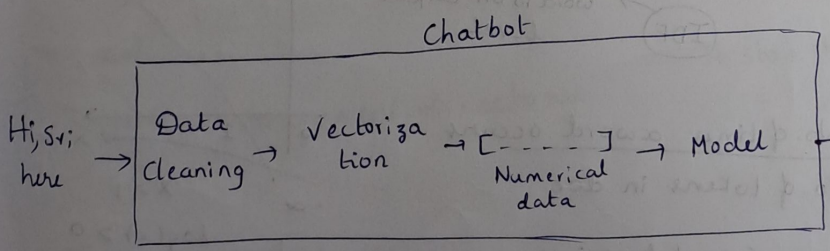
- .Net } CSharp framework

frontend
- VueJS
- ReactJS
- Angular

backend
- NodeJS

} - Java Script

② Entity Extraction



Intent classification

I/p : Hi, Sri here

O/p : Hi user
How are you ?

Vs

Intent classification + Entity extraction

I/p : Hi, Sri here

O/p : Hi Sri
How are you?

Spam/ham detection:

1. Data Visualization [Countplot]
   To get count of values in % we use Normalize = True.

- tqdm is a library used for creating progress bars.
- wordcloud to represent the words in a visual pattern larger words represents the high frequently occured words.

Check 31 cell in this python file i.e.,

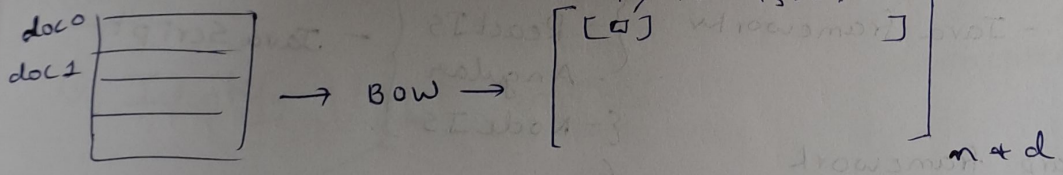
» spam_df = X_train.loc[y_train == 'spam', :]

Preprocessing Technique:

* TF-IDF (Term Frequency - Inverse Document frequency)
  - Tell u the importance of words.

BOW Technique

Clean text corpus



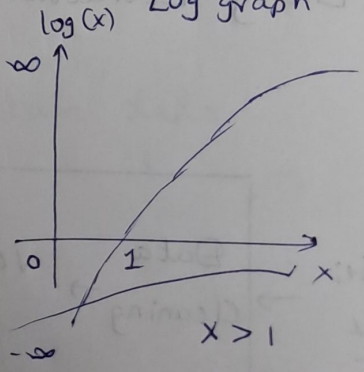
Coming to TF-IDF

- TF → how important the word is on doc level
- IDF → how important this word is on corpus level

* TF(w, doc) = $\frac{\text{No. of times a word occurs}}{\text{No. of tokens in doc}}$

IDF(w, corpus) = $\log_n \left[\frac{\text{No. of docs in corpus}}{\text{No. of doc which contains words}}\right]$

Log graph



$x > 1$

$\log(x) > 0$

(Python code)

1. python installed

The version of python should be '3.9'.

2. Virtual Environment using python

cmd → python -m venv name-of-env ← environment

3. Activate the environment

cmd → name-of-env\Scripts\activate → for linux/mac we use bin scripts/activate.

4. Install Rasa { Rasa is a framework that helps in chatbot development }

cmd → pip install rasa

Good server gives data in less time

5. Create a Rasa Project

cmd → rasa init

↵

command prompt

Some rasa Commands :

- Initialize the project
  - rasa init
- To interact with the chatbot
  - rasa shell
- To train the chatbot
  - rasa train

To jump back to previous directory

- cd ..

To jump into current directory

- cd .

* To know python version

python -V

python --version

Create new folder

↓

click cmd

To change directory

Click cmd

↓

cd Desktop

↓

Enter

To check whether the rasa is installed

↓ pip list

To stop the server

- /stop

domain.yml

- define the intents
- contains utterances

data/nlu.yml

Contains intents

Eg:

data/stories.yml

contains flow of dialogues

* .ipynb → Only opens on jupyter notebook (interactive python notebook)
  .py → Opens on notepad i.e., text file
  .yml → "

→ intent → 'greet'
  action → 'utter-greet'
  intent → 'mood-unhappy'
  action → 'utter_cheer_up'

story → defines dialogue flow

To train a chatbot :

- ~~Ident~~ify the intent

data/nlu.yml :

```
nlu :
- intent : greet
  exampler :
  - hey
  - hello
- intent : goodbye
  exampler :
  - cu
  - goodbye
  - ceeyou later
  - bye
  - Goodnight
```

data/stories.yml :

```
stories :
- story : happy path
  steps :
  - intent : greet
  - action : utter greet
  - intent : mood great
  - action : utter happy
```

domain.yml

```
For Eg :
intents :
  - greet
  - goodbye
responses :
  utter greet :
  - text : Hey ! How are y
  utter goodbye :
  - text : ' Bye '
  utter happy :
  - text : ' Great, carr
```

- Single Shot detection
- Zero Shot detection.