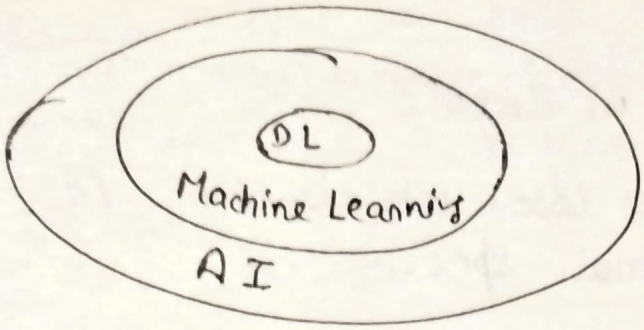


AI - Pattern recognition

ML - we statistical data results to find patterns. we have to define features

DL - algorithms are inspired by neuron. no need to define features, machine will itself do it.
accuracy increases with no. of layers.
performance increases with more data.
Image, text related.
used for complex problem.

Types of ML (on the basis of amount of supervision needed)

- Supervised
  - Regression
  - Classification
- Unsupervised
  - Clustering
  - Dimensionality reduction
  - Anamoly detection
  - Association
- Semi supervised
- Reinforcement

Supervised learning - I/P and o/p both in data. algorithm learns pattern for data and gives prediction on new data.

Regression - Numerical data
Classification - categorical data
> of o/p

- Unsupervised Learning
  Only i/p in data.

  Clustering - divided data into clusters in n dimensional space.

  dimensionality reduction -
  input colums are so many.
  reduce no. of columns.

  Anomaly detection - Remove outliers.

  Association - draw conclusions between different columns.

- Semi-supervised Learning
  Labelling is difficult.
  So we label for few data
  algorithm will label rest with its own.
  ex- Google Photos will label photos if you label once the person.

- Reinforcement Learning
  no data will be given.
  Model will do mistakes, learn and improve.
  ex- Self driving Car
  Go Game

Types on basis of how ML model trains on production

- Batch / offline ML
  whole data is used to train model at once.
  in offline system.
  Once when it is trained, test model and put on server.
  problem- model won't get updated on data growth.
  u need to train model periodically.

disadv - Lots of data
Hardware limitation
if there isn't internet connectivity with ur model
Availability

Online ML

dynamically model gets trained on server with data.
done incrementally.
feed data in batches sequentially.

Ex - chatbot — more you use model, it increases its performance.

when to use?

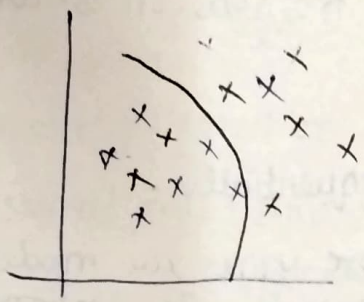
1. where there is a concept drift
   └stock exchange, e-commerce site
2. Cost effective.
3. Faster solution
4. learning rate - how frequently you want your model learn.
5. Out of core learning - when data is so big that it can't fit into memory.

disadv - tricky to use
Risky to ensure system is safe.

Instance vs Model Based Learning

Instance based - no concept of learning
Just based on data, algorithm predicts for new point. KNN

Model based -



decision function we get which is drawn from data.

Linear regression

Conventional ML

- Store model in suitable form.
- predict for unseen scoring instance using model.
- Can throw away input data after model training.
- Storing models generally requires less storage.

Instance based ML

There is no model to store.

Predict for unseen scoring instance using training data directly.

input data must be kept since each query uses part or full set of training.

Storing training data generally requires more storage.

Challenges

1- Data Collection
2- Insufficient data
huge amount of data - algorithm doesn't matter much.

4- Poor quality data
   - not correct format
   - outliers
   - null values
5- Irrelevant features
6- Overfitting
7- Underfitting
8- Software prediction
   to integrate it with software
9- Offline learning/ deployment
10- Cost Involved

Tensors

↓
Basic ds of all leading libraries.
Container to store data i.e. numbers

0D Tensor/Scalar - (2) (3)

.ndim - to find dimension

1D Tensor/Vector [1 2 3 4]
↑4D

2D Tensor/Matrices $\begin{bmatrix} [1,2,3] \\ [4,5,1] \\ [6,7,8] \end{bmatrix}$

```
mat = np.array([[1,2,3],[4,5,6],[7,8,9]])
```

ND Tensors

Rank - no. of dimensions = no. of axis

Shape (,)
↓ elements in row axis
↳ elements in column axis

Ex of 3D Tensors

NLP

Timeseries data

	Highest	Lowest
Day 1		
Day 2	—	—
⋮	—	—
⋮	—	—

[365, 2)

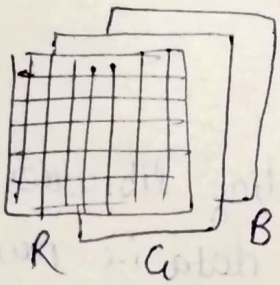
for 10 years

(10, 365, 2)

↳ time axis

Ex of 4D Tensors

Images



(3, 1200, 800)

for 1 image

50 color images

(50, 3, 1200, 800)

4D Tensor

Ex of 5D Tensors

Videos

↳ collection of frames

60 sec video

↳ 30 fps

↳ 480p → 480 × 720 (3 chann

[4, 1800, 480, 720, 3]

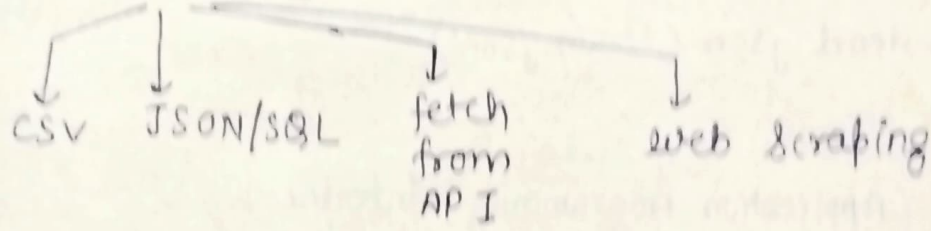
4 videos

no of images in 1 video

df.iloc [: , 1 :]

all rows
1 column onwards

Data Gathering



CSV (comma separated values)
TSV (Tab separated values)

pd.read_csv(' ', sep = '\t', names = [' ', ' ', ' '])

column name

index_col = ' '
override default index column

header = 1
when column names are treated as row.

usecols = [' ', ' ', ' ']

skiprows = [0,2]

nrows = 100 — importing only n rows

error_bad_lines = False (Parser error)

dtypes parameter

dtype = {'target': int}

Handling dates

parse_dates = ['date']

Convertors

convertors = {'team1': rename}
function

na_values parameter

na_values = ['-',]

loading a huge dataset in chunks

- JSON / SQL

API → JSON

SQL → database

```
pd.read_json('train.json')
```

SQl

- API (Application Programming Interface)
  - ↳ data pipelines
    - ↳ comm. b/w two softwares

```
import requests
response = requests.get('                ')
pd.DataFrame(response.json()['results']).head()

df.to_csv('   ')
```

- Web Scraping

```
import requests
from bs4 import BeautifulSoup
headers={                                    }
webpage = requests.get('                    ',headers).text
                                        : (whole html from site)
Soup = BeautifulSoup(webpage, 'lxml')
print (Soup.prettify())

soup.find_all('h1')[0].text
soup.find_all('p', class='rating')
```

Understanding data

1. How big is data? df.shape
2. How does data look like? df.head() or df.sample(5)
3. data type of cols? df.info()
4. Are there any missing values? df.isnull().sum()

5. How does data look mathematically ?

```
df.describe()
```

6. Are there duplicate values ?

```
df.duplicated().sum()
```

7. How is the correlation b/w cols ?

```
df.corr()['  ']
```

Exploratory data Analysis (EDA)

Univariate analysis

analysis of single column individually

Categorical Data

```
import seaborn as sns
```

a) Countplot

```
sns.countplot(df['Survived'])
```

or

```
df['Survived'].value_counts().plot(kind='bar')
```

b) Piechart

```
kind='pie'
autopct='%.2f'
```

Numerical data

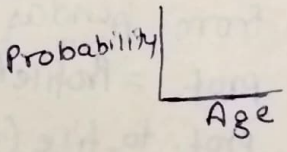
```
import matplotlib.pyplot as plt
```

a) Histogram

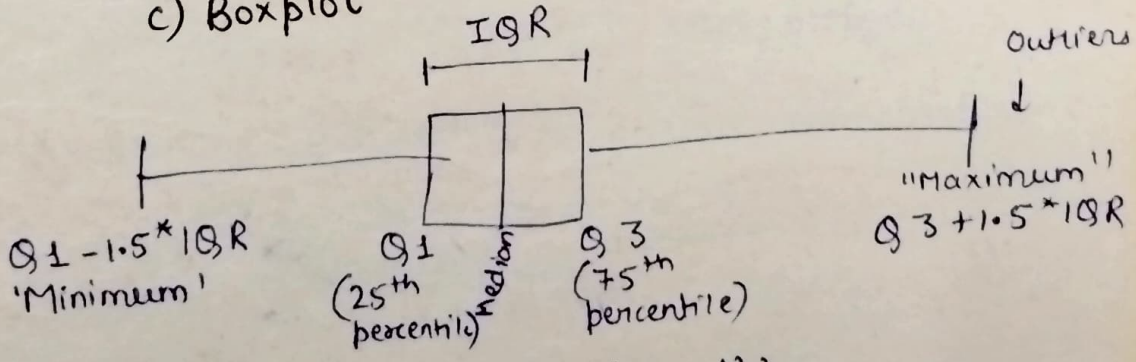
```
plt.hist(df['Age'])
```

b) Distplot

```
sns.distplot(df['Age'])
```



c) Boxplot




```
sns.boxplot(df['fare'])
```

Bivariate / Multivariate Analysis

a) Scatterplot (Numerical - Numerical)

```
sns.scatterplot(tips['totalbill'], tips['tip'],
                hue = df['sex']
                style =
                size =
```

b) Barplot (Numerical - Categorical)

```
sns.barplot(                )
```

c) Boxplot (Numerical - Categorical)

```
sns.boxplot(                )
```

d) Distplot (Num. - Cate.) pdf

e) Heatmap (Cate. - Cate)

```
sns.heatmap(pd.crosstab(titanic['Pclass'], titanic['Survived
```

Similar data place near

f) Clustermap

```
sns.clustermap(
```

g) Pairplot

```
sns.pairplot(iris, hue = 'species')
```

h) Lineplot (Num. - Num.)

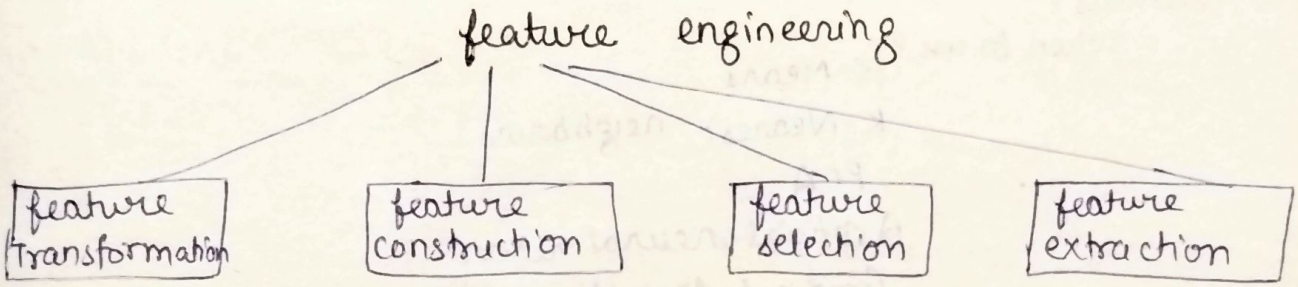
X-axis - timebased

Pandas Profiler

```
!pip install pandas-profiling
from pandas-profiling import ProfileReport
prof = ProfileReport(df)
prof.to_file(output_file = 'output.html')
```

Feature Engineering

extract features from raw data. These features can be used to improve performance of ML algorithms.



- Missing value Imputation
- Handling categorical data
- Outlier detection
- feature Scaling

1. ~~Missing values Imputation~~

Feature Scaling
- Standardization
- Normalisation

Standardization

Standardize the independent features present in data in fixed range.

why do we need feature scaling

KNN - needs to find euclidean distance between points.

(age, salary) - salary will dominate.

KNN won't work properly.

Standardization - also called as Z-score normalisation.

$$x' = \frac{x_i - \bar{x}}{\sigma}$$

after scaling whole column:-

$\mu = 0$ $\sigma = 1$

```
X_train_scaled = scalar.transform(X_train)
X_test_scaled  =      "     "   (X_test)
```

Standardization doesn't have any impact on outliers.

When to use?
- K-Means
- K-Nearest neighbours
- PCA
- Artificial neural network
- Gradient descent

Normalization

goal is to change the values of numeric column in dataset to use a common scale, without distorting differences in the ranges of values or losing information.

- MinmaxScaling
- Mean normalisation
- Max absolute scaling
- Robust Scaling

Minmax Scaling

weight
130
67
81
61
32
!
!
100

Normalize →

$$X_i = \frac{X_i - X_{min}}{X_{max} - X_{min}}$$

$[0, 1]$

```
from sklearn.preprocessing import MinMaxScalar
scalar = MinMaxScalar()
scalar.fit(x_train)
x_train_scaled = scalar.transform(x_train)
X_test_scaled = scalar.transform(x_test)
```

Mean normalization

$$X_i' = \frac{X_i - X_{mean}}{X_{max} - X_{min}} \qquad [-1 \text{ to } 1]$$

Sklearn doesn't support

Max Abs Scaling

$$X_i' = \frac{X_i}{|X_{max}|}$$

MaxAbsScalar

↳ Sparse data
↳ 0's

Robust Scaling

$$X_i' = \frac{X_i - X_{median}}{IQR \{75^{th} per - 25^{th} per\}}$$

RobustScalar → Robust to outlier

Normalization vs standardization

Is feature scaling required?

Mostly standardization.

MinMax
(when min and Max known)
image → CNN [0 - 255]

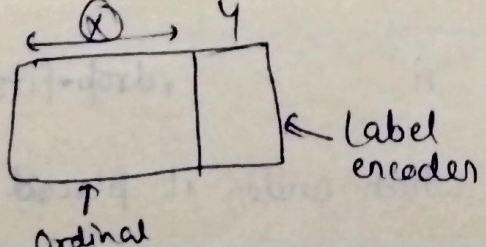
- Encoding categorical data

Nominal — Ordinal (relationship b/w values)

State Branch

Ordinal encoding

One hot encoding (Nominal)



Education

HS	0
UG	1
PG	2
PG	2
UG	1
HS	0

PG > UG > HS

PG → 2

UG → 1

HS → 0

```
from sklearn.preprocessing import OrdinalEncoder
oe = OrdinalEncoder(categories = [['Poor', 'Average', 'Good'],
                                  ['School', 'UG', 'PG']])

oe.fit(X_train)
X_train = oe.transform(X_train)
```

Labelencoder

```
le = LabelEncoder()
```

One Hot Encoding

1 column for each category

n categories → n-1 columns (to avoid multicollinearity)

OHE using most frequent variables

will keep frequent categories and for rest will keep only 1 column.

using Pandas

```
pd.get_dummies(df, columns = ['fuel', 'owner'])
```

k-1 OHE

```
"                "                ,drop_first = True)
```

Panda doesn't remember in which order it placed first time

using Sklearn

```
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder( ) [drop='first' for k→]
X_train_new=ohe.fit_transform (X_train[['fuel','owners']]).toarray()
np.hstack ((X_train[['brand','km']].values, X_train_new))
```

We have to perform OHE on categorical columns separately then we can append that.

toarray( ) → sparse = false in OHE
dtype = np.int32

OHE with top categories

```
counts = df['brand'].value_counts( )
threshold = 100
repl = counts[counts <= threshold].index
pd.get_dummies (df['brand'].replace (repl, 'uncommon'))
```

Column Transformer in Sklearn

In feature transformation, every column can have diff. problem, we need to transform them separately and then append. Solution of this is column transformer.

```
from sklearn.compose import ColumnTransformer
transformer = ColumnTransformer (transformers=[ ],
                                 remainder='passthrough')
```

in that bracket [ ] ('tnf1', SimpleImputer( ), ['fever']),
('tnf2', OrdinalEncoder (categories=[[ ]],
['cough']))
('tnf3', OneHotEncoder(sparse=false), ['gender'])

I st - name of transformer
II nd - which transformation
III - on what column

Scikit Learn Pipelines

Pipelines chains together multiple steps so that the output of each step is used as input to the next step.

Pipelines makes it easy to apply the same preprocessing to train and test!

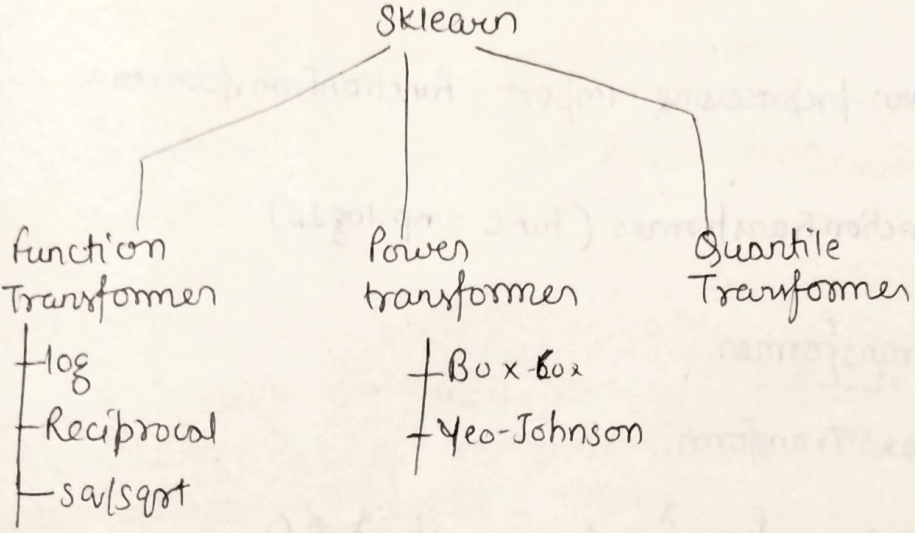
```
from sklearn.pipeline import Pipeline, make_pipeline

pipe = Pipeline([
        ('trf1', trf1),
        ('trf2', trf2),
        ('trf3', trf3),
        ('trf4', trf4),
        ('trf5', trf5)
    ])

or

pipe = make_pipeline(trf1, trf2, trf3, trf4, trf5)

pipe.fit(X_train, y_train)

# display pipeline
    from sklearn import set_config
    set_config(display='diagram')

y_pred = pipe.predict(X_test)
```

Cross validation using pipeline

```
from sklearn.model_selection import cross_val_score
cross_val_score(pipe, X_train, y_train, cv=5,
                scoring='accuracy').mean()
```

Mathematical Transformations

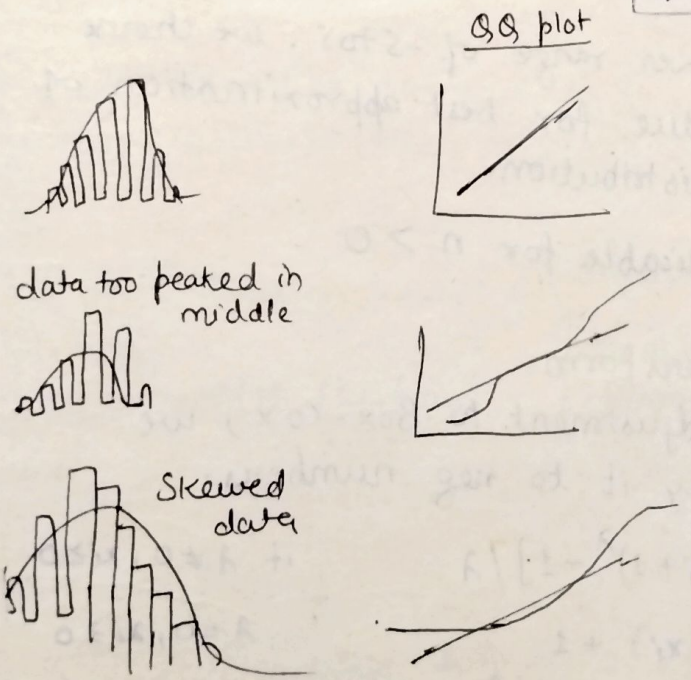
Goal - to make data normally distributed.
Many algorithms perform better on normally distributed data.



how to find if data is normal? ~~sns~~. distplot

pd.skew == 0

QQ plot - Reliable



Log transform

Can not used in -ve data.
should be used for right skewed data.

additive scale ⟶ Multiplicative scale

Reciprocal transform $(1/x)$

Sq (x^2) - left skewed

sqrt $\sqrt{x}$

```
from sklearn.preprocessing import FunctionTransformer

trf = FunctionTransformer(func = np.log1p)
```

Power Transformer

- Box-Cox Transform

$$x_i(\lambda) = \begin{cases} \dfrac{x_i^{\lambda} - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \ln(x_i) & \text{if } \lambda = 0 \end{cases}$$

λ varies over range of -5 to 5. we choose optimal value for best approximation of normal distribution.

only applicable for $n > 0$

- Yeo-Johnson Transform

it is an adjustment to Box-Cox, we can also apply it to neg numbers.

$$x_i^{(\lambda)} = \begin{cases} [(x_i+1)^{\lambda} - 1]/\lambda & \text{if } \lambda \neq 0, x_i \geq 0 \\ \ln(x_i) + 1 & \lambda = 0, x_i \geq 0 \\ -[(-x_i+1)^{2-\lambda} - 1]/(2-\lambda) & \lambda \neq 2, x_i < 0 \\ -\ln(-x_i+1) & \lambda = 2, x_i < 0 \end{cases}$$

scipy.stats as stats - for QQ plot

pt = PowerTransformer(method='box-cox')

Encoding numerical features

Age
22
26 → categorical
34

1) Discretization (Binning)
2) Binarization

Discretization

transforming continuous variables → discrete variables by creating set of contiguous intervals that span the range of variable's values.

Why discretization:
a) To handle outliers
b) To improve the value spread

Age - 23, 42, 57, 81 - - -
0-10, 10-20, 20-30

Types

- Unsupervised Binning
  - Equal width (uniform)
  - Equal frequency (quantile)
  - K-means binning
- Supervised
  - Decision tree binning
- Custom binning

Equal width/Uniform binning - Bins - 10 - Let

$$\text{Interval size} = \frac{\max - \min}{\text{Bins}}$$

- No change in spread of data.
- Handle outliers

K-Mean binning

Let interval = 5 ⇒ centroid

1) Place anywhere these centroids. Calculate distance of point from each centroid. It will be in cluster of closest distance centroid.

2) Now calculate mean of all points in a specific cluster and place its centroid there.

3) Repeat above two steps until it stops changing or improving.

Centroid value - binning value

SKlearn

↳ KBinsDiscretizer (, ,)

- no-of bins
- strategy: uniform, quantile, kmeans
- encoding: ordinal, onehotencoding

Binarization

0, 1

Binarizer (, ,)

- threshold
- copy = True (new column), False (change in same column)

• Handling mixed variables

Mixed data – numerical + categorical

Cabin
B5
C23
D41

class	num
B	5
C	23
D	41

7
3
1
A
C
4

→

num	cat
7	NA
3	NA
1	NA
NA	A
NA	C
4	NA

extract numerical part

```
df['number_numerical'] = pd.to_numeric(df["number"],
                                       errors='coerce',
                                       downcast='integer')
```

extract categorical part

```
df['number_categorical'] = np.where(df['number_numerical'].
                                    isnull(), df['number'],
                                    np.nan)
```

```
.str.extract('(\d+)')   captures numerical part
.str[0]                 # captures first letter
```

working with date and time

by default - ~~string~~ object

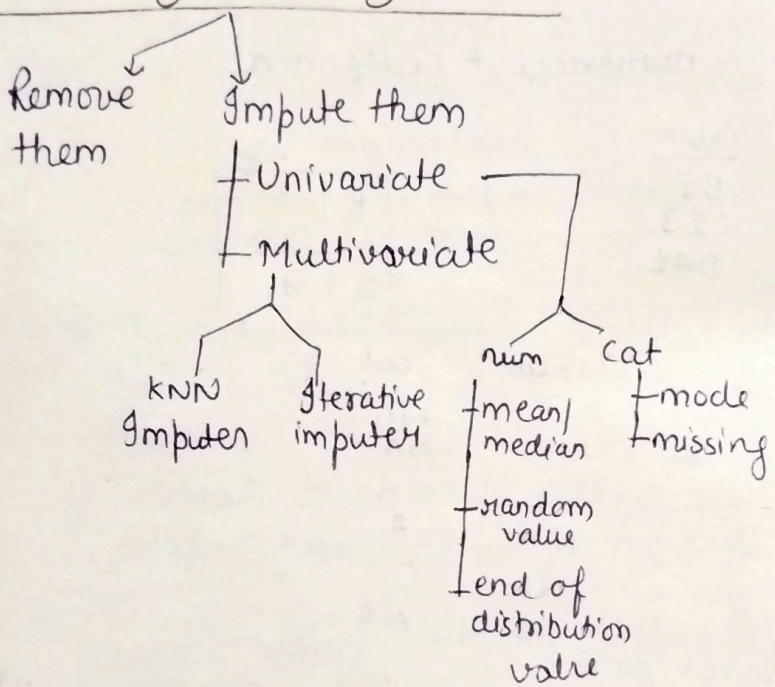
converting to datetime datatype

```
pd.to_datetime(date['date'])
```

extract year

```
date['date'].dt.year
date['date'].dt.month
.dt.month_name()
.dt.day
.dt.dayofweek
.dt.day_name()
.dt.week
```

- Handling missing data



Complete case analysis

also called 'list wise deletion' of cases, consists in descarding observations where values in any of variables are missing

assumptions for CCA -

missing completely at random.

adv - easy to implement.

Preserves variable distribution if data is MCAR.

No data manipulation.

disadv - It can exclude a large fraction of original dataset.

Excluded observations could be informative.

when using model in production, model won't know how to handle missing data.

when to use CCA?

1) MCAR
2) less than 5% missing data rows.

Handling missing numerical data

Num

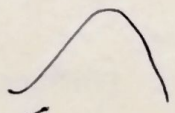
- Univariate (using same column)
  - Mean/median
    - disadv - change distribution shape
      - outliers
      - covariance changes
  - arbitrary
  - end of distribution
  - Random Sample Imputation
    - Sklearn - Simple Imputer (strategy = 'median/mean')
- Multivariate (using different columns)

categ. data - NA - "missing"

Num. data - 1, 0, 99, 999, anything which is not in data

- when data is not missing at random

extension of arbitrary - If



- Normally distributed: (mean + 3σ), (mean - 3σ)
- Skewed (IQR proximity data): $(Q_1 - 1.5\,IQR)$, $(Q_3 + 1.5\,IQR)$

Handling categorical missing data

- Most frequent — Replace with mode. MCAR
  - 5% < — changes distribution of data
  - 10% < — make another category "missing"

Random Simple Imputation

(Both for num and categorical data)

Randomly select any data of that column to fill missing value.

can be done using pandas.

- Preserves the variance of variable.
- Memory heavy for deployment, as we need to store original training set to extract values from and replace NA in coming observations.
- Well suited for linear models

Missing Indicator

Age	Fare	Age NA
27	32	F
41	35	F
NA	41	T
62	32	F

sakit learn - Missing Indicator

mi = MissingIndicator

or

si = SimpleImputer(add_indicator = True)

Automatically select value for Imputation

GridSearchCV

Multivariate Imputation

KNN Imputer

~~The most sim~~

The nearest neighbour column value is use to replace.

nan-euclidean

$$dist(x,y) = sqrt\left(weight * sq.\ distance\ from\ present\ coordinates\right)$$

where,

$$weight = \frac{Total\ no.\ of\ coordinates}{no.\ of\ present\ coordinates}$$

adv
more accurate

dis
more no. of calculation
Production → X_train

```
SKlearn.impute import KNNImputer

knn = KNNImputer(n_neighbours = 3,
                 weights = 'distance'/ )
                           'uniform
```

Iterative Imputer (MICE algorithm)

Multivariate Imputation by chained equations

assumptions
- Missing at random

adv
accurate

disad
slow
production - training data ~~de~~ memory

step 1- Replace all null values with mean of that respective column.
Move left to right in col
2- Remove all col 1 missing values.
3. Predict missing values of col 1 using other cols-
4- Remove all col2 missing values.
5- Predict missing values of col 2 using other cols.
Repeat these ~~fo~~ steps for all cols.

Iteration 0 - ~~Missing vo~~ Replaced with mean

Iteration 1 - all predicted values

now difference between 0^{th} and 1^{st}

whatever difference we got we want to make it close to 0 or equal 0.

So, we will keep repeating the whole process unless we do the same.

Next round, we will take, iteration 1 data as base.

What are outliers

data which behaves completely different.

When is outlier dangerous?

Age

300

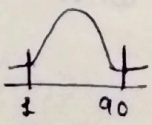
Effect of outlier on ML algorithms

- Linear reg
- Logistic reg
- Adaboost
- Deep Learning

} weights based algorithm

How to treat outliers?

- Trimming
  - data thin
  - fast
- Capping



- treat as missing value
- discretization

How to detect outliers?

1) Normal distribution

$\mu + 3\sigma >$ } Then outlier
$\mu + 3\sigma <$ }

2) Skewed Distribution

$Q_1 - 1.5\,IQR >$

$Q_3 + 1.5\,IQR <$

3. Other distribution

>99%.

<5%.

Techniques for outlier detection and removal

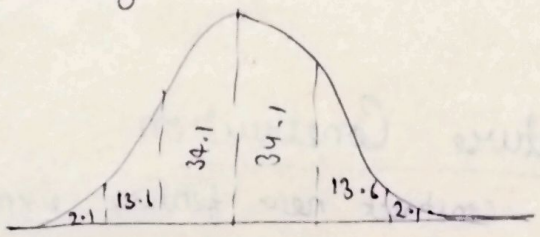
Z score treatment
IQR based filtering
Percentile
Winsorization

- Using Z-score Method

assumption- column is normally distributed on which we are working.

$\mu+\sigma$ } 68%.
$\mu-\sigma$ }

{ $M+2\sigma$ } 95%.
{ $M-2\sigma$ }



If outside the range, then outlier

Zscore $x_i' = \frac{x_i - \mu}{\sigma}$

1
-3×10^3

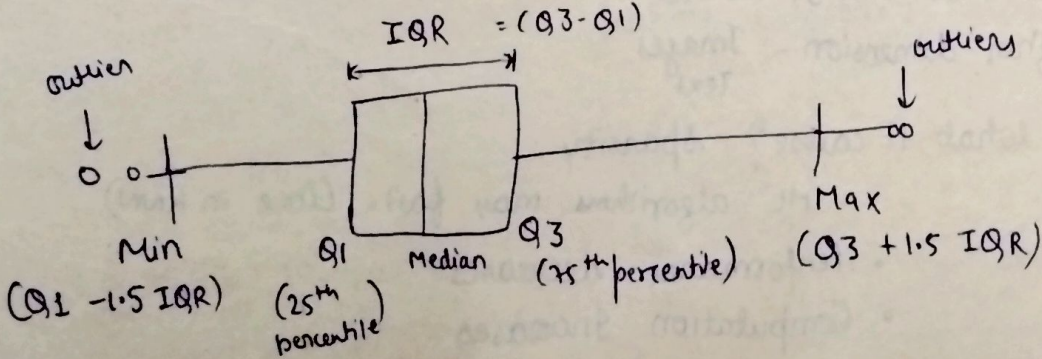
Outlier Treatment

(Trimming)
remove rows that have outliers

(Capping)
(cap outliers with boundaries)

- Using IQR

when data is skewed



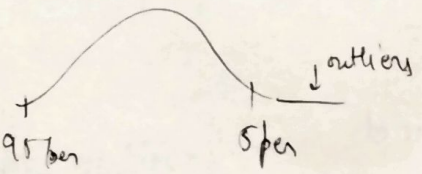
Boxplot

IQR proximity rule

- quantile (0.25)
- quantile (0.75)

np.where(, ,)
condition → if condition true → if condition false

Using Percentile Method



Then after detecting, either trim or capping.
winsorization

Feature Construction

construct new feature using existing feature column.

feature splitting

when more than atomic information is in one column, we go for feature splitting.

Curse of dimensionality

If you increase your features than optimal number, than it may not benifit, computations will increase, model will be complex and accuracy might decrease.

Ex - In MNIST dataset
Higher dimension - Images
Text

What it cause? Sparsity
ML algorithms may fail. (like in KNN)

- Performance decreases
- Computation Increases

Solution → dimensionality Reduction

- feature selection
  - forward selection
  - backward "
- feature extraction
  - PCA
  - LDA
  - tsne

Principal Component Analysis

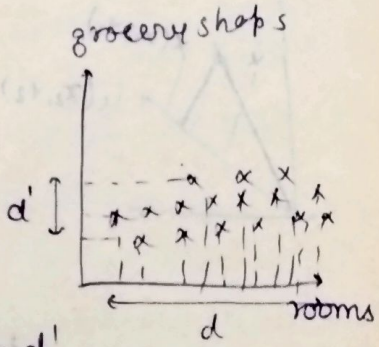
Unsupervised ML problem
feature extraction technique

tries to bring higher dimension data to best possible lower dimension data while keeping essense of data.

- faster execution of algorithm
- visualization

Geometric Intuition

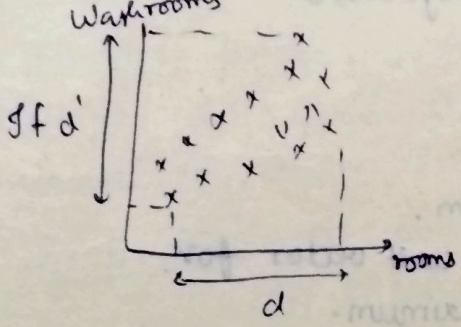
no. of rooms	no. of grocery shops	price
3	2	60
4	0	130
5	6	170
2	10	90



$d > d'$

axis which has max spread
i.e. more variance, that's important.

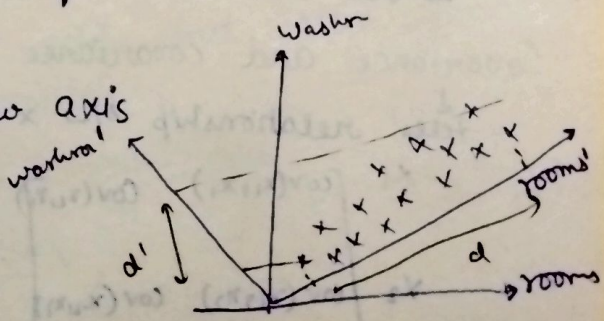
Rooms	Washrooms	Price



$d \simeq d'$

then feature selection is difficult, then feature extraction helps.

What will PCA do?
It will try to find new axis



Why variance is important?

It tells about spread of data.

$$\sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n}$$

spread is not exactly variance

spread $\propto$ Variance

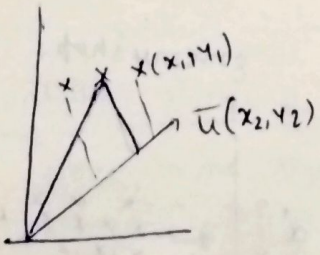
Why mean absolute deviated is not used in place of variance in PCA?

because mod is not differentiable at 0, but square is which is used in variance.

Variance is important in PCA.

We maximize PCA so that covariance doesn't change with other columns.

Problem formulation



$$\frac{\vec{u}\cdot\vec{x}}{|u|=1} = \vec{u}\cdot\vec{x} = u^T x$$

$$[x_1\ y_1]\begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$$

$x_1x_2 + y_1y_2 =$ scalar

$$[u^T x_1]\ [u^T x_2] \cdots [u^T x_n]$$

Variance from all these projections

$$\sum_{i=1}^{n} \frac{(u^T x_i - u^T \bar{x})^2}{n}$$

↑ It should be maximum.

So PCA needs to find such unit vector for which this variance is maximum.

Covariance and covariance matrix

↓ tells relationship b/w x and y.

$$\begin{matrix} x_1 \\ x_2 \end{matrix}\begin{bmatrix} \text{cov}(x_1,x_1) & \text{Cov}(x_2,x_1) \\ \text{cov}(x_1,x_2) & \text{cov}(x_2,x_2) \end{bmatrix}$$

$$\begin{bmatrix} \text{var } x_1 & \text{cov}(x_2, x_1) \\ \text{cov}(x_1, x_2) & \text{var } x_2 \end{bmatrix}$$

$$\text{cov}(a,b) = \text{cov}(b,a)$$

$$\begin{array}{c} \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{bmatrix} V_x & C_{x,y} & C_{x,z} \\ C_{y,x} & V_y & C_{y,z} \\ C_{z,x} & C_{z,y} & V_z \end{bmatrix} \end{array}$$

Linear transformations

Matrices are basically linear transformations which are applied on coordinate system to make changes.

Eigen vectors

Vectors which doesn't change their direction after linear transformation. $A\vec{v} = \lambda\vec{v}$ (matrix, eigen vector, eigen value)

Eigen values

After transformation, how much change is in eigen vector in magnitude.

The largest eigen vector of covariance matrix always points into the direction of target variance of the data and the magnitude of this eigen ~~value~~ is vector is eigen value.

Step by step solution

1. Mean centering the data
2. Find covariance matrix.
3. Find eigen vector and value for covariance matrix.

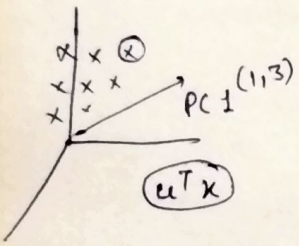
$\lambda_1 > \lambda_2 > \lambda_3$ ← eigen value

PCA1 PCA2 PCA3

After getting points, we need to transform them into lower dimension.

How to transform points?

20 → 1D



f_1	f_2	target
(1000,	3)	

(000, 3) (3, 1)
(1000, 1)

PC1	target
$u^T x$	

3D → 20

(1000, 3) (2, 3)
(1000, 3) (3, 2)
(1000, 2)

PCA using SciKit learn

```
from sklearn.decomposition import PCA
pca = PCA(n_components = )
```

- pca.explained_variance
  # eigen values
- pca.components_
  # eigen vectors
- finding optimum number of Principle Components

$$784 = \lambda_1, \lambda_2, \lambda_3 \dots \lambda_{784}$$

$\lambda_{1\%}$ $\frac{\lambda_1}{\lambda_1 + \lambda_2 + \dots \lambda_{784}} \times 100$ → perentage variance it individually explained of the original data.

will take that many PCA so that 90% variance completes.

$\lambda_1\% + \lambda_2\% + \dots = 90\%$

(pca.explained_variance_ratio_)

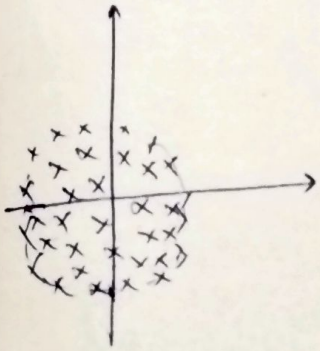
np.cumsum(")

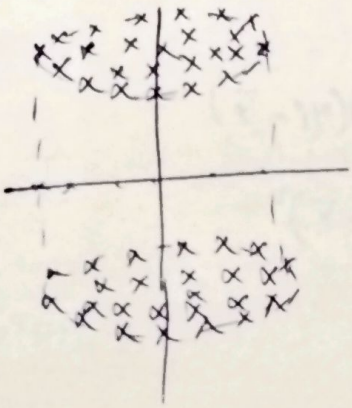
plt.plot(")

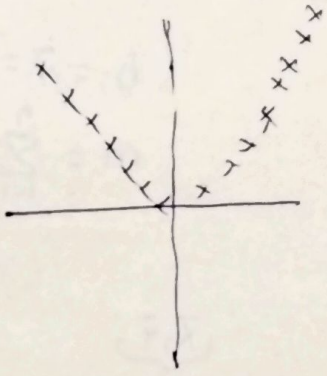
[cumulative sum to check when it reaches 90%]

can see using plot.

When PCA does not work?







you will lose curve if in lower dimension.

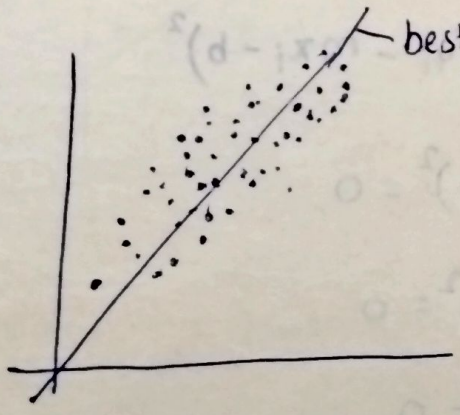
Simple Linear Regression

Supervised ML

1 i/p column | 1 output col

sort of linear data.

why not completely linear?
↳ Real world dataset
Stochastic error



best fit line (closely passes from all points)

$y = mx + b$

m → weightage

package = (m) × cgpa + (b)

m → weightage of cgpa

b → offset

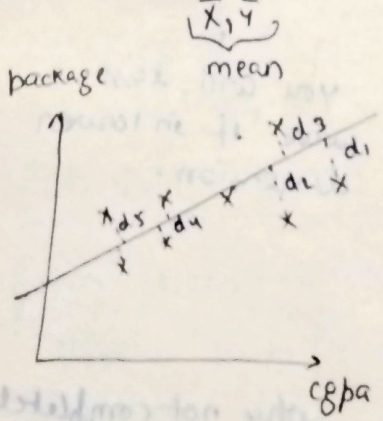
how to find m and b?

- Closed form solution
  - direct formula (Ordinary least squares)
- Nonclosed form Solution (approximation based technique)
  - Gradient descent

Closed form solution-

$$b = \bar{y} - m\bar{x}$$

$$m = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$



$$E = d_1 + d_2 + d_3 + \cdots + d_n$$

$$E = d_1^2 + d_2^2 + d_3^2 + \cdots + d_n^2$$

$$\boxed{E = \sum_{i=1}^{n} d_i^2}$$ ← Error function

why not mod but square

~~✓~~ it won't be differentiable at 0.

$$d_i = y_i - \hat{y}_i$$

$$E = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \qquad \hat{y}_i = mx_i + b$$

$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

$$\frac{\partial E}{\partial b} = \frac{\partial}{\partial b} \sum_{i=1}^{n} (y_i - mx_i - b)^2 = 0$$

$$= \sum \frac{\partial}{\partial b} (y_i - mx_i - b)^2 = 0$$

$$= \sum -2 (y_i - mx_i - b) = 0$$

$$= \sum (y_i - mx_i - b) = 0$$

$$\frac{\sum y_i}{n} - \frac{\sum mx_i}{n} - \frac{\sum b}{n} = \frac{0}{n}$$

$$\bar{y} - m\bar{x} - b = 0$$

$$E = \sum (y_i - mx_i - \bar{y} + m\bar{x})^2$$

$$\frac{\partial E}{\partial m} = \sum \frac{\partial}{\partial m} (y_i - mx_i - \bar{y} + m\bar{x})^2 = 0$$

$$\sum 2(y_i - mx_i - \bar{y} + m\bar{x})(-x_i + \bar{x}) = 0$$

$$\sum -2(y_i - mx_i - \bar{y} + m\bar{x})(x_i - \bar{x}) = 0$$

$$\sum [(y_i - \bar{y}) - m(x_i - \bar{x})](x_i - \bar{x}) = 0$$

$$\sum [(y_i - \bar{y})(x_i - \bar{x}) - m(x_i - \bar{x})^2] = 0$$

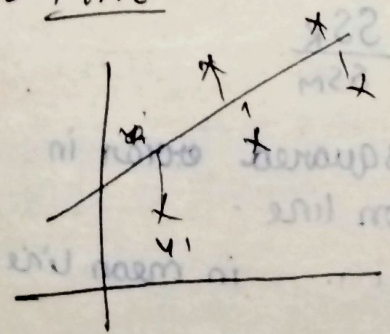
$$\sum (y_i - \bar{y})(x_i - \bar{x}) = m \sum (x_i - \bar{x})^2$$

$$m = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

Regression Metrics

1) MAE
2) MSE
3) RMSE
4) R_2 Score
5) Adjusted R_2 score

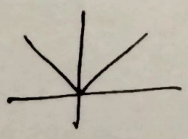
- **MAE**



$$\frac{|y_1 - \hat{y}_1| + |y_2 - \hat{y}_2| + \cdots + |y_n - \hat{y}_n|}{n}$$

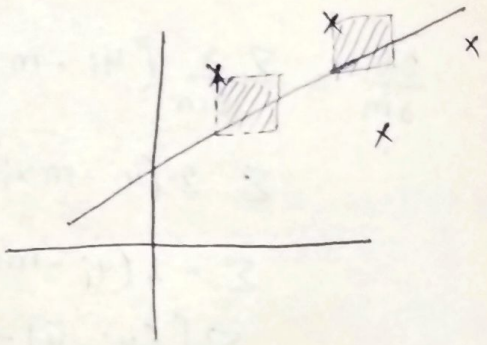
$$mae = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$$

Advantages :- (i) same unit
(ii) Robust outlier

disadv :-  Modulus is not differentiable at 0.

- MSE

$$= \frac{\sum_{i=1}^{n} (y_i - \hat{y})^2}{n}$$



disadv - y - lpa
mse - (lpa)^2

- penalizes the outliers (not Robust to outliers)

- RMSE

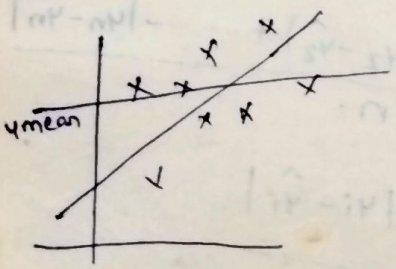
$$RMSE = \sqrt{MSE}$$

$$= \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}}$$

Benifit - RMSE is in same unit to y.

disadv - not robust to outliers.

- R_2 score (coefficient of determination)



$$R_2 = 1 - \frac{SS_R}{SS_M}$$

SS_R - Sum of squared error in regression line.

SS_M = " " in mean line.

R_2 score should go towards 1.

If $R_2 \to 0.80$
cgpa explains 80% of variance in lpa.

- ## Adjusted R_2 score

Usually increase in data → Increase in R_2 score.
But sometimes even after including irrelevant features R_2 score increases, instedd of decreasing.
Then Adjusted R_2 score comes.

$$R^2_{adj} = 1 - \left[\frac{(1 - R_2)(n-1)}{(n-1-k)}\right]$$

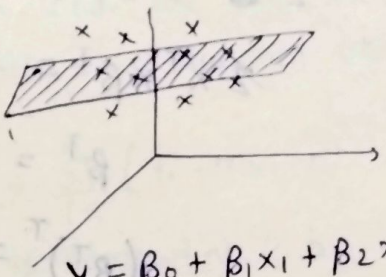
$n \rightarrow$ no. of rows
$k \rightarrow$ no. of independent columns
$R_2 \rightarrow R_2$ score

Using sklearn

from sklearn.metrics import

Multiple Linear Regression

x_1	x_2	x_3	y



$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

$[\beta_0, \beta_1, \beta_2]$

hyperplane that goes closely to points

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_{100} \end{bmatrix} = \begin{bmatrix} \beta_0 & \beta_1 x_{11} & \beta_2 x_{12} & \beta_3 x_{13} \\ \beta_0 & \beta_1 x_{21} & \beta_2 x_{22} & \beta_3 x_{23} \\ \vdots & & & \\ \beta_0 & \beta_1 x_{1001} & \beta_2 x_{1002} & \beta_3 x_{1003} \end{bmatrix}$$

$$\hat{y} = x\beta$$

$$E = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \qquad E = e^T e$$

$$e = \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ y_3 - \hat{y}_3 \\ \vdots \\ y_n - \hat{y}_n \end{bmatrix}$$

$$E = e^T e = (y-\hat{y})^T (y-\hat{y})$$

$$= (y^T - \hat{y}^T)(y-\hat{y})$$

$$= [y^T - (x\beta)^T](y - x\beta)$$

$$= y^T y - y^T x\beta - (x\beta)^T y + (x\beta)^T x\beta$$

$$E = y^T y - 2y^T x\beta + \beta^T x^T x\beta$$

$$\frac{dE}{d\beta} = \frac{d}{d\beta}[y^T y - 2y^T x\beta + \beta^T x^T x\beta] = 0$$

$$= 0 - 2y^T x + \frac{d}{d\beta}[\beta^T x^T x\beta] = 0$$

$$= -2y^T x + 2x^T x\beta^T = 0$$

$$2x^T x\beta^T = 2y^T x$$

$$\beta^T = y^T x (x^T x)^{-1}$$

$$(\beta^T)^T = [y^T x (x^T x)^{-1}]^T$$

$$\beta = [(x^T x)^{-1}]^T (y^T x)^T$$

$$\beta = [(x^T x)^{-1}]^T x^T y$$

$$\beta = x^T x^{-1} x^T y$$

$$\beta = (x^T x)^{-1} x^T y$$

$x \rightarrow$ X_train
$y \rightarrow$ Y_train

Why gradient descent

Calculation of inverse of matrix has high complexity time $(O(n)^3)$. If data is huge, there will be too much computation in higher dimension. To avoid it, we use computation technique called gradient descent.

Gradient descent

first order iterative optimization algorithm for finding a local minimum of a differentiable function.

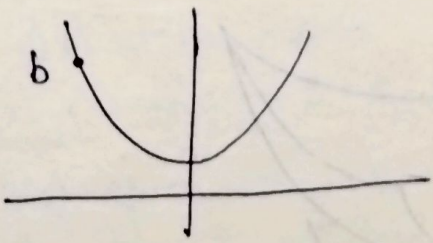
Intuition

(m, b)

Suppose we know m, and we have to find b.
We will start with any random value of b.

$$L = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

L is somewhat $\propto b$

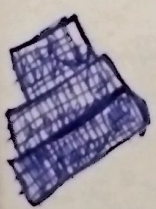


The we will increase or decrease b in the direction opposite to gradient at that point

$b_{new} = b_{old} - slope$

When we do this, there will be great decrease in b.
That's why we use learning rate.

$$b_{new} = b_{old} - \eta \, slope$$



When to stop?

- diff between b_{new} and b_{old} > 0.0001
- we fix iteration number.

Mathematical formulation

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$\Rightarrow \frac{dL}{db} = \frac{d}{db}\left(\sum_{i=1}^{n} (y_i - \hat{y}_i)^2\right)$$

$$= \frac{d}{db} \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

$$= 2 \sum_{i=1}^{n} (y_i - mx_i - b)(-1)$$

$$\text{slope} = -2 \sum_{i=1}^{n} (y_i - mx_i - b)$$

adding m

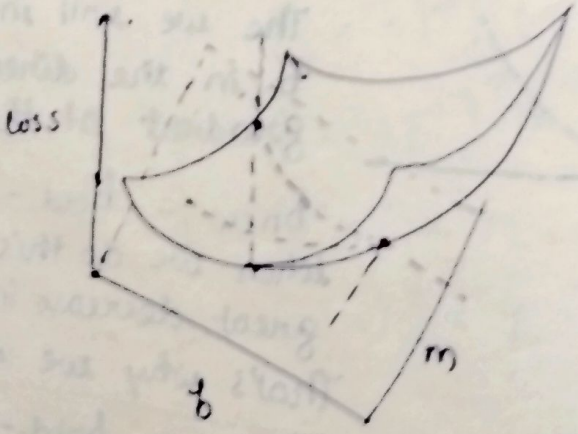
1) Initialize random values for m and b.
2) epochs = 100, lr = 0.01

for i in epocs

$b = b - \eta \cdot \text{slope}$

$m = m - \eta \cdot \text{slope}$

$$L(m,b) = \sum (y_i - mx_i - b)^2$$



$$b\text{-slope} = \frac{\partial L}{\partial b}$$

$$m\text{-slope} = \frac{\partial L}{\partial m}$$

$$\frac{\partial L}{\partial b} = -2 \sum (y_i - mx_i - b)$$

$$\frac{\partial L}{\partial m} = -2 \sum (y_i - mx_i - b) x_i$$

Effect of learning rate

If it will be too low, it will take lot more time and epochs to converge.

If it will be too big, it may never converge.

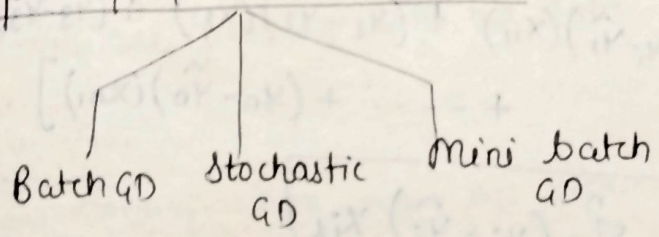
Effect of loss function

If function has local minimum, then initialization point matters so much, otherwise it may never reach to solution.

Effect of data

should be scaled otherwise difficulties may arise i.e. will take more time.

Types of Gradient Descent



Batch Gradient Descent

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \cdots$$

1) Random values

$\beta_0 = 0$, $\beta_1 = \beta_2 = 1$

2) epoch = 100, $\eta = 0.1$

$\beta_0 = \beta_0 - \eta \text{ slope}$ — $\frac{\partial L}{\partial \beta_0}$

$\beta_1 = \beta_1 - \eta \text{ slope}$ — $\frac{\partial L}{\partial \beta_1}$

$\beta_2 = \beta_2 - \eta \text{ slope}$ — $\frac{\partial L}{\partial \beta_1}$

$\frac{n \text{ dim}}{(n+1)\beta_0 - \beta_n}$

$$L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$= \frac{1}{2} \left[(y_i - \hat{y}_i)^2 + (y_2 - \hat{y}_2)^2 \right]$$

$$L = \frac{1}{2} \left[(y_1 - \beta_0 - \beta_1 x_{11} - \beta_2 x_{12})^2 + (y_2 - \beta_0 - \beta_1 x_{21} - \beta_2 x_{22})^2 \right]$$

$$\begin{cases} \hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \\ \hat{y}_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} \\ \hat{y}_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} \end{cases}$$

$$\frac{\partial L}{\partial \beta_0} = \frac{1}{2}\left[2(y_1-\hat{y}_1)(-1) + 2(y_2-\hat{y}_2)(-1)\right]$$

$$= \frac{-2}{2}\left[(y_1-\hat{y}_1) + (y_2-\hat{y}_2)\right]$$

$$= \frac{-2}{n}\left[(y_1-\hat{y}_1) + (y_2-\hat{y}_2) + (y_3-\hat{y}_3) + \cdots (y_n-\hat{y}_n)\right]$$

$$= \frac{-2}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i) = \frac{\partial L}{\partial \beta_0}$$

$$L = \frac{1}{2}\left[(y_1-\hat{y}_1)^2 + (y_2-\hat{y}_2)^2\right]$$

$$L = \frac{1}{2}\left[(y_1-\beta_0-\beta_1 x_{11}-\beta_2 x_{12})^2 + (y_2-\beta_0-\beta_1 x_{21}-\beta_2 x_{22})^2\right]$$

$$\frac{\partial L}{\partial \beta_1} = \frac{1}{2}\left[2(y_1-\hat{y}_1)(-x_{11}) + 2(y_2-\hat{y}_2)(-x_{21})\right]$$

$$\frac{\partial L}{\partial \beta_1} = \frac{-2}{n}\left[(y_1-\hat{y}_1)(x_{11}) + (y_2-\hat{y}_2)(x_{21}) + (y_3-\hat{y}_3)(x_{31}) + \cdots + (y_n-\hat{y}_n)(x_{n1})\right]$$

$$\boxed{\frac{\partial L}{\partial \beta_1} = -\frac{2}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)\,x_{i1}}$$

x_{i1} - 1st col data

$$\frac{\partial L}{\partial \beta_2} = -\frac{2}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)\,x_{i2}$$

$$\boxed{\frac{\partial L}{\partial \beta_m} = -\frac{2}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)\,x_{im}}$$

Problem with Batch GD

after seeing whole data in one epoch, we take ~~one~~ single update on our coefficient.

1 epoch → 1 update

Let n = 1000
col = 5
6 coeff

epoch → 50

50 —

for 1 coefficient → 1000 derivatives
for 6 — 6000 derivatives

6000 x 50
300000 derivatives

- When n increases, derivative increases. So many computations for large data. Algorithm will be slow.
- Hardware problem with big data.

Stochastic GD

seeing single row - we update,

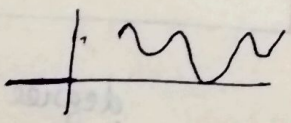
for 1 epoch - n updates

less no. of epochs will be needed so less time.

- faster convergence.
- Somewhat inconsistent.
- random nature as we rely on random index row.

When to use?

- Big data
- Non-convex function



When we reach near solution also.
It shows random nature,
we can use learning schedulers for this problem.

Sklearn.linear_model.SGDRegressor

Mini-Batch Gradient Descent

Batch
1 update / epoch
{slow}
{small and convex data sets}

Stochastic
n updates / epoch
{fast}
big data
non-convex
(random)

Mini-batch Gradient descent

batches update / epoch

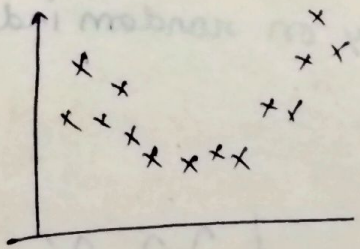
sgd.partial_fit( )

Polynomial Regression

$y = \beta_0 + \beta_1 x$ - Simple linear Regression

$y = \beta_0 + \beta_1 x + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_n x_n$

↳ Multiple linear regression



$y = \beta_0 + \beta_1 x + \beta_2 x^2$

↳ simple polynomial reg.

degree
↳ hyperparameter

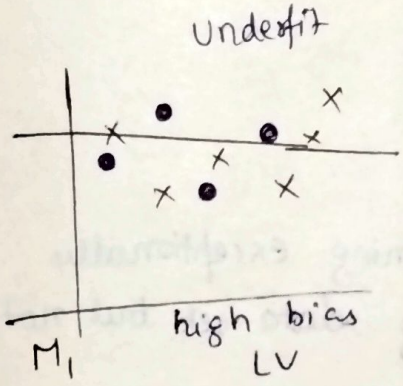
```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2)
```

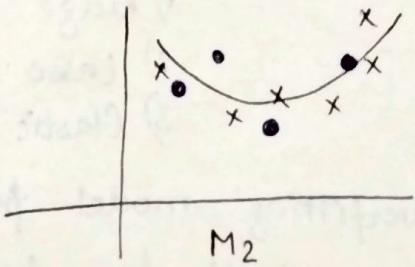
for 1 row - 3 values will come

x^0, x^1, x^2

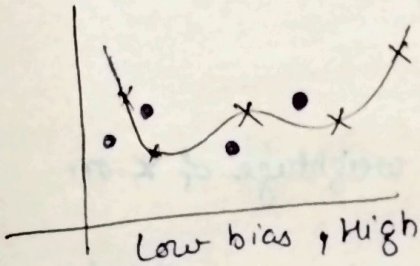
If degree will increase too much - overfitting may occur.

Bias Variance Trade-off





x - training set
● - test set



Bias- the inability ~~to~~ of machine learning model to truly capture the relationship in training data.

Variance - diff. in error in training and testing dataset

Overfitting - test error >> train error

Underfitting - not even performing well on training set

Goal - { Low Bias, Low variance }

- Regularization
- Bagging
- Boosting

Ridge regularization

Regularization - You induce some extra information to model to reduce or avoid overfitting.

3 types -

1) Ridge (L_2)
2) Lasso (L_1)
3) Elastic Net

Overfitting - model performing exceptionally well for training data set but not for testing.

$$y = mx + b$$

m tells about weightage of x on predicting y.

If m will be too large, y will only depend on x, it may overfit.

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

after regularization

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda(m^2)$$

L2 norm

λ - hyperparameter $\qquad \lambda(m_1^2 + m_2^2 + \cdots)$

from sklearn.linear_model import Ridge

Ridge regression formulation

$$y = mx + b$$

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda m^2$$

$$L = \sum_{i=1}^{n} (y_i - mx_i - b)^2 + \lambda m^2$$

after $\frac{\partial L}{\partial b}$ $\quad b = \bar{y} - m\bar{x}$ $\qquad \frac{\partial L}{\partial b} = 0$

$\frac{\partial L}{\partial m} = 0$

$$\frac{\partial L}{\partial m} = 2\sum_{i=1}^{n}(y_i - mx_i - \bar{y} + m\bar{x})(-x_i + \bar{x}) + 2\lambda m = 0$$

$$\Rightarrow -2\sum_{i=1}^{n}(y_i - \bar{y} - mx_i + m\bar{x})(x_i - \bar{x}) + 2\lambda m = 0$$

$$\lambda m - \sum_{i=1}^{n}[(y_i - \bar{y}) - m(x_i - \bar{x})](x_i - \bar{x}) = 0$$

$$\lambda m - \sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x}) - m(x_i - \bar{x})^2 = 0$$

$$\lambda m + m\sum_{i=1}^{n}(x_i - \bar{x})^2 = \sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})$$

$$\boxed{m = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2 + \lambda}} \quad - \text{Ridge}$$

$$m = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad = \text{original regression}$$

for nD data

x_1	x_2	---	x_n	y
w_1	w_2	---	w_n	$(n+1)$

m rows

$$L = \sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

$$= (xw - y)^T(xw - y)$$

$$x = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & & \\ 1 & x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

$$w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

$$L = (xw - y)^T (xw - y) + \lambda \|w\|^2$$

$w^T w$ $\{\lambda w_0^2 + \lambda w_1^2 \dots + \lambda w_n^2\}$

$$L = (xw - y)^T (xw - y) + \lambda w^T w$$

$$L = [(xw)^T - y^T](xw - y) + \lambda w^T w$$

$$= (w^T x^T - y^T)(xw - y) + \lambda w^T w$$

$$= w^T x^T x w - w^T x^T y - y^T x w + y^T y + \lambda w^T w$$

$$L = w^T x^T x w - 2 w^T x^T y + y^T y + \lambda w^T w$$

$$\frac{dL}{dw} = 2 x^T x w - 2 x^T y + 0 + 2\lambda w = 0$$

$$x^T x w + \lambda w = x^T y$$

$$(x^T x + \lambda I) w = x^T y$$

$$\boxed{w = (x^T x + \lambda I)^{-1} x^T y}$$

$I = ((n \times 1), (n \times 1))$

Ridge Regression using Gradient Descent

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$L = (xw - y)^T (xw - y) + \lambda \|w\|^2$$

$$L = (xw - y)^T (xw - y) + \lambda w^T w$$

$$x = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ 1 & x_{m1} & x_{m2} & x_{m3} & \dots & x_{mn} \end{bmatrix} \quad \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \quad \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

$w_0, w_1, \dots w_n$ (parameters)

$$w_0 = w_0 - \eta \frac{\partial L}{\partial w_0}$$

$$w_1 = w_1 - \eta \frac{\partial L}{\partial w_1}$$

$$w_{new} = w_{old} - \eta \frac{\Delta L}{\Delta w} \quad \text{gradient}$$

$$L = (xw - y)^T (xw - y) + \lambda w^T w$$

$$= \frac{1}{2}(w^T x^T - y^T)(xw - y) + \frac{1}{2}\lambda w^T w$$

$$= \frac{1}{2}\left[w^T x^T xw - w^T x^T y - y^T w x + y^T y\right] + \frac{1}{2} y w^T w$$

$$= \frac{1}{2}\left[w^T x^T xw - 2y^T w^T x^T + y^T y\right] + \frac{1}{2}\lambda w^T w$$

$$\frac{dL}{dw} = \frac{1}{2}\left[2x^T xw - 2y^T x^T\right] + \frac{1}{2} \times 2\lambda w$$

$$\frac{dL}{dw} = x^T xw - y^T x^T + \lambda w$$

Key Points - Ridge Regression

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \|w\|^2$$

$$\hookrightarrow \lambda (w_1^2 + w_2^2 + \cdots + w_n^2)^2$$

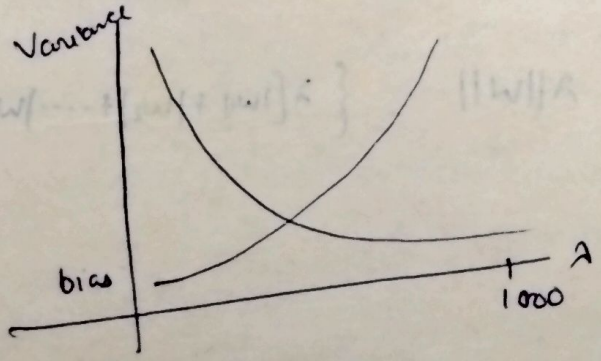
shrinkage coef

how the coefficients get affected?

$$\lambda \to 0 \to \infty$$

$w_1, w_2 \cdots w_n \longrightarrow \downarrow 0$ tends to 0.

- Higher values coeff are impacted more.
- Bias variance tradeoff



Impact on Loss function

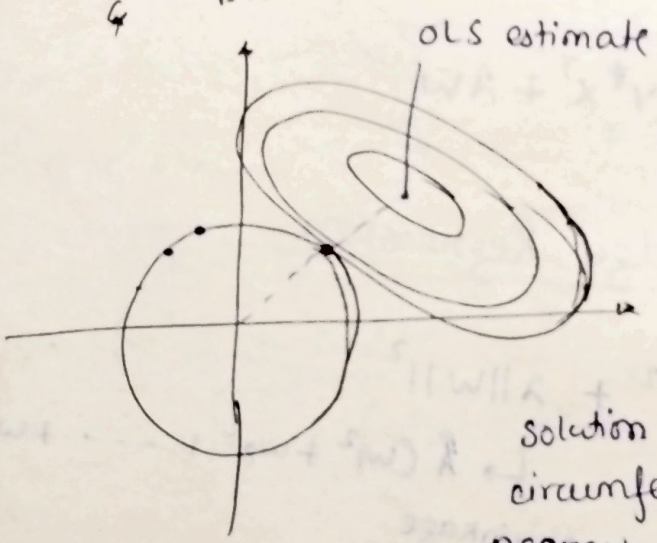
$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \|w\|^2$$

m, b → b is constant

$$L = \sum_{i=1}^{n} (y_i - m x_i)^2 + \lambda m^2$$

$\lambda \uparrow$ Loss function shrink, move towards origin

why called ridge ?



solution will always be at circumference of circle on point nearest to OLS estimate solution.

That's why called ridge.

Practical tip

$x_1, x_2, x_3 \dots$

$> = 2$

Lasso Regression

L_1 Regularization

$$L = MSE + \lambda \|w\| \quad - \{\lambda[|w_1| + |w_2| + \dots |w_n|]\}$$

Like for high dimensional data

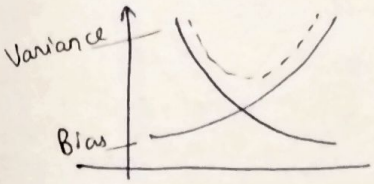
$x_1, x_2, x_3 \dots\dots x_n$

In lasso regression,

When λ increases, some coefficients will become 0.

Inherently feature selection

- higher coefficients are affected more
- Impact on bias and variance



$\lambda \uparrow$ overfitting $\downarrow$ Bias $\uparrow$

$\lambda \uparrow$ Variance $\downarrow$

- Effect of regularization on loss function
  On increasing λ, it moves towards 0 and stops at 0 only on further increasing.

Why Lasso creates sparsity?

Simple

$y = mx + b$

$b = \bar{y} - m\bar{x}$

$$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda |m|$$

$$L = \sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x})^2 + 2\lambda |m|$$

$m > 0$

$$\frac{d}{dm} \sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x})^2 + 2\lambda m$$

$$= 2 \sum (y_i - mx_i - \bar{y} + m\bar{x})(-x_i + \bar{x}) + 2\lambda = 0$$

$$-2 \sum [(y_i - \bar{y}) - m(x_i - \bar{x})](x_i - \bar{x}) + 2\lambda = 0$$

$$- \sum [(y_i - \bar{y})(x_i - \bar{x}) - m(x_i - \bar{x})^2] + \lambda = 0$$

$$m \sum (x_i - \bar{x})^2 = \sum (y_i - \bar{y})(x_i - \bar{x}) - \lambda$$

$m > 0$

$$m = \frac{\sum (y_i - \bar{y})(x_i - \bar{x}) - \lambda}{\sum (x_i - \bar{x})^2}$$

$m = 0$

$$m = \frac{\sum (y_i - \bar{y})(x_i - \bar{x})}{\sum (x_i - \bar{x})^2}$$

$m < 0$

$$m = \frac{\sum (y_i - \bar{y})(x_i - \bar{x}) + \lambda}{\sum (x_i - \bar{x})^2}$$

In ridge case, λ is in denominator, then 0 can only be depend on numerator, and num won't be 0.

In lasso, λ will be in numerator so from there it drives m to 0 and stop it there.

ElasticNet Regression

Ridge - when all columns are important.
Lasso - Inherently feature selection.
Elasticnet - combination of both when huge data.

$$L = \sum (y_i - \hat{y}_i)^2 + a||w||^2 + b||w||$$

$$\lambda = a + b$$

$$l\_ratio = \frac{a}{a+b}$$

by default, $\lambda = 1$
$l_1 ratio = 0.5$

$l_1 ratio > 0.9$ either multicollinearity exists between input columns.
↓
90% ridge
10% lasso

multicollinearity:- when one input depends or changes due to other input.

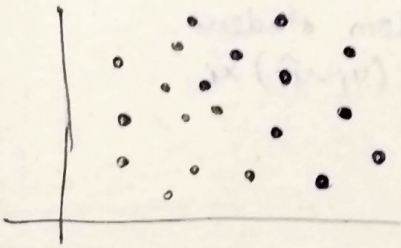
either directly ElasticNet or SGD → scikit-learn

Logistic Regression

2 perspective { Geometry, Probability

is supervised ML algorithm that performs binary classification tasks by predicting the probability of outcome.

Perceptron trick



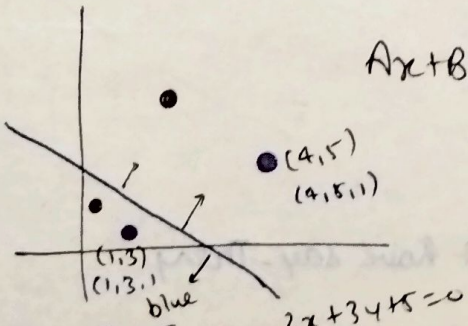
Start with a random line, and start asking each point, if it has classified correctly or not, if not then transform that line to move toward that point to transform that correctly.

- will do it while convergence.
  or
- no. of epochs

Transformations

either point will be in +ve region or -ve region.

$Ax + By + C = 0$



$$\begin{array}{r} 2\ \ 3\ \ 5 \\ -\ 4\ \ 5\ \ 1 \\ \hline -2x - 2y + 4 = 0 \end{array}$$

$$\begin{array}{r} 2\ \ 3\ \ 5 \\ +\ \eta 1\ \ \eta 3\ \ \eta 1 \\ \hline 3x + 6y + 6 = 0 \end{array}$$

coef = coef − η coordinate

Algorithm

epoch = 1000 $\quad \eta = 0.1$

for i in range (epochs)

randomly select student

if $x_i \in N$ and $\sum_{i=0}^{2} w_i x_i \geq 0$

$w_{new} = w_{old} - \eta i$

if $x_i \in P$ and $\sum_{i=0}^{2} w_i x_i < 0$

x_0	(x_1) CGPA	(x_2) IQ	y placed
1	7.5	6.2	1
1	8.9	109	1
1	7.0	81	0

$Ax + By + c = 0$

$w_0 + w_1 x_1 + w_2 x_2 = 0$

Simplified algo

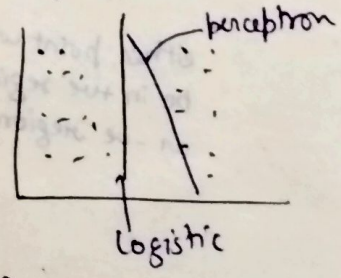
$$w_n = w_o + \eta (y_i - \hat{y}_i) x_i$$

y_i	$\hat{y}_i$	$y_i - \hat{y}_i$
1	1	0
0	0	0
1	0	1
0	1	-1

for i in range (epochs):
  select random student
  $w_n = w_o + \eta (y_i - \hat{y}_i) x_i$

Scikit logistic Regression gives better result. gives best line.

Test error may be large, even we minimize loss in training error. This is a major flaw of perceptron trick.



Possible solution?

Even correctly classified point will have say. They will push the line.

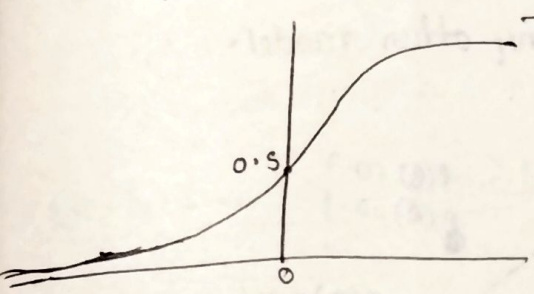
For misclassified points - points that are far will pull line strongly comparing to points that are near.

For correctly classified points - Near points will push much more than far point.

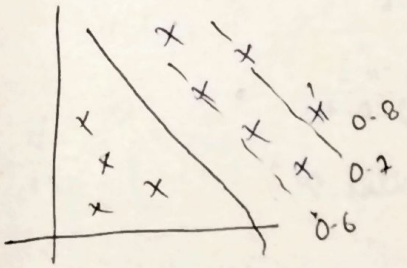
$$w_n = w_o + \eta (y_i - \hat{y}_i) x_i$$

Till now, we are using step function either 0 or 1.

Now, we will use sigmoid function.



$$a = \frac{1}{1 + \exp(-z)}$$



$\sum w_i x_i = 0 \quad \sigma = 0.5$

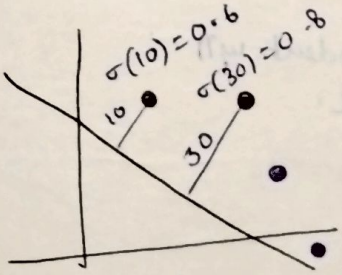
$\sum w_i x_i > 0 \quad \sigma > 0.5$

Impact of Sigmoid

$$w_n = w_o + \eta (y_i - \hat{y}_i) x_i$$

$y_i = \sigma(z)$

where $z = \sum w_i x_i$

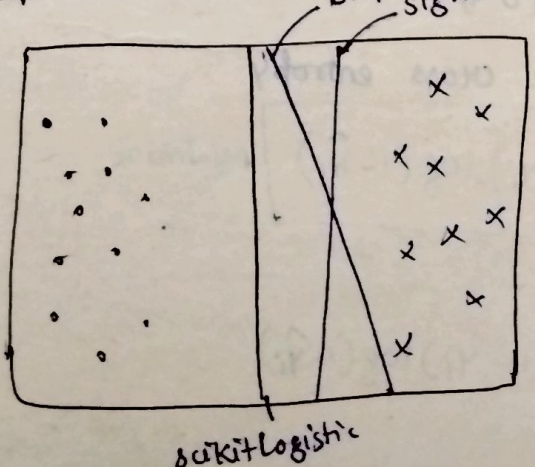


y_i	$\hat{y}_i$	$y_i - \hat{y}_i$
1	0.6	0.4
1	0.8	0.2
0		
0		

$w_n = w_o + \eta \times 0.4 \times x_i = x_1$

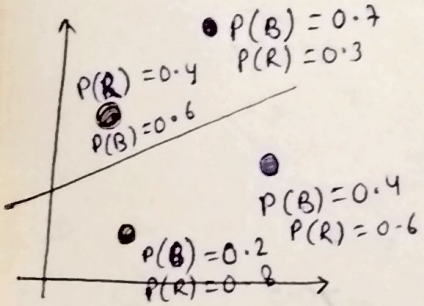
$w_n = w_o + \eta * 0.2 \, x_i = x_2$ $\quad x_1 > x_2$

Improvement

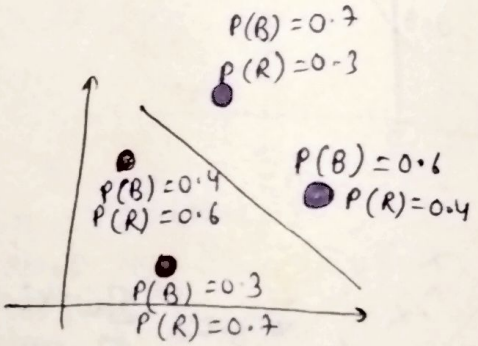


we are selecting 1000 random points.
that doesn't guarantee optimal function.
We'll try to find loss function which tells how better the model is from any other model.

Maximum likelihood



model 1



model 2.

will take probability of actual output and multiply

$\hat{y} = \sigma(z)$

$= 0.7 \times 0.4) \times 0.4 \times 0.8$

$= 0.089$

$0.7 \times 0.6 \times 0.6 \times 0.2$

$= 0.176$

model 2 is better than 1.

will convert product to sum, because product will be too small for large dataset to compare.

$\log(ab) = \log a + \log b$

$0 < a < 1$, log will be -ve

$-\log a - \log b$

$\sum$ of log of maximum likelihood – cross entropy

$\log(0.1) > \log(0.9)$

So we will minimize cross entropy

$$L = \sum_{i=1}^{2} -y_i \log(\hat{y}_i) - (1-y_i) \log(1-\hat{y}_i)$$ Minimize

Avg.

$$L = \frac{1}{n} \sum_{i=1}^{n} -y_i \log(\hat{y}_i) - (1-y_i) \log(1-\hat{y}_i)$$

log loss error
binary cross entropy

no closed form solution
find w_0, w_1, w_2 --- to minimize loss function.
So, we will use gradient descent.

- **Derivative of sigmoid**

$$a = \frac{1}{1+\exp(-z)}$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\frac{d}{dx}\left[\frac{1}{1+e^{-x}}\right] = \frac{d}{dx}\left[(1+e^{-x})^{-1}\right]$$

$$= \frac{-1}{(1+e^{-x})^2}\frac{d}{dx}(1+e^{-x})$$

$$= \frac{-1}{(1+e^{-x})^2}\frac{d}{dx}(e^{-x})$$

$$= \frac{-e^{-x}}{(1+e^{-x})^2}\frac{d}{dx}(-x)$$

$$= \frac{e^{-x}}{(1+e^{-x})^2}$$

$$= \frac{1}{(1+e^{-x})}\cdot\frac{e^{-x}}{(1+e^{-x})}$$

$$= \sigma(x)\left[\frac{e^{-x}}{1+e^{-x}}\right]$$

$$= \sigma(x)\left[\frac{1+e^{-x}-1}{1+e^{-x}}\right]$$

$$= \sigma(x)\left[\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right]$$

$$= \sigma(x)\left[1-\sigma(x)\right]$$

Gradient descent

Rows = m Columns = n

	1	2	3	...	n	y	$\hat{y}$
1	x_{11}	x_{12}	x_{13}	------	x_{1n}	y_1	
2	x_{21}	x_{22}	x_{23}	------	x_{2n}	y_2	
⋮	⋮						
m	x_{m1}	x_{m2}	x_{m3}	------	x_{mn}	y_m	

$$\sigma(w_1x_{11} + w_2x_{12} + w_3x_{13} + \dots + w_nx_{1n} + w_0) = \hat{y}_1$$

$$\sigma(w_1x_{21} + w_2x_{22} + w_3x_{23} + \dots + w_nx_{2n} + w_0) = \hat{y}_2$$

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_m \end{bmatrix} = \begin{bmatrix} \sigma(w_0 + w_1x_{11} + w_2x_{12} + \dots + w_nx_{1n}) \\ \sigma(w_0 + w_1x_{21} + w_2x_{22} + \dots + w_nx_{2n}) \\ \vdots \\ \sigma(w_0 + w_1x_{m1} + w_2x_{m2} + \dots + w_nx_{mn}) \end{bmatrix}$$

$$\hat{y} = \sigma\left(\begin{bmatrix} w_0 + w_1x_{11} + w_2x_{12} + \dots + w_nx_{1n} \\ w_0 + w_1x_{21} + w_2x_{22} + \dots + w_nx_{2n} \\ \vdots \\ w_0 + w_1x_{m1} + w_2x_{m2} + \dots + w_nx_{mn} \end{bmatrix}\right)$$

$$\hat{y} = \sigma\left(\begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{22} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & & \\ 1 & x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix}\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}\right)$$

$$\boxed{\hat{y} = \sigma(xw)}$$

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$$

$$\sum_{i=1}^{m} y_i \log \hat{y}_i = y_1 \log \hat{y}_1 + y_2 \log \hat{y}_2 + \dots + y_m \log \hat{y}_m$$

$$= \begin{bmatrix} y_1 & y_2 & y_3 & \dots & y_m \end{bmatrix}\begin{bmatrix} \log \hat{y}_1 \\ \log \hat{y}_2 \\ \vdots \\ \log \hat{y}_m \end{bmatrix}$$

$$L = -\frac{1}{m}\left[y \log \hat{y} + (1-y)\log(1-\hat{y})\right]$$

where $\hat{y} = \sigma(xw)$

Loss function in matrix form

$$L = -\frac{1}{m}\left[y \log(\sigma(xw)) + (1-y)\log(1-\sigma(wx))\right]$$

for i in epoch

$$w = w - \eta \frac{\Delta L}{\Delta w}$$

$$\frac{\Delta L}{\Delta w} = \left[\frac{\partial L}{\partial w_0}, \frac{\partial L}{\partial w_1} \frac{\partial L}{\partial w_2}, \cdots \frac{\partial L}{\partial w_n}\right]$$

$$L = -\frac{1}{m}\left[y \log \hat{y} + (1-y)\log(1-\hat{y})\right]$$

$\frac{dL}{dw}\, y \log \hat{y} \Rightarrow y \frac{dL}{dw} \log \hat{y}$

$\Rightarrow \frac{y}{\hat{y}} \frac{d}{dL}(\hat{y})$

$\Rightarrow \frac{y}{\hat{y}} \frac{d}{dL} \sigma(wx)$

$\Rightarrow \frac{y}{\hat{y}} \sigma(wx)\left[1-\sigma(wx)\right] \frac{d}{dw}(wx)$

$\Rightarrow \frac{y}{\hat{y}} \hat{y}(1-\hat{y})x = y(1-\hat{y})x$

$\frac{d}{dw}(1-y)\log(1-y) \Rightarrow (1-y)\frac{d}{dw}\log(1-\hat{y})$

$\Rightarrow \frac{(1-y)}{(1-\hat{y})} \frac{d}{dw}[1-\hat{y}]$

$\Rightarrow -\frac{(1-y)}{(1-\hat{y})} \frac{d}{dw} \sigma(wx)$

$\Rightarrow -\frac{(1-y)}{(1-\hat{y})} \left[\sigma(wx)(1-\sigma(wx)\right] \frac{d}{dL}(w$

$\rightarrow -\frac{(1-y)}{(1-\hat{y})} \hat{y}(1-\hat{y})x = -\hat{y}(1-y)x$

$$\frac{dL}{dw} = -\frac{1}{m}\left[y(1-\hat{y})x - \hat{y}(1-y)x\right]$$

$$= -\frac{1}{m}\left[y(1-\hat{y}) - \hat{y}(1-y)\right]x$$

$$= -\frac{1}{m}\left[y - y\hat{y} - \hat{y} + y\hat{y}\right]x$$

$$\boxed{\frac{\Delta L}{\Delta w} = \frac{-1}{m}(y-\hat{y})x}$$

$$\boxed{w = w + \eta\frac{1}{m}(y-\hat{y})x}$$

$$w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}_{(n+1,1)} \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{21} & x_{22} & \cdots & x_{2n} \\ 1 & \vdots & & & \\ \vdots & x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}_{(m,n+1)} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}_{(m,1)}$$

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_m \end{bmatrix}_{(m,1)}$$

Classification Metrics

SKlearn.metrics

- accuracy_score

$$\text{accuracy} = \frac{\text{no. of correct pred}}{\text{Total no. of pred}}$$

how much accuracy is good?
depends on problem.

Problem with accuracy.
doesn't tells which classification is wrong mostly.

- Confusion matrix

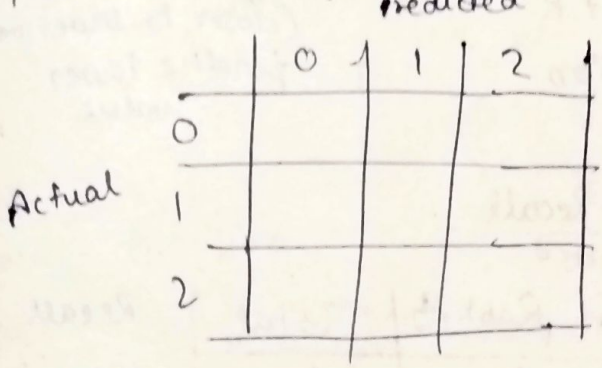
Actual \ Prediction	1	0
1	TP	FN
0	FP	TN

FP
FN

Type 1 error - False Positive

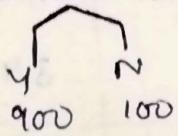
Type 2 error - False Negative

Confusion matrix for multi-classification problem



when accuracy is misleading?

Imbalanced dataset



1000 people

999 not terrorist — 1 terrorist

model → No one is terrorist

Predicted

Actual	1	0
1	0	1
0	0	999

$$\text{Accuracy} = \frac{999}{999+1} = 99.9\%$$

Precision - what proportion of predicted positive is truly positive?

Spam Email detection

$$\frac{TP}{TP+FP}$$

Recall :- what proportion of actual positives is correctly classified?

$$\text{Recall} = \frac{TP}{TP+FN}$$

Cancer detection

Whether to choose precision or recall depends on nature of problem.

F1 Score

combination of precision and recall

$$F1\ score = \frac{2PR}{P+R}$$ - Harmonic mean (closer to lower value) penalize lower value

P - precision
R - Recall

Multi-class Precision and Recall

	Predicted: Dog	Cat	Rabbit	Total	Recall
Dog	25	5	10	40	0.62
Cat	0	30	4	34	0.88
Rabbit	4	10	20	34	0.58
Total	29	45	34		
Precision	0.86	0.66	0.58		

$P_D = \frac{25}{29} = 0.86$ $P_C = \frac{30}{45} = 0.66$ $P_R = \frac{20}{34} = 0.58$

Macro precision - $\frac{0.86 + 0.66 + 0.58}{3} = 0.70$

weighted precision - $\frac{40}{108} \times 0.86 + \frac{34}{108} \times 0.66 + \frac{34}{108} 0.58$

for imbalanced classes

$= 0.71$

$R_D = \frac{25}{40}$ $R_C = \frac{34}{30}$ $R_R = \frac{20}{34}$

Macro recall
weighted recall

F_{1D} F_{1R} F_{1R}

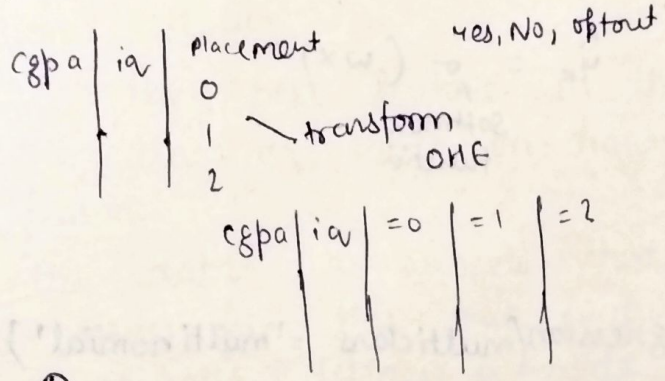
Softmax Regression (multiple classes)
(Multinomial Logistic Regression)

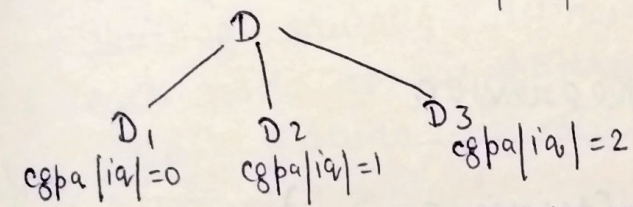
K = no. of classes

Softmax function

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{i=1}^{K} e^{z_j}}$$

Training Intuition

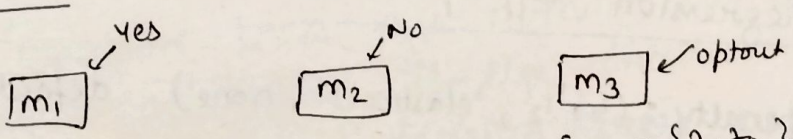




train 3 diff logistic regression model

w_0^0, w_1^0, w_2^0 w_0^1, w_1^1, w_2^1 $w_0^{(2)}, w_1^{(2)}, w_2^{(2)}$

Prediction



$S_x = \{7, 70\}$

$z_1 = 7 \times w_1^{(1)} + 70 w_2^{(1)} + w_0^{(1)}$

$z_2 = 7 w_1^{(2)} + 70 w_2^{(2)} + w_0^{(2)}$

$z_3 = 7 w_1^{(3)} + 70 w_2^{(3)} + w_0^{(3)}$

$$\sigma(Y) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$\sigma(N) =$

$\sigma(O) =$

This approach could be slow for large datasets, classification classes is more.

So will change in loss function **using** which we can **train** for only one model.

Loss function

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$$

↑ for binary

for softmax

$$L = -\frac{1}{m}\sum_{i=1}^{m}\sum_{K=1}^{K} y_K^{(i)} \log\left(\hat{y}_K^{(i)}\right)$$

$$\hat{y}_K = \sigma(wx)$$

↑ Softmax function

LogisticRegression(multiclass = 'multinomial')

Polynomial Logistic Regression

```
poly = PolynomialFeatures(degree = , )
X_trf = poly.fit_transform(X)
```

Logistic Regression Hyperparameters

penalty : ('l1', 'l2', 'elasticnet', 'none') default = 'l2'

C : float, default = 1.0
↳ Inverse of regularization strength.

class_weight : dict or 'balanced', default=None
↳ used for imbalanced dataset

Solver : ('newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga')

max_iter : int, default = 100

multi_class ('multinomial', 'auto', 'ovr')

warm_start : bool, default=False
when true, reuse the solution of previous call to fit as initialization

l1_ratio = float, default=None

Decision Trees

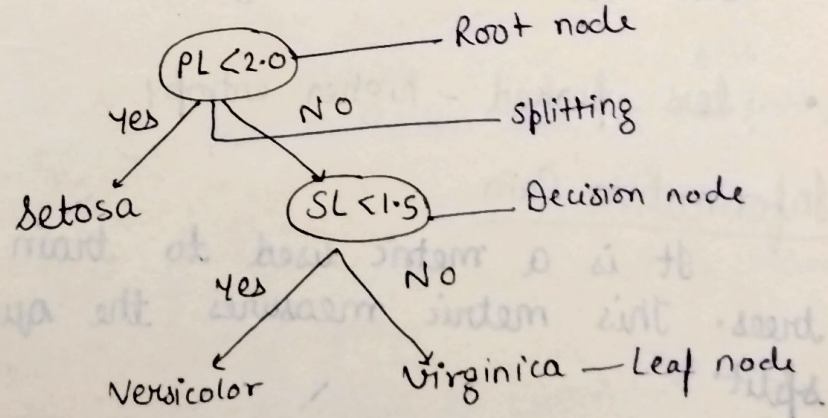
Pseudo code

- Begin with training dataset, which should have some feature variables and classification or regression output.
- determine the 'best feature' in the dataset to split the data on, more on how we define 'best feature'.
- Split the data into subsets that contain the correct values for this best feature. This splitting basically defines a node on the tree i.e. each node is a splitting point based on a certain feature from our data.
- Recursively generate new tree nodes by using the subset of data created from step 3.

Conclusion

Decision trees are nothing but a giant structure of nested if-else condition.

Mathematically, decision trees use hyperplanes which runn parallel to any one of the axes to cut your coordinate system into hyper cuboids.



Advantages

- Intuitive and easy to understand.
- Minimal data preparation is required.
- The cost of using the tree for inference is logarithmic in the number of data points used to train the tree.

Disadvantages

Overfitting

Prone to errors for imbalanced datasets.

CART - Classification and Regression Trees

Decision Trees Entropy

Entropy is measure of disorder.
or measure of purity / impurity.

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

p_i is frequentist probability of an element i in our data.

- More the uncertainty more is entropy.
- For 2 class problem, min entropy is 0 and max is 1.
- For more than 2 classes, the min entropy is 0 but the max can be greater than 1.
- less peaked - higher entropy

Information Gain

It is a metric used to train decision trees. This metric measures the quality of a split.

Information gain is based on the decrease in entropy after a data set is split on an attribute. Constructing decision tree is all about finding attribute that returns highest information gain.

Information Gain = E(Parent) − {Weighted Avg}* E{children}

Step 1 - Entropy of parent.
Step 2 - Calculate entropy for children.
Step 3 - Calculate weighted entropy of children.
Step 4 - Calculate Information Gain.
Step 5 - Calculate Information Gain for all the columns. whichever column has the highest information gain the algorithm will select that column to split the data.
Step 6 - Find Information Gain recursively.

decision tree then applies a recursive greedy search algorithm in top bottom fashion to find Information Gain at every level of the tree.

Once a leaf node is reached (Entropy = 0), no more splitting is done.

Gini Impurity

$$GI = 1 - (P_Y^2 + P_N^2)$$

Same like entropy.

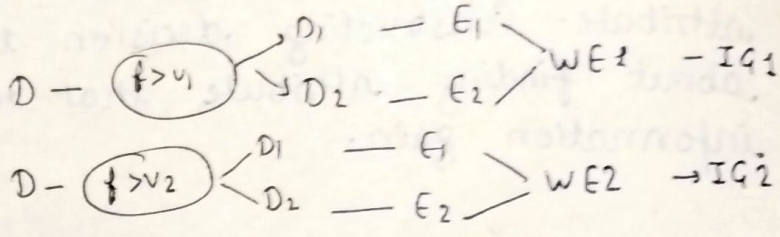
Gini is fast than entropy.
But in certain datasets, entropy give better splitting and gini may lead to overfitting.

Handling numerical data

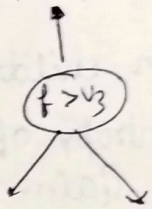
We can have n subtrees and splitting becomes difficult by grouping.

Step 1 - Sort the data on basis of numerical column.

2- Split the entire data on the basis of every value of user_rating.



3- Max {IG1, IG2, IG3 --- IGn}



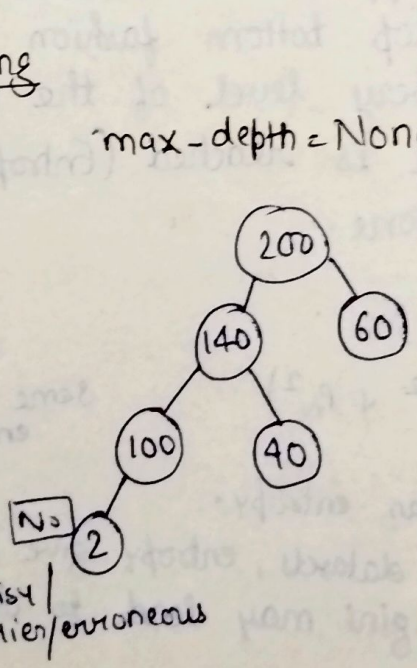
Do this recursively until you get all the leaf nodes.

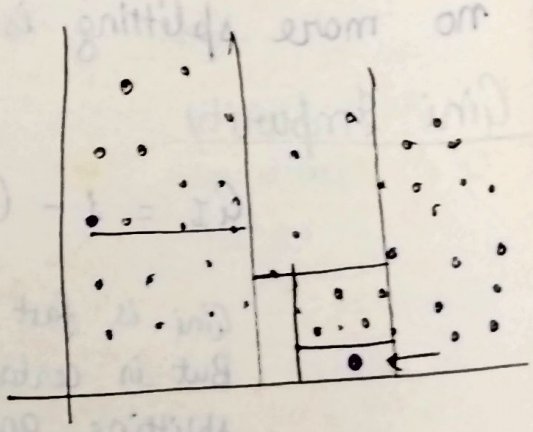
Hyperparameters

max_depth

Overfitting

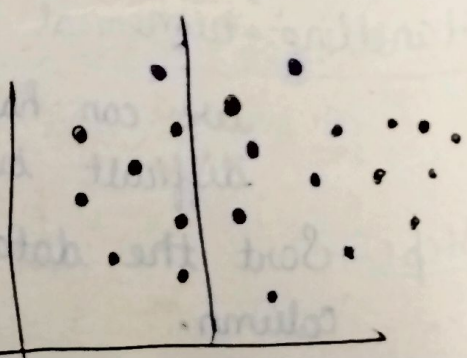
max_depth = None





Underfitting

max_depth = 1



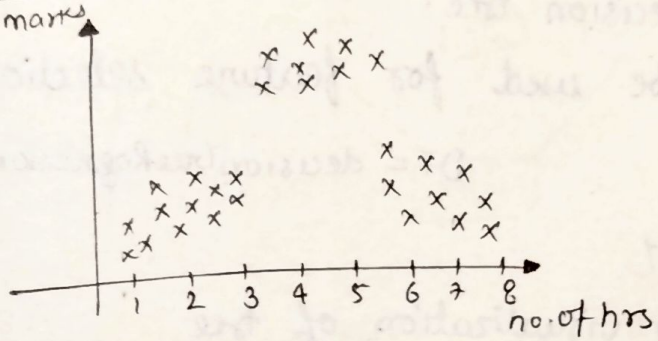
Min samples split

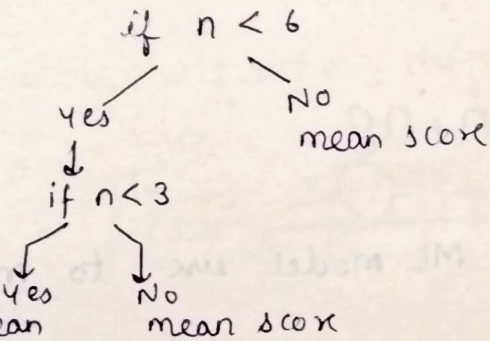
Min samples leaf

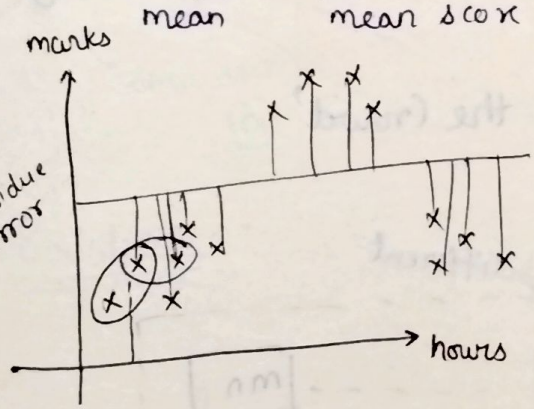
Max features

Min Impurity decrease

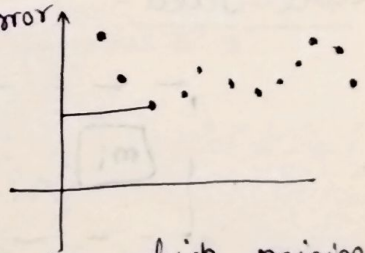
Regression Trees





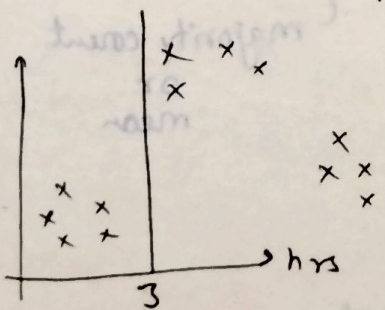


$SSE_1 = e_1^2 + e_2^2 + \cdots e_n^2$

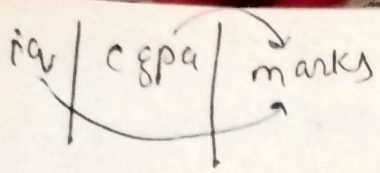


Point for which minimum error, will be first split

no. of point = 4



again repeat process



feature_importances_ how many times particular column is being used for making decision tree.

can be used for feature selection.

DT = decisionTreeRegressor( )

Dtreeviz Library

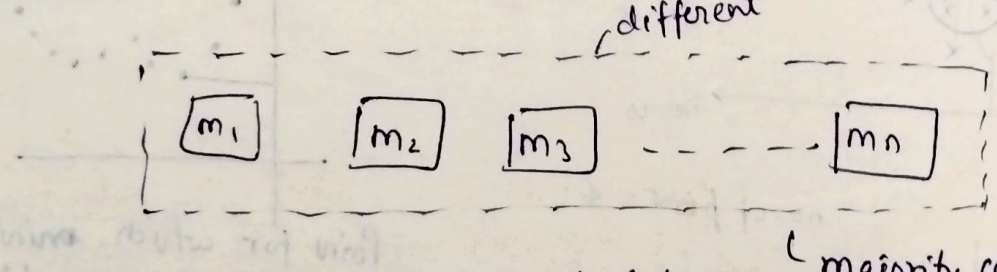
for visualization of tree

Ensemble Learning

group of ML model use to make big model.

based on fact 'Wisdom of the Crowd'

Core Idea -

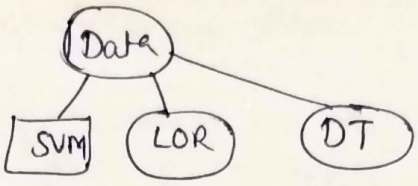


If same model, different data

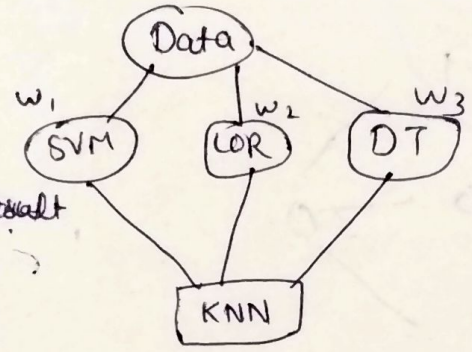
Type of Ensemble Learning

- Voting ensemble
- Bagging — Random forest
- Boosting
  - Adaboosting
  - Gradient Boosting
  - XgBoost
- Stacking

Voting



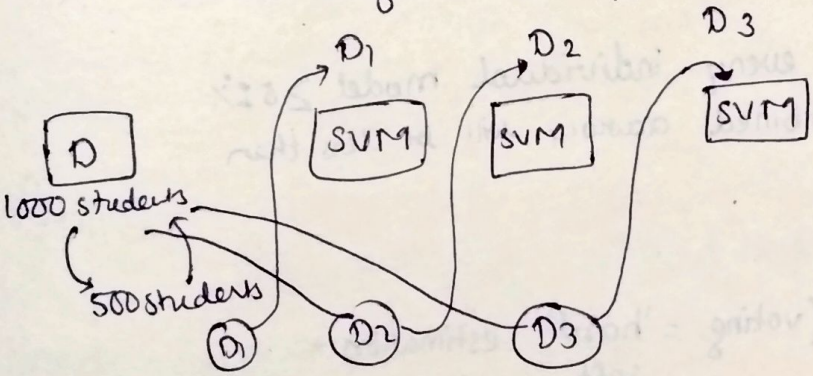
Stacking



SVM	LOR	DT	~~Result~~
1	0	1	1
0	0	1	0

Bagging – Bootstrap aggregation

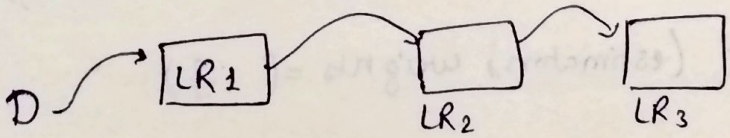
same algorithm, different data.



Random forest {Decision trees}

Boosting

boosting data i.e. ~~data~~ improve mistakes of previous ones.

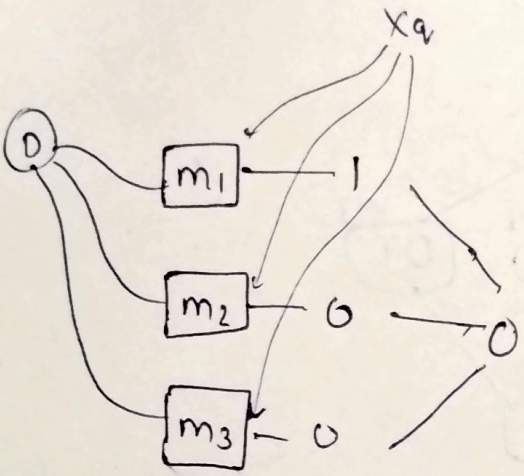


Benifits Improvement in performance

Bias-variance low

When to use? always

Voting Ensemble



Assumption -

- dissimilar and independent model perform better
- Accuracy of every individual model $\geq 51\%$
  if not, combined accuracy will be less than worst.

Voting Classifier

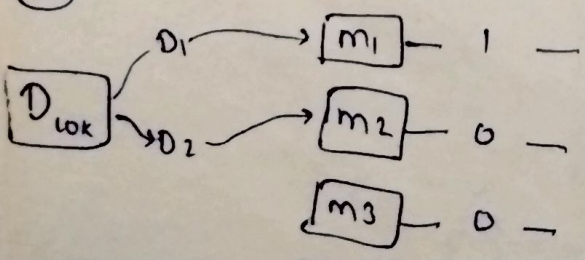
VotingClassifier (voting = 'hard' / soft estimators = / list of tuples ({name, object of model});

votingRegresor (estimators, weights = [ ])

Bagging

Bootstrapping — Aggregation

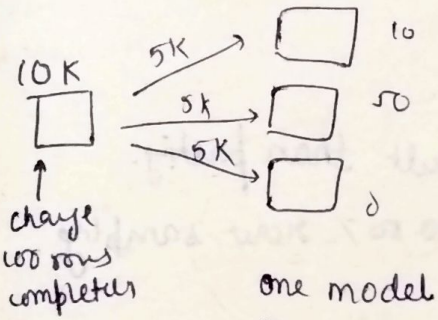
(1K)



same algorithm different data

Generally bias and variance are negatively correlated. But if you use bagging properly, you can get low bias + low variance.

DT
↳ LBHV - overfitting



one model won't get all the impurity, thus output won't change much, so variance will be low LBLV.

Types:

- Pasting
  Row sampling without replacement
- Random subspaces
  column sampling axis=1
- Random patches
  both row and column sampling

Bagging Classifiers

```
bag = BaggingClassifier(
        base_estimator = SVC(),
        n_estimators =
        max_samples =
        bootstrap =
        random_state =
      )
```

OOB - out of Bag

Statistically, around 37% rows are left, which no dt sees while training. That are out of bag rows. (cos of replacement)

unseen data.

oob_score = True

Tips:

- Bagging generally gives better result than pasting.
- Good results come around 25% to 50% row sampling mark.
- When high dimensional data then only - column sampling
- To find correct hyperparameter - GridSearchCV / RandomsearchCV.

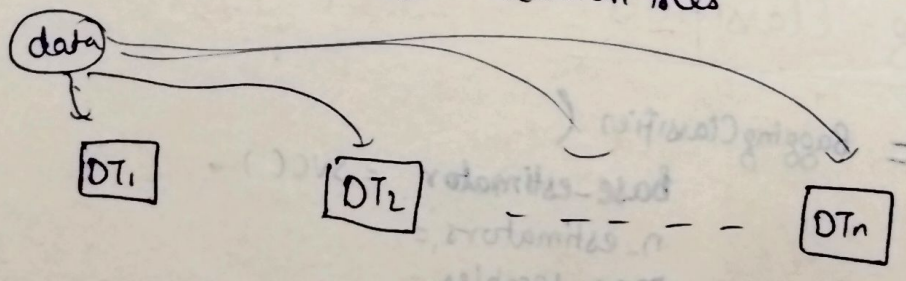
bagregressor

Random forest

It's a bagging technique

for both regression and classification problems.

Base model - decision trees



Bias Variance and Random forest

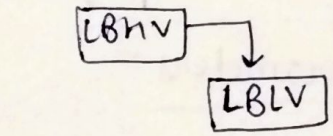
LB HV

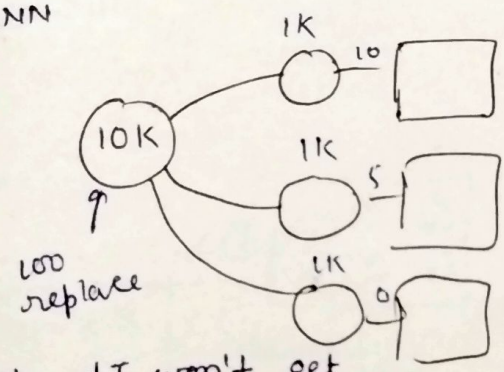
- full grown DT
- → SVM
- → KNN

HBLV

- LR

Random forest





single dT won't get all noise points. It will be distributed. Variation will be less. Thus, low variance while keeping low bias.

Bagging vs Random forest

Bagging

DT KNN SVM

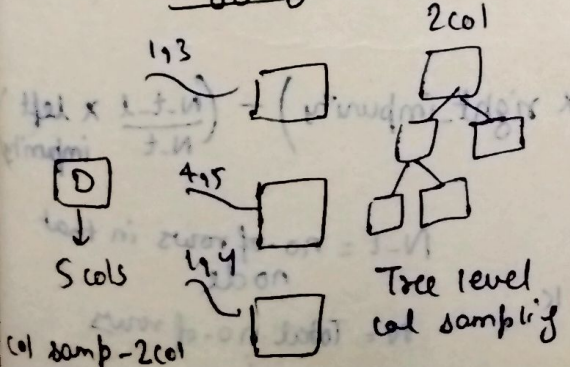
general technique in which base algorithm can be anything.

Random forest

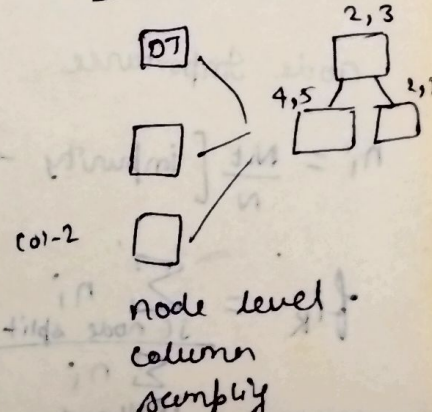
base model is Decision Trees.

what if we use DT as base model for bagging ? will it be consider as RF ? NO.

Bagging



Random forest



Before forming trees, it finalizes which column to use, in bagging.

In RF, every time node creates, it do sampling.
(More randomness)

Thus why RF performs better than bagging.

Hyperparameters

RF classifier

- n_estimators
- max features
- bootstrap
  default = True (replacement)
- max_samples

OOB Evaluation

Out of Bag Evaluation

In sampling with replacement, some rows may be duplicate, and some will never be used.
Statistically, 37% rows never used. These are out of bag samples.
These can be used for testing as validation set.

For oob evaluation, oob_score = True

Feature Importance using RF and DT

How important particular feature is:-

rf.feature_importances_

Node Importance

$$n_i = \frac{N_t}{N}\left[\text{impurity} - \left(\frac{N_{t\_r}}{N_t} \times \text{right\_impurity}\right) - \left(\frac{N_{t\_l}}{N_t} \times \text{left impurity}\right)\right]$$

$$fi_K = \frac{\sum_{j \in \text{node split on feature } K} n_i}{\sum_{j \in \text{all nodes}} n_i}$$

N_t = no. of rows in that node

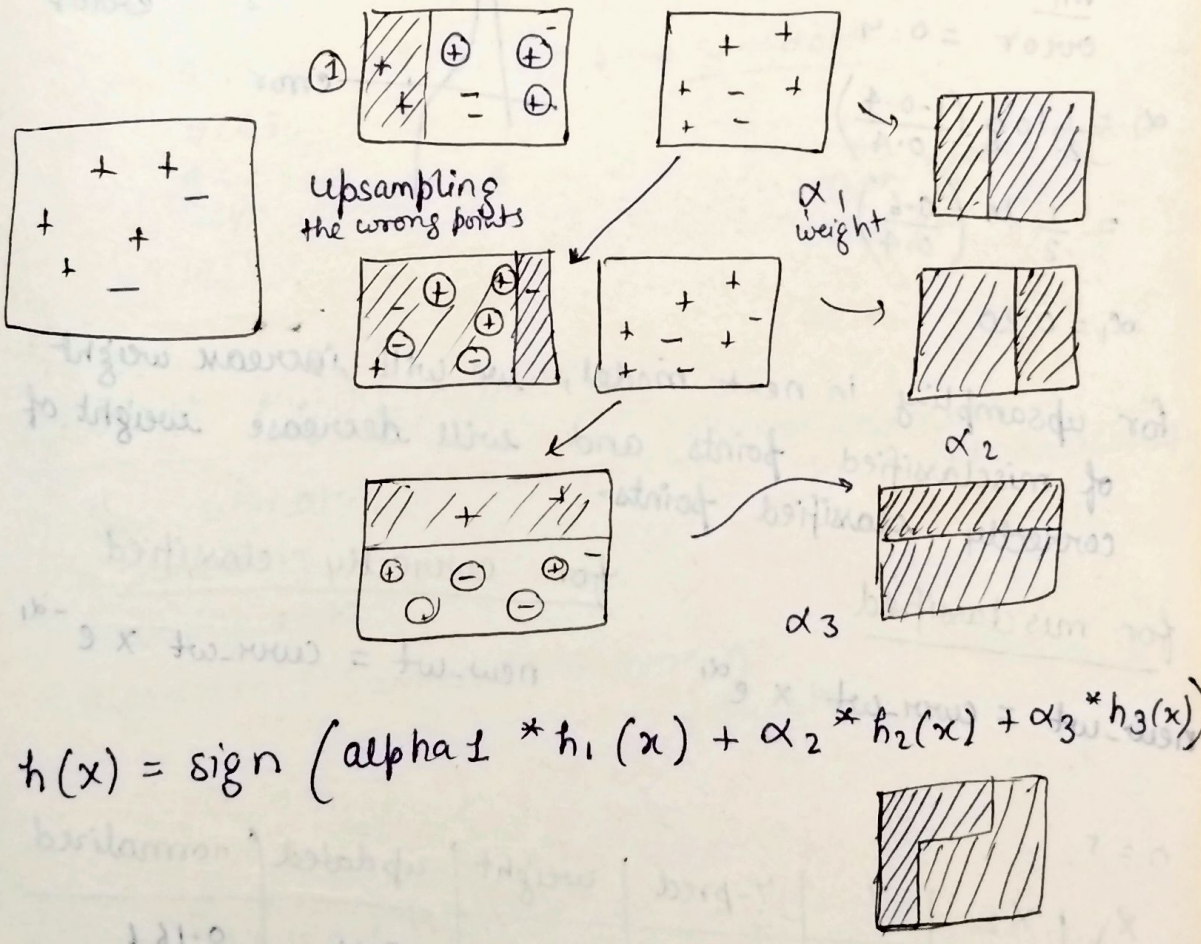
N = Total no. of rows

AdaBoost Classifier

weak learners - algorithms whose accuracy is just little more than 50%.

Decision stumps (max_depth=1)

Adaboost - stagewise additive method



$h(x) = sign\ (alpha\ 1 * h_1(x) + \alpha_2 * h_2(x) + \alpha_3 * h_3(x))$

Bagging vs Boosting

Type of model used

Target - LBLV

Bagging - LBHV
full grown dt

Boosting - HBLV
shallow dt / decision stump

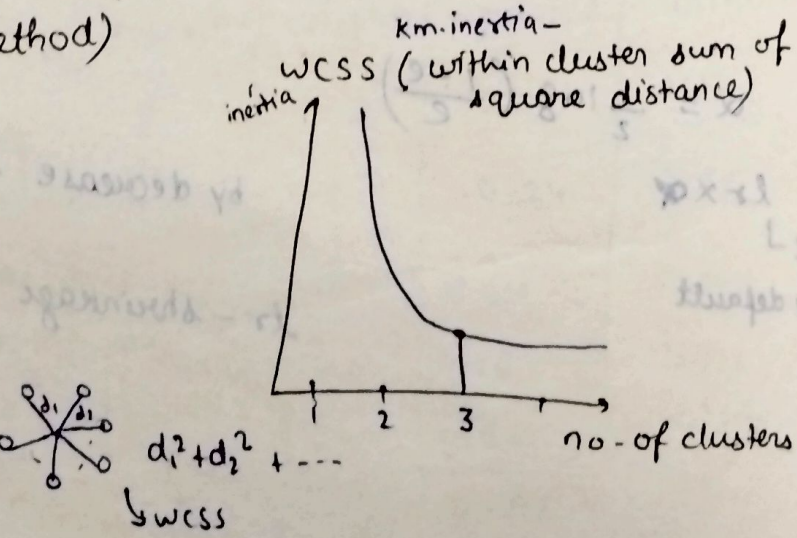
- Bagging - parallel training
  Boosting - sequential

- Weightage of base learners
  Bagging - equal
  Boosting - weights are diff to each model

K-Means Clustering

- decide n clusters
- Init centroids
- Assign cluster
- Move centroid
- Finish

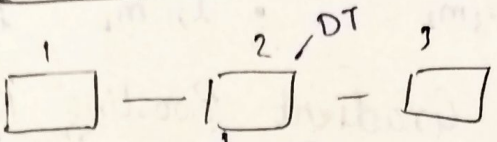
determine how many clusters need to be (Elbow method)



from sklearn.cluster import KMeans

km.inertia_

Gradient Boosting



This will always give mean

This will try to predict error given by ①

actual - predicted = pseudo_residual

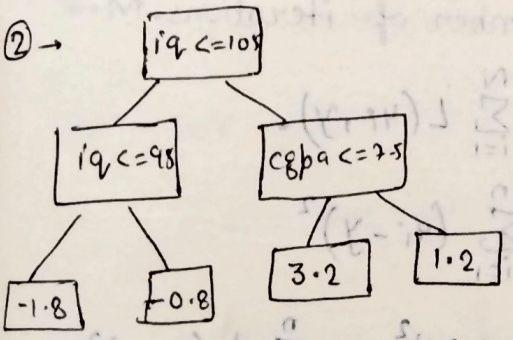
iq	cgpa	lpa	pred 1	res 1	pred 2	res 2	pred 3	res 3
90	8	3	4.8	-1.8	-1.8	-1.62	-1.62	
100	7	4	4.8	-0.8	-0.8	-0.72	-0.72	
110	6	8	4.8	3.2	3.2	2.88	2.88	
120	9	6	4.8	1.2	1.2	1.08	1.08	
80	5	3	4.8	-1.8	-1.8	-1.62	-1.62	

$$y\text{-pred}_2 = m_1 + \underset{0.1}{lr} \times m_2$$

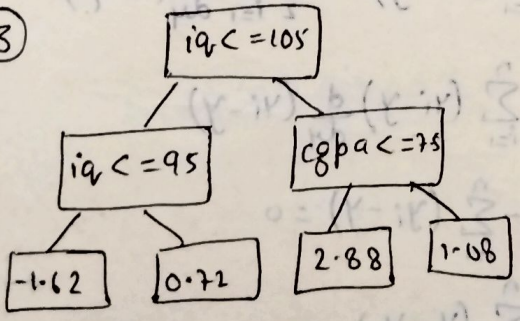
$$res_2 = actual - pred_2$$

$$y\text{-pred}_3 = m_1 + 0.1\,m_2 + 0.1\,m_3$$

② →



③



Adaboost vs Gradient Boost

- decision Stump (max_depth = 1)
- DT max_depth = 8 to 32

- $w_1 m_1$ $w_2 m_2$ $w_3 m_3$
- $lr\, m_1$ $lr\, m_2$

Mathematics of Gradient Boosting Regression

- uses concept of additive modelling

ML is nothing but a relationship b/w input and output fns. Sometimes it is linear or sometimes polynomial. But sometimes it could be so complex that can't be explained through linear and polynomial, then that composite function need to be break down to get small functions on addy that we get composite function.

This technique is additive modelling.

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$ a differentiable loss function $L(y, F(x))$, number of iterations M.

1. Initialize $f_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, \gamma)$.

$$f_0(x) = \underset{\gamma}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{n} (y_i - \gamma)^2$$

$$\frac{d f_0(x)}{d\gamma} = \frac{d}{d\gamma} \frac{1}{2} \sum_{i=1}^{n} (y_i - \gamma)^2 = \frac{1}{2} \sum_{i=1}^{n} \frac{d}{d\gamma} (y_i - \gamma)^2$$

$$= \sum_{i=1}^{n} (y_i - \gamma) \frac{d}{d\gamma} (y_i - \gamma)$$

$$= -\sum_{i=1}^{n} (y_i - \gamma) = 0$$

$$\sum_{i=1}^{n} (\gamma - y_i) = 0$$

2) for $m = 1$ to M:

(a) for $i = 1, 2, \ldots N$ compute — residual

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f = f_{m-1}}$$

$$\hat{y}_i = f(x_i)$$

$$r_{i1} = -\left[\frac{\partial}{\partial \hat{y}_i} L(y_i, \hat{y}_i)\right]_{f = f_0}$$

$$r_{i1} = -\left[\frac{\partial}{\partial \hat{y}_i} \frac{1}{2}(y_i - \hat{y}_i)^2\right]_{f = f_0}$$

$$= \left[(y_i - \hat{y}_i)\right]_{f = f_0}$$

$$= \left[(y_i - f_0(x_i)\right]_{f = f_0}$$

$$r_{11} = y_1 - f_0(x_1)$$

$$r_{21} = y_2 - f_0(x_2)$$

(b) fit a regression tree to the targets r_{im} giving terminal regions R_{jm}, $j = 1, 2 \ldots J_m$

(c) for $j = 1, 2 \ldots J_m$ compute

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$

(d) update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

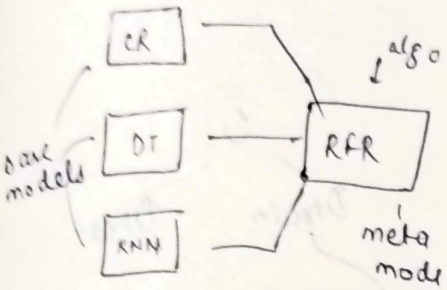
$$f_1(x) = f_0(x) + dT$$

$$f_2(x) = f_1(x) + dT_2$$

$$f_3(x) = f_2(x) + dT_3$$

(3) output $\hat{f}(x) = f_M(x)$

Stacking



1) Train your base models.
2) pred using base models.
3) Train meta model on new data.

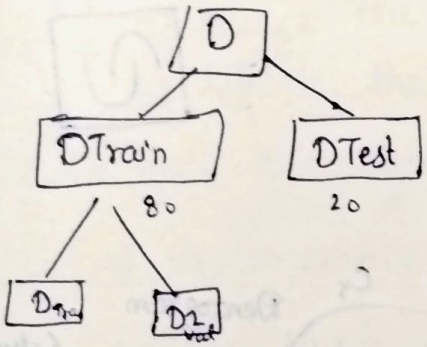
→ base models can be different

→ training on meta data

Problem - overfitting

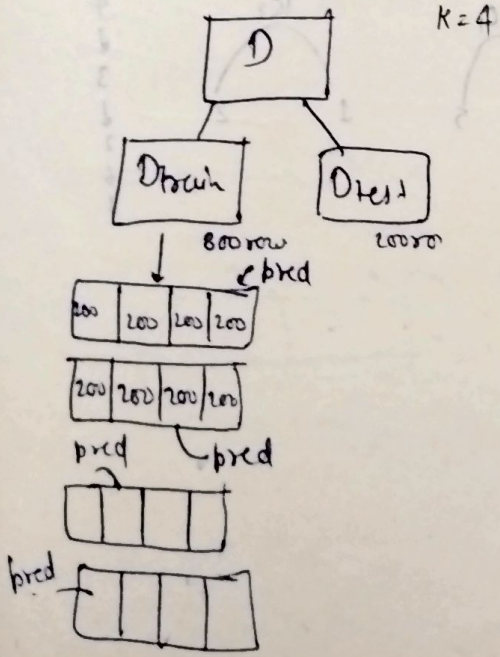
Solutions - Hold out method → Blending
K-fold method - Stacking

hold out approach - Blending



- Train 3 base models on Dtrain.
- form new data set
- Train meta model on new dataset.

K-fold approach - Stacking



- Decide K.
- Train every base model K times every time new fold for prediction.
- Train meta data with this new pred data set
- Now again retrain all base models with Dtrain.

K Nearest Neighbours

- Decide K.
  We will consider K nearest neighbours.
- Calculate euclidean distance from query point to all the training points.
- Sort distance.
- Consider K nearest distance neighbours and predict result.

```
sklearn.neighbours import KNeighboursClassifier
n_neighbours =
```

How to select K ?

- Heuristic
  - $\sqrt{n}$
  - n → observations
- Experimentation
  - by cross-validation

K is too small, overfitting

K is too large underfit

Decision Surface

mlxtend library

Limitations of KNN

KNN is lazy learning tech

training - nothing

prediction - calculate everything

distance → sort → majority

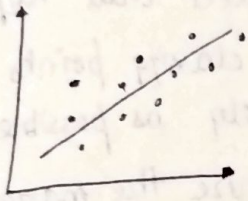
1) large datasets
2) High dimensional data

   curse of dimensionality - distance concept - not reliable
3) Outliers
4) Non-homogenious scales

   exp / salary (dominant) / hr
5) Imbalanced dataset
6) Inference and not for prediction

Assumptions of Linear Regression

1. Linear relationship b/w input and output.



2. Multicollinearity

It shouldn't be.

features should be Independent.

To check,

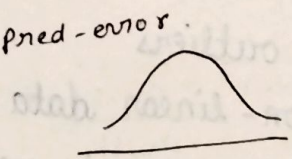
from statsmodel.stats.outliers_influence import variance_inflation_factor

if <1 - then not multicollinearity

or

sns.heatmap(df[ ]).corr( )

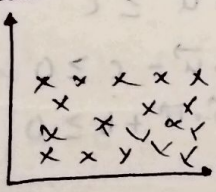
3. Normal residual



4. Homoscedasticity

having the same scatter.

residual



should be like this

spread should be uniform

5. No autocorrelation of error

no pattern should be there while plotting residual.

Handling numerical data

- Gaussian NV
- Binomial NV
- Multinomial NV
- Possion NV

XGBoost Introduction

Tianki chair

library based on gradient boosting optimizations on this

Why only gradient boosting?

- flexibility (any differentiable loss fn)
- performance
- Robust

optimizations for performance and to handle big data.

Xgboost features

concerns were - performance
- speed
- flexibility

1) flexibility -
- Cross platform
- multiple language support
- Integration with other libraries and tools
- support all kinds of ML problems.

2) speed - Parallel processing (njobs = -1)
Optimized data structure (stores in col block fashion)
Cache awareness
Out of core computing
distributed computing
GPU support

use all DS concepts

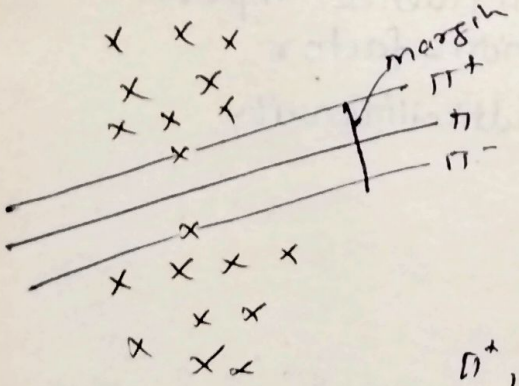
Support Vector Machine

select that hyperplane to classify points as widely as possible

maximize the margin or gap between hyperplane and points.

Margin maximizing hyperplane

Geometric Intuition



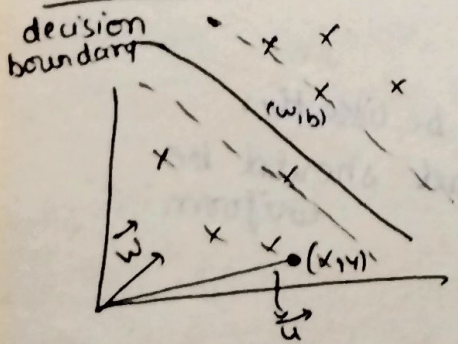
(d)

Repeat, select hyperplane, calculate π^+, π^- and select that hyperplane for which d is maximum.

π^+, π^- → support vectors

- Robust to outliers
- work on non-linear data also. → Kernels
- Both classification and regression.

Mathematics



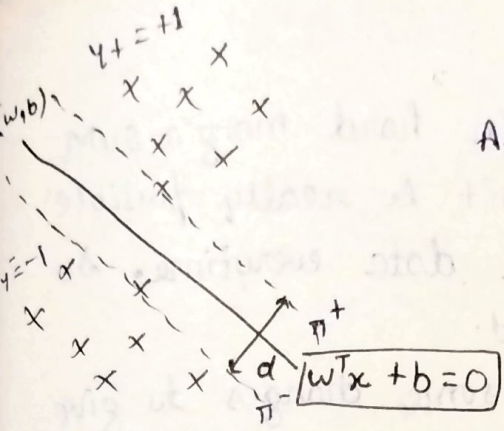
$\vec{w}$ ⊥ to all the π, π^+, π^-

decision rule →

$\vec{w} \cdot \vec{u} \geq c$

$\vec{w} \cdot \vec{u} - c \geq 0$

$\vec{w} \cdot \vec{u} + b \geq 0$



Assumption:

$\pi^+ \Rightarrow w^T x + b = 1$

$\pi^- \Rightarrow w^T x + b = -1$

for any $\vec{x_i}$

$$\hat{y} = \begin{cases} +1 & \text{if } \vec{w} \cdot \vec{x_i} + b \geq 0 \\ -1 & \text{if } \vec{w} \cdot \vec{x_i} + b < 0 \end{cases}$$

$\vec{w} \cdot \vec{x} + b \geq 1$

$\vec{w} \cdot \vec{x} + b \leq -1$

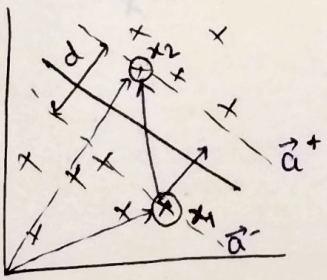
$y_i (\vec{w} \cdot \vec{x_i} + b) \geq 1$

$y_i (\vec{w} \cdot \vec{x_i} + b) \geq 1$

Constraint:

- $y_i (\vec{w} \vec{x_i} + b) \geq 1$ for all points above or below π^+, π^-
- for support vectors:

$y_i (\vec{w_i} x_i + b) = 1$



$$d = (\vec{x_2} - \vec{x_1}) \cdot \frac{\vec{w}}{\|w\|}$$

$$= \frac{\vec{x_2} \cdot \vec{w} - \vec{x_1} \cdot \vec{w}}{\|w\|}$$

$$= \frac{1 - b + b + 1}{\|w\|}$$

$$\boxed{d = \frac{2}{\|w\|}}$$

$y_i (\vec{w} \cdot \vec{x_i} + b) = 1$

$1 (\vec{w} \cdot \vec{x_2} + b) = 1$

$\vec{w} \cdot \vec{x_2} = 1 - b$

$-1 (\vec{w} \cdot \vec{x_1} + b) = 1$

$\vec{w} \cdot \vec{x_1} = -b - 1$

$$\boxed{\underset{(w^*, b^*)}{\text{argmax}} \frac{2}{\|w\|} \quad \text{such that} \quad y_i (\vec{w} \cdot \vec{x_i} + b) \geq 1}$$

Hard margin SVM