Linear Regression (single variable.

$y = mx + c$

slope or gradient.
coefficient

price = m * Area + b

depend

Inde

from sklearn import linear_model

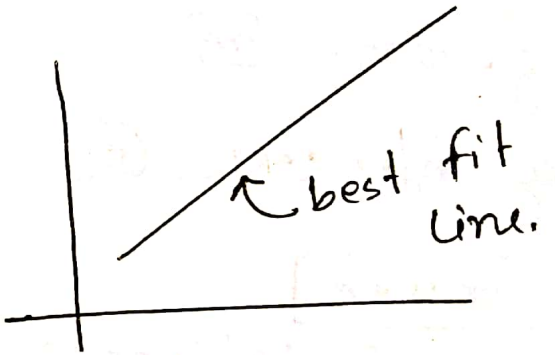
df = pd.read_csv ('homeprice.csv')

plt.xlable ('area')

plt.ylable ('price')

plt.scatter (df.area, df.price) color = red

x y



reg = linear_model.LinearRegression ()

reg.fit

reg.fit(df[['area']], df[price])

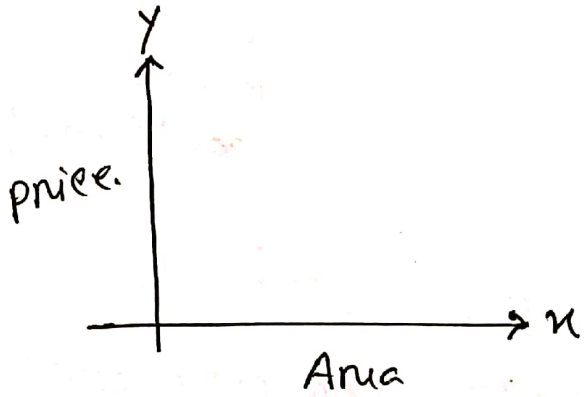
LinearRegression(copy_X=true, fit_intercept=true, n_jobs=1, normalize=false)

reg.predict([[3000]]) → [m]

> array(628715.75)

reg.coef_ → [m]

reg.intercept_ → [c]



$y = mx + c$

y = reg.predict(3000)

— x —

fill up the colum with prediction.

d = pd.read_csv('areas.csv')

p = reg.predict(d)

d[price] = p

```
d.to_csv('prediction.csv', index=False)

plt.scatter(df.area, df.price)
plt.plot(df.area, reg.predict(df[['area']]), color='blue')
```

excineise:

Given: net national income per capital

Q. predicte net income year - 2020.

Linear Regression (multiple variable)

multivariante regression.

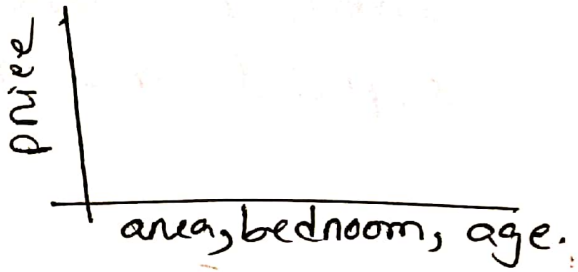
Q. given two home prices.

independent variable.

price = $m_1 \times area + m_2 \times bednoomn + m_3 \times age + b$

dependent variable.

$$y = m_1 x_1 + m_2 x_2 + m_3 x_3 + b$$



df = pd.read_csv ('homepriee.csv)

① DATA PROCESSING : Handl NA valuen

new_df = df.fillna ({'bednoomn': 0})

new_df = df.interpolate()

② LINEAR Regression :

Reg = linear_model.LinearRegression()

for multiple value.

reg.fit (df[['area', 'bednoomn', age]], df[priee])

reg. coef-

> $m_1 =$ $m_2 =$ $m_3 =$

reg. intercept

> c

x_1 → area, x_2 → bed rooms, x_3 → age

reg.predict([[3000, 3.40]])

> price = 444400.0

Application 2

gradient Descent and cost function

$y = 2x + 3$

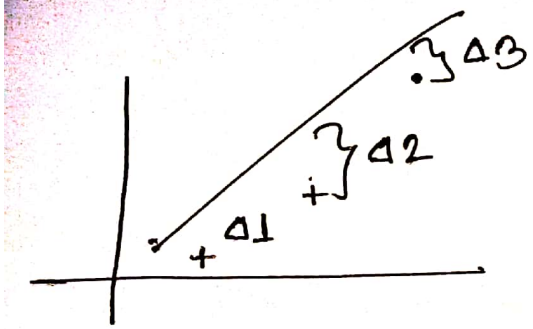
$x = [1, 2, 3, 4, 5]$

$y = [5, 7, 9, 11, 13]$

There given x, y value and find out the equation.

In plot.scatter() always we cant get bestfil line.

price = m x area + c



$$MSE = \frac{1}{n}\sum (y_i - (mx_i + b))^2$$

actual

cost function

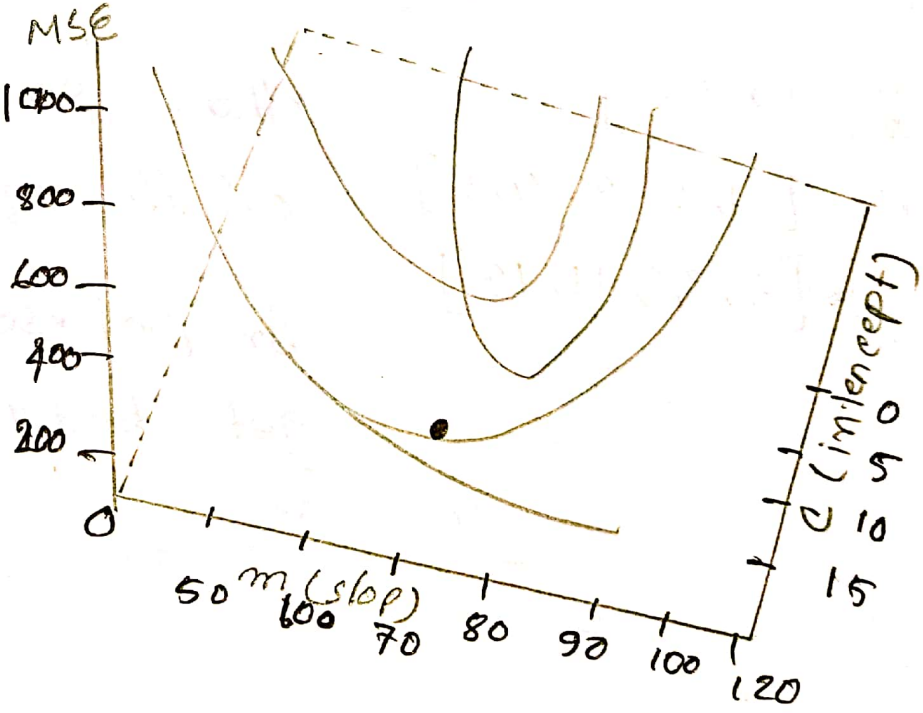
predicted

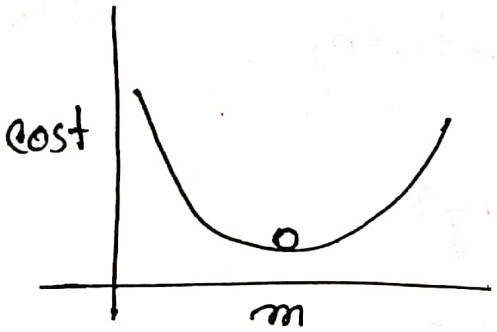
$$MSE = \frac{1}{2}\sum (\Delta i)^2$$

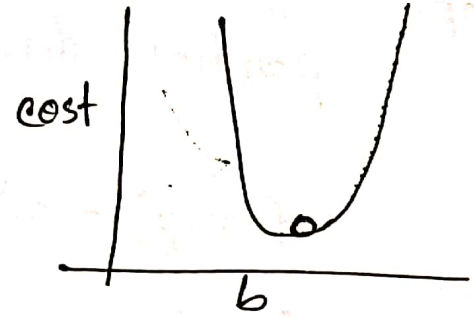
sqn them they could be negative.

MSE = mean square error.

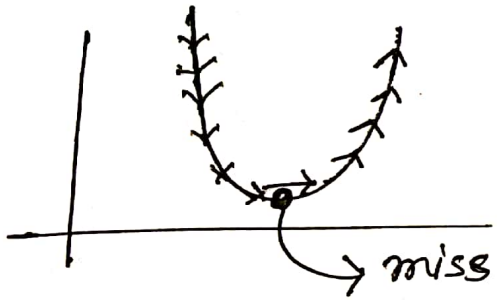
[gradient descent is an algorith that finds best fit line for given training data set]





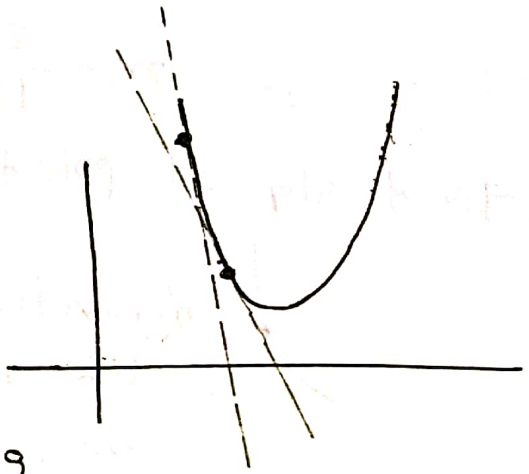


how to do this:

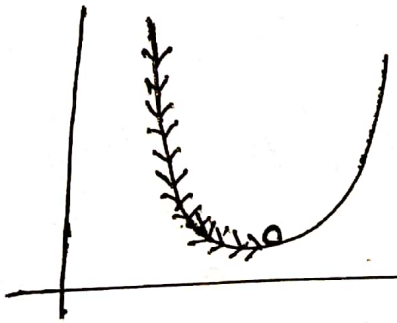


if the all steps are same on equal. (fixed size)

→ missed the global minimum.



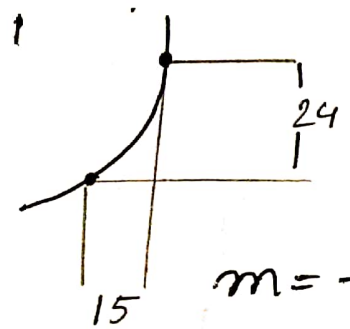
what can work:



here each steps are following the curvature.
the step size are reducing.
there each point calculate the slop.
slop also says the direction.

derivative all about slop.

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$



$$m = \frac{24}{15}$$

find slop at point:

derivativen are use in small differencee.

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

equation = $f(x) = x^2$

slop = $\frac{d}{dx} x^2$

$= 2x^{2-1}$

partial derivativen.

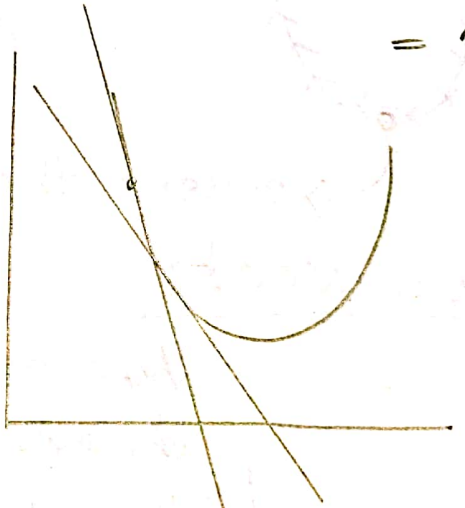
$$f(x,y) = x^2 + y^3$$

$$f'(x) = 2x + 0$$

$$f'(y) = 0 + 3y^{3-1}$$

$\frac{d}{dx} x^2$

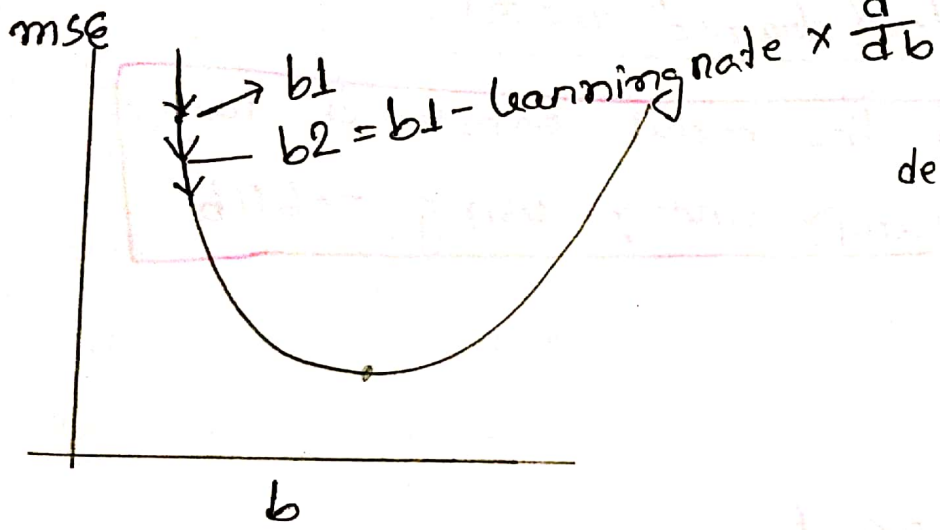
$= 2x^{2-1}$



partial derivativen.

$\frac{d}{dm}$

$\frac{d}{db}$




```
x = df['x']
y = df['y']
```

1st step

```
m = 0
c = 0
learn_rate = 0.001
n = int(len(x))
iteration = 100
```

2nd step

```
for i in range
    prediction = mx + c
```

```
def gradient_descent(x,y)
    m_curr = b_curr = 0
    iteration n = 1000
    for i in range (iteration):
        y_pre = m_curr * x + b_curr
        md = -(2/n) * sum(x*(y-ypre))
```

Save and load trained model

1) pickle.
2) sklearn joblib

diffrent: if the model constands large numpy array. using joblib

Sol.

```
import pickle.

with open('model-pickel', 'wb') as f
    pickle.dump(model, f)

with open('model-pickel', 'rb') as f
    mpdel = pickle.load(f)

from sklearn.externals import joblib

joblib.dump(model, 'model-joblib')

mj = joblib.load('model-joblib')
mj.predict(500)
```

model are saved in this file.

Categorical, Dummy variable, one hot encoding

build a prediction function to predict price of home.

① 3400 sqft area in west windson

② 2800 sqft home in robbinsville

Town	area	price
westwi	2600	58500

how shold handle text data? in numeric data.

dummy variable.

Categorical variable.

Nominal → no numric order.

→ city
gender
color.

red
green
blue

ordinal → having numric order

satisfied
neutral
dissatisfied

graduate
masters
phd

high
Low

df = pd.read_csv ('home_price.csv')

dv = pd.get_dummies (df.town) ← dummy variable.

town 1	town 2	town 3
1	0	0
1	0	0
0	1	0
0	1	0
0	0	1
0	0	1

merged = pd.concat ([df, dummies], axis = 'column')

now drop the town colum, which not use in reg. model.

dummy variable trap: → to solve this problem drop a column

when even one variable can be derived from. anest of the vari. these variable are said to be multico-linear.

```
final-marge = marged.drop(['town', 'west windson'], axis='column')
```

```
>from sklearn.linear_model import LinearRegression.
model = LinearRegression()
```

Here price in the dependent variable.

```
x = final.drop('price', axis = 'column')
y = final.price
model.fit(x,y)
```

area

nobin

```
model.predict([[2800, 0,1]])
model.score(x,y)
```

↳ compain the predi value with actual value.

```
impont LabelEncoden.
Le = LabelEncoden()
dfle = df
```

→ taken table column.

```
le.fit_fnamiform(dfle.town) = dfle.town
```

X is training dataset

X is dependent variable.

```
x = df1[['town','area']].values.
y = df1.price.
impont OneHotEncoden. categorical_featuren=[0])
ohe = oneHotEncoden()
x= ohe.fit_tranform(x).toarray()
model.fit(x,y)
mode.predict([[1,0,2800]])
```

exci: predict price of a meneedez benz that in 4 yeri old with 45000 mileage.

Training and Testing Data

A	P	Q	
a	1	10	Training 80%
b	2	20	
c	3	30	
d	4	40	
e	5	50	
f	6	60	test.
g	7	70	

plt.scatter(df['mileage'], df['sell price'])

> A clean relationship in Linear reg.

plt.scatter(df['Age'], df['sell price'])

> short of apply of Linear reg.

X = df[['milage', 'Age']]

Y = df['sell price']

```
from sklearn.model_selection import train_test_split
train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

len(x_train)
clf = linearRegression()
clf.fit(x_train, y_train)
clf.predict(x_test)
clf.score(x_test, x_test)
(classifier)
```

Logistic Regression.
(Binary Classification)

Logistic reg. is one of the technique used for classification

Linear Reg.
Home price
weather
Stock price
predicted value in continous

Classification.
- Email in spam or not
- will customer buy life insurance?

→ binary classification

- which party a person is going for vote for?
  - a. Democratic
  - b. republic
  - c. independent

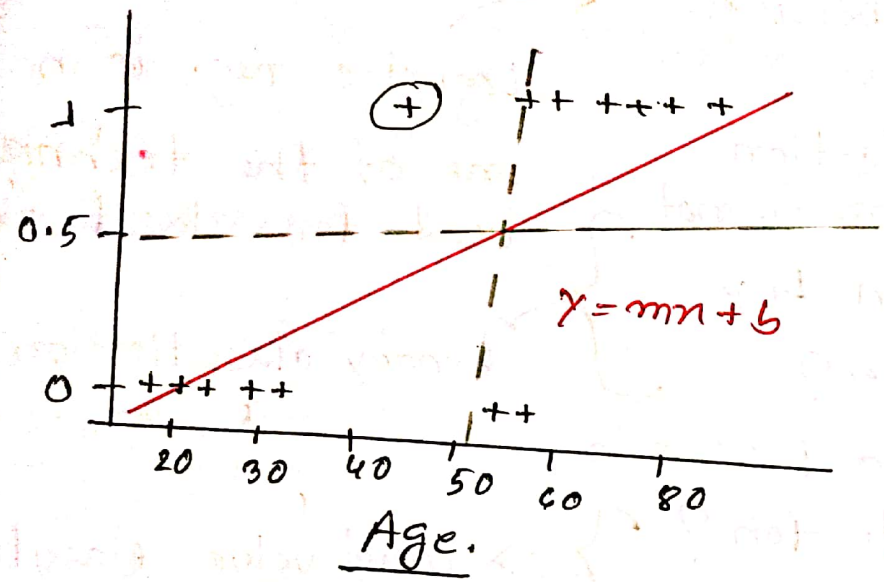
→ multivalue classifica

predicted value in categorical

Binary classification

means

yes → 1
no → 0

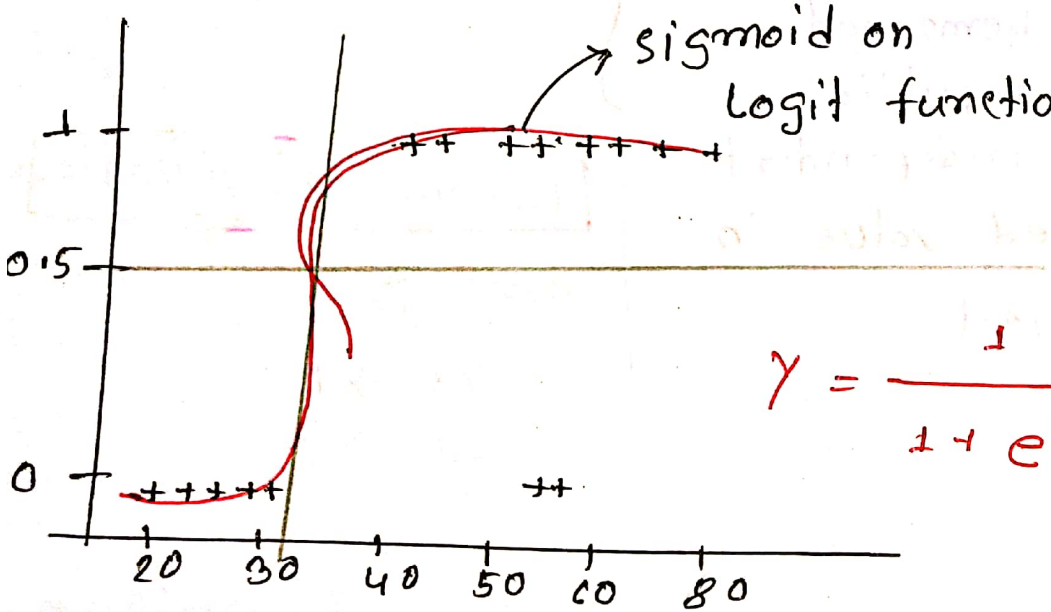


$$\text{sigmoid}(z) = \frac{1}{1 + e^{-z}}$$

$e \cong 2.71828$

↳ Euler's

sigmoid function converts input into range **0 to 1**



$$y = \frac{1}{1 + e^{-(mx+b)}}$$

plt.scatter(df[age], df[bought_insurance])

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(df[[age]], df[bough_ins], test_size=0.1)

↳ more to more score

x_test → | age |
|---|
| 18 |
| 23 |
| 40 |

x_train

from sklearn.linen_model import LogistieRegression

model = logistieRegression()

model.fit(x_train, y_train)

model.predict(x_test)?

> 0, 0, 1

not insurance.

↳ inexcesise

model.predict_proba(x_test)

>	0.6653	0.3346
	0.6061	0.3938
X	0.3918	0.6081 ✓

probability of buying insuran or not.

model.score(x_test, y_test)

> 1 ← perfect.

Logistic Regression
(Multiclass Classification)

```
from sklearn.datasets import load_digits
```

there in 1797 hand written obj
8×8

```
digiten = load_digits()
```

din (digiten) → data has numpice data

DESR, data, images, target, tenget_names.

digits.data[0] → img have actual img.

> return array length 8×8

```
plt.gray()
plt.matshow(digits.imggen[0])
```

```
for i in range(5):
    plt.matshow(digits.imge[i])
```

digits.target [0:5]

>0,1,2,3,4,0

```
from sklearn.model_selection import train_test_split
train_test_spilt (digits.data, digits.target, test_size = 0.2)
                  x             y

model = logisticRegression()
model.fit (x_train, y_train)

plt.matshow (digiten.imagen [67]) → 6
digits.target [67]
>6
model.predict ([digits.data [67]])
```

quen is how I will know where the ennon.

Coomfussing matrix.

```
y_predicted = model.predict(x_test)
from sklearn.metrics import confusion_matrix
```

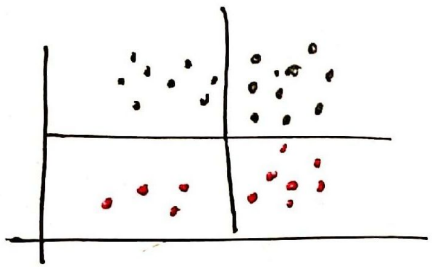
```
cm = confusion_matrix(y_test, y_predicted)
```

↓ truth. ↓ what model predicted.

```
import seaborn as sn
plt.figure(fig size
sn.heatmap(cm, annot=True)
plt.xlable('predict')
plt.ylable(truth)
```

Decision tree.

classification problem



when we get this type of data we need

independent

```
input = df.drop('salary>100k', axis=1)
target = df['salary']
```

converting text data in number.

```
import lable encoder.
le_company = lable encoder
le_job =
le_degree.
```

masurement of randomness

Company

- facebook 0/0
- ABC phan 3/3

high info gain

Degree

- Bachelon 4/4
- masten 0/2

Low info. gain.

~~inputn[company_n] = le_company.fit~~

```
inputn[company_n] = le_company.fit_transform(input[comp])
job_n
degree_n
```

```
inputs_n = inputs.drop(['company', 'job', 'degree'], axis='column')
```

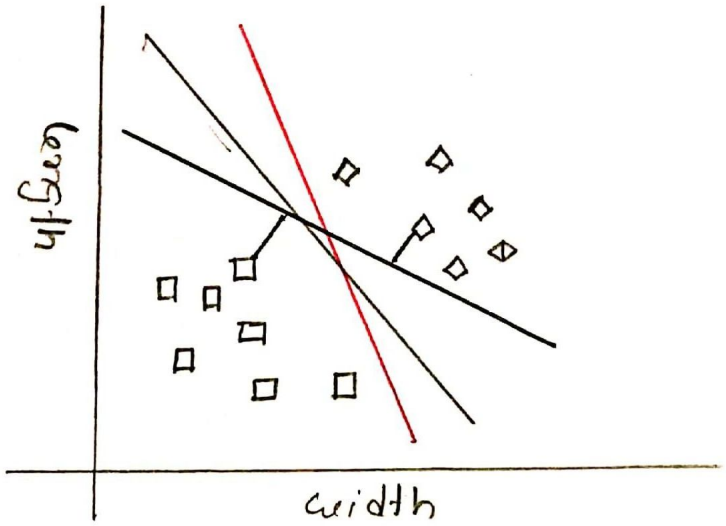
ready to train classifier.

```
from sklearn import tree
model = tree.DecisionTreeClassifier()
model.fit(inputs_n, target)
model.score(input_n, target)
```

↑ cause of same dataset.

```
>1
model.predict([2,2,1])
>0
```

SVM (~~Staff~~ classification Algorithm:)



```
from sklearn.dataset import load_iris
iris = load_iris()
dir(iris)
> data, feature_names, target, target_name.
iris.feature_names
  sepal. length (cm)
         width
  petal  length (cm
         width

~~df=pd.head~~
df=pd.Dataframe(iris.data, colums=iris.feture_names)
>
df['tenge'] = iris.target
>
iris.target_names
> setosa, versicolor, virginica
    0          1          2
df[df.target==2].head()
```

```
df['flower_name'] = df.target.apply(lambda x: iris.target_names[x])

df0 = df[df.target==0]
df1                  1
df2                  2

df2
>
plt.scatter(df0['sepal length'], df0['sepal width'])
plt.        df1                  df1
```

x = sepal length
y = sepal width.

df0 petal length df0 petal width
df1 df1

```
import train_test_split

X = df.drop(['target', 'flower_name'], axis='columns')
Y = df['target']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)

from sklearn.svm import SVC
```

model = SVC() ⇒ gamma, kernel

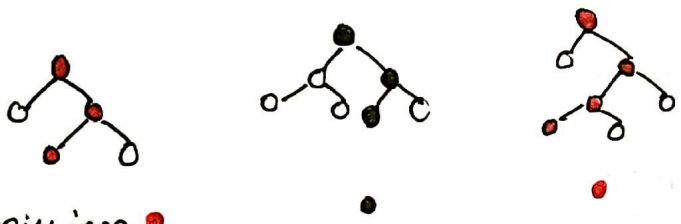
model.fit (x_train, y_train)

model.~~test~~ score (x_test, y_test)

Random forest Algorithm

Classification and regression

when there are so many decission tree then there are a random forest.



Decission ● ● ● result in ●

```
digiten = load_digiten()
din(digiten)
plt.gray()
~~dtgiten.image[1]~~
plt.matshow(digiten.image[:])
```

```
df = pd.DataFrame(digits.data)
df.head()
~~dfgiten~~.[target] = digits.target.
```

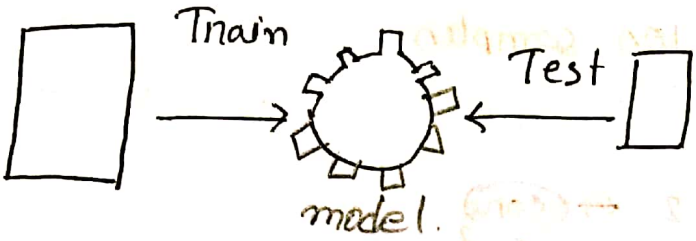
AI Auto regressive Language model.

```
x = df.drop(['target'])
y = digits.target.
impont train-test-spilt
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
len(x_test)
from sklearn.ensemble impont RandomforestClassifien
model = RandomForestClassifien()
model.fit(x_train, y_train
```

Date: ____/____/____

kFold cross validation.

cross validation in a technique, which allows you to Answare the exac

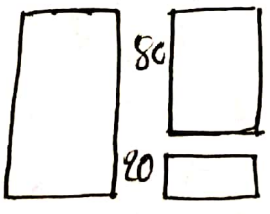


There many several ways you perform thin training step.

option-1 :
use all available data for training and test on same data set.

exac
100 ques, and make the paper on 100 ques.
thin test in not suitable.

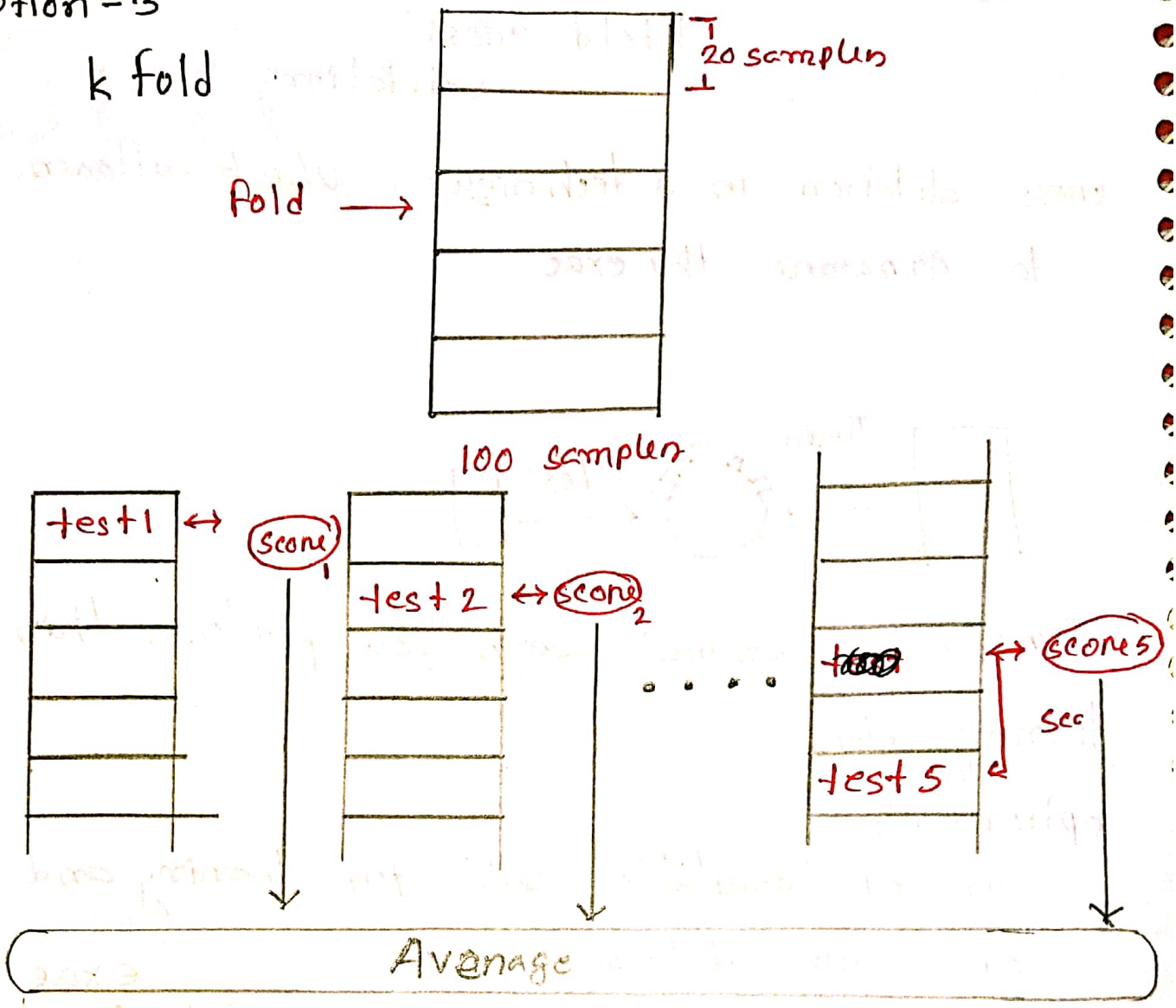
option-2 :
split data into train and test.



all the type of ques are not in train

option - 3

k fold



giving variety of sample. to model, and taking individual score. and avenaging them.

Healthcare Date: ...../...../.....

```
digits = load_digits()
train_test_split(digits.data, digits.target, test_size=0.3)
```

① In logisticRegression score 95%.

In svm cone 40%.

In Random forestClassifien score 97%.

This model score value will chang. because of the chagen of test size/on spilt

```
from sklearn.model_selection import kFold
kf = kfold(n_splits=3)
kf
```

↳ How many folds.

```
for train_index, test_index in kf.split([1,2,3,4,5..9]):
    print(train_index, test_index)
```

↳ method. ↳ iteration

divided in to 3 folds.

9/3 = 3

[3 4 5 6 7 8] [0 1 2]

[0 1 2 6 7 8] [3 4 5]

0 1 2 3 4 5] [6 7 8]

```
def get_score(model, X_train, X_test, Y_train, Y_test):
    model.fit(X_train, Y_train)
    return model.score(Y_train, Y_test)
```

```
from sklearn.model_selection import StratifiedKFold
folds = StratifiedKFold(n_splits = 3)
```

↳ better k fold.

```
score_logi = []
score_svm = []
score_RanFor = []
```

real datasets.

folds. ↑ [1, 2, 3 .... 9]

```
for train_index, test_index in ~~f~~.split(digits.data):
    X_train, X_test, Y_train, Y_test
        = digits.data[train_index], digits.data[test_index]
        /digits.target[train_index], digits.target[test_index]
```

```
score_logi.append(get_score(LogisticRegression(), X_train,
                            X_test, Y_train, Y_test))
score_svm.
score_rf.
```

Date: ...../...../.....

(M) Anom sklearn.model_selection impont cross_val_score

cross_val_score(logisiticRegression(), digits.data, digits.target)

exc.

1. use iner flower datasets.
2. use diffrent elasiffire.
3. mesure the seore.

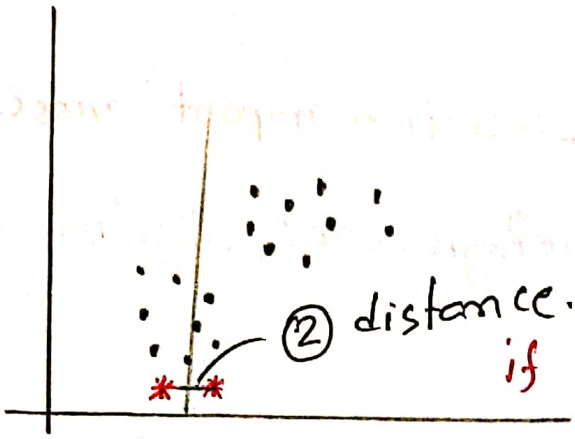
k Means clustering Algorithm.

step-1) scaleing
step-2) find centri
step-3) SSE + Snapl

machine learning algorithms are three main categori

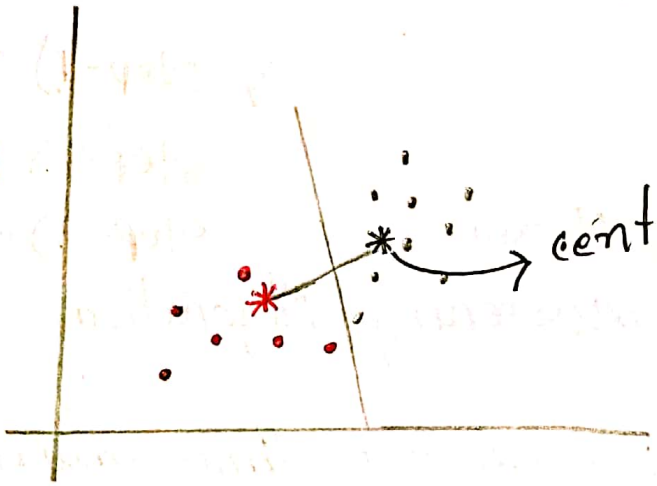
1. supervised → given data set, clann lavel on tenget.
2. unsupervised → set of feature, dont know about clann lable / tenget
3. rein forcement learning.

to find the useful prediction. } try to identyfy the under lining structure. in that data.
try to find ~~data~~. cluster.



if $k=2$ any random place.

imperfect clunky clusters and now we try to improve. improve this cluster.



③ if the red point is more near to set the cluster red.

1. starts with k centroids by putting them at random place $k=2$
2. compute distance of every point from centroid and clusters them accordingly.
3. Adjust centroids so that they become center of gravity for give cluster.

Date: ...../...../.....

4) agin re-cluster every point based on their distance.

5) Agin adjust centroids.

6) recompute cluster

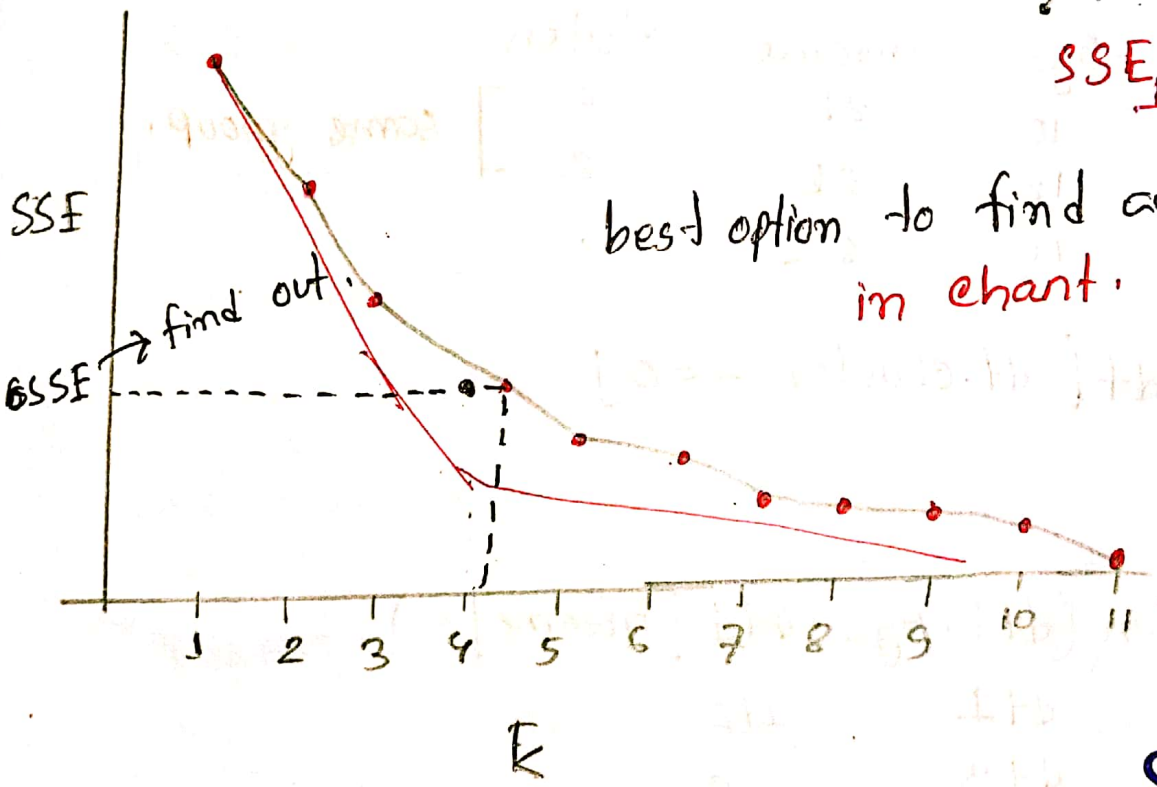
Aften final cluster, most important point here, need to supply k.

Q. what in the good number of k?

A. there in a techneque. called elbow method.

Thy to compute sum of square error (SSE)

$SSE_j = \sum dis(x-c)^2$



best option to find an elbo in chart.

```
df= Pd.read_csv(income_.csv)
plt.scatten( df[Age], df[income])
```

↳ 3 clusten.

```
from sklearn.clusten import kMeann
km = kMeann(n_clusten = 3)
km.fit_predict( df[[Age, income]])
```

y_predicted.

[2,2, 0,0, 1,1 . . . . 1, 0,0,0,0, 2,2,0]

2 0 1 ← three clusten.

```
df[cluten] = y_predict
```

Name	Age	income	cluten	
x	10	2k	2	] same group.
y	12	5k	2	
z	14	8 lc	0	

```
df1 = df[df.clusten == 0]
df2
df3
plt.scatten (df1.Age, df1[income] )
```

df2 df2

df3 df3

Date: ____/____/____

there Age and income in not in scale

```
from sklearn.preprocessing import MinMaxScaler.

scaler = MinMaxScaler()

scaler.fit(df[[income]])

df[income] = scaler.transform(df[[income]])

df (the income now scaled 0 to 1)

scaler.fit(df.Age)

then fit the data sets into kMeans

km = kMeans(n_cluster = 3)

y_predicted = km.fit_predict(df[[Age, Income]]

ypredicted

df[cluster] = y_predicted.

df = df.drop(cluster)

Clustering Aging.

df1 = df[df.cluster == 0]

2

plt.scatter(df1.[Age], df1[income])

df2 df2
```

How to get the centniodn.

kM.clusten_centern-

$$\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \end{bmatrix}$$

plot on centnoid.

plt.scatten (~~kM~~ (kM.clusten_centern-[:,0], kM.clusten_centen_[0:,1])

k_nag = nange (1,10) ← Assuming

SSE = [ ]

fon k in k_nag:

kM = kMeang (n_clustenn=k)

kM.fit (df [[Age, income]])

SSE.append (kM.inentia_)

SSE

plot_ing SSE And k

plt.xlable ('k')

plt.ylable (SSE)

plt.plot (knang, SSE)

Date: ....../....../......

predict

Nive Bayes

Application: spam email, hand writting, weather pre
face dection, News articale.

it worken independent variable. sex, Age, class

```
df= pd.read-csv(titanic.csv)
df.drop([passengerId, name, class, parch, Tic])

target = df.survived
inputs = df.drop('survived')

[dummies = pd.get-dummies(input.sex)
inputs = pd.concat([inputs, dummies])
inputs.drop(Sex)
inputs.columns[inputs.isna().any()]

[inputs.age = inputs.age.fillna(inputs.Age.mean())

x-train, x-test, y-train, y-test =
        train-test-split(inputs, target, testsize=0.2)
```

create naive bayen model.

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(x_train, y_train)
model.score(x_test, y_test)
```

part-2

Email spam

ham → good massage.
spam

```
df = pd.read_csv('spam.csv')
df.groupby(category).describe()
```

category and massage → numbers.

```
df[spam] = df[category].apply(lambda x: 1 if x == spam
                                        else 0)
df.
```

→ massage(text

```
x_train, x_test, y_train, y_test =
        train_test_split(df.massage, df.spam)
                                test_size = 0.25
```

countVectonize → the massage colom convent into numbenn.

each of thin doc. find the unique wond. → compun.

Thin in the first doc	Thin doc in the second doc	And thin in the Third one.
and, doc, first, in, one, second the, third, thin ↑ unique		Is thin the first doc.

3 typen of classifien.

1. Bennoulli → fetune (0,1)
2. Multinomial → discnete data (movie nating.
2. gaussian. → nonmal distribution (bell cunve)

from sklean.feature_extnaction.text impont countVectonizen.

```
V= CountVectonizen( )
x-tnain count = V.fit_tnansfonm(Xtnain.valuen)
x_tnain_count.toannay( ) [ : 3]
```

↳ Num of unique wonds.

from sklearn.naive_bayes import MultinomialNB

model = multinomialNB() ← for classification

model.fit(x_train_count, Y_train)

↳ Email converted into a num. metric

~~super vector~~

Email = ['first text', 'second text']

email_count = v.transform(email)

model.predict(email_count)

[0.] ← 2nd email is spam.

In pipeline where can define a transformation

pipeline is super useful.

from sklearn.pipeline import pipeline.

clf = pipeline([↗ convert the text into vector

('vectorizer': countVectorizer()),

(nb, MultinomialNB())

])

clf.fit(x_train, Y_train)

Date: ...../...../.....

Hyper para

How to choose the best model youn given machine learning problem?

* can be classify our model, by using kfold and loop but it in not convienient.

```
-from sklearn.model_selection import GridSearchCV
clf = GridSearchCV(SVM.SVC(gamma = 'auto'),{
    'C': [1, 10, 20],
    'kernal': ['rbf', 'linear']
}, cv= 5 }, return_train_score = False)

clf.fit(iris.data, iris.target)
clf.cv_results_
```

paramiter grid. — for hyperparameter tuning. — permutation and combin of this paramiter.

→ tabular views

```
df = pd.DataFrame(clf.cv_results)
df [['param_c', 'param_kernal', 'mean_test_score']]
din(clf)
clf.best_score_
    .best_params_
```

← best of model.

```
from sklearn.model_selection import RandomizedSearchCV
RS = RandomizedSearchCV(svm.SVC(gamma='Auto'),{
    'C' = [1,10,20],
    'kernal' = [rbf, linear]
},
cv = 5,
return_train_score = false,
n_iter = 2
)
rs.fit(iris.data, iris.target)
pd.DataFram(rs.cv_results_)[[pname, pram|kernal, mean_te
```

not try even single permutation and combinati of paramiters.

cross val.

try on random parms.

num of ittration.

Chossing best model.

```
import SVM
import RandomForestClassifier
import LogistieRegression.

Model_Parms = {
    SVM: {
        model : SVM.SVC( ),
        parms : {
            C : [1, 10, 20]
            kernal : [rbf, linear]
        }
    }
```

Date: ...../...../.....

```
'random_forest': {
    'model': RandomForestClassifier(),
    'Prams': {
        'n_estimator': [1,5,10]
    }
},
Logistic_Regration:{
    model : logisticRegression()
    'prams' : {
        'c' : [1,5,10]
    }
}
}
```

~~Scores = []~~
~~for model_name, mp, in model_~~ pram

```
scores = []
for model_name, mp in model_params.items():
    clf = RandomizedSearchCV(mp['model'], mp['params'], cv=5,
                             return_train_score=False)
    clf.fit(iris.data, iris.target)
    scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })
df = pd.DataFrame(score, columns=['model', best_score,
                                  best_params])
```

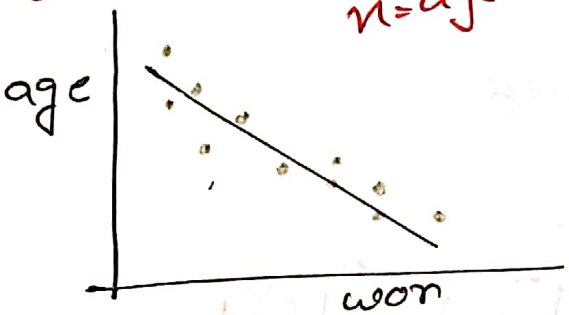
Date: ____/____/____

Regularization

L1 and L2 are use in overfitting issue.

How a model in overfit and can be addressed on accuracy can be improved.

① under fitting



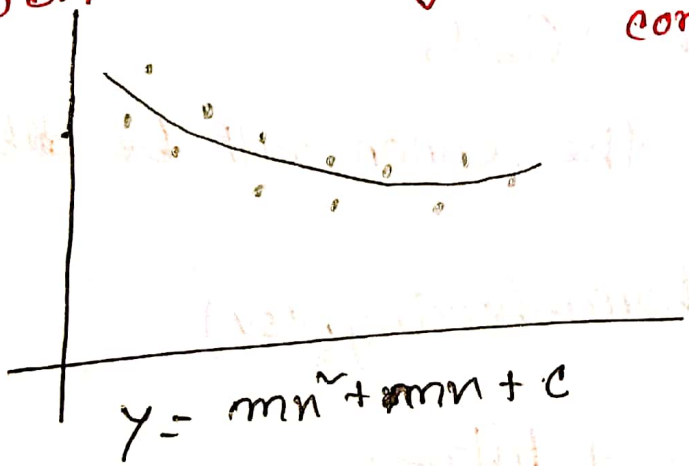
$y = mx + c$

② over fitting

$y = mx^4 + mx^3 + mx^2 + mx + c$

thin equ. in to much complicated.

③ Balanced fitting.



$y = mx^2 + mx + c$

thin in the Ballance of 1 and 2

How to reduce overfitting.

In overfitting eqn.

$$y = \theta_0 + \theta_1 x age + \theta_2 x age^2 + \theta_3 x age^3 + \theta_4 x age^4$$

try to make θ_3, θ_4 almost close to zero.

$$y = \theta_0 + \theta_1 x age + \theta_2 x age^2$$ [shrink the parameter]

we did this by using mse

L1

$$mse = \frac{1}{n} \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} |\theta_i^2|$$

$$h_\theta(x_i) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3$$

if the θ in biggen the ennon will be biggen.

```
df = pd.read_csv(Melbourn-housing.csv)
df
```

Data cleaning + fillna.

```
df.isna().sum()
cols_to_fill_zero = [popentycount, Distant, Bednoom2,
                     Bathnoom, car]
```

Date: ...../...../.....

Use categorical feature using dummy variable

```
x = df.drop(price, axis = 1)
y = df.price
train_test_spilt(x, y, test_size = 0.3)
impont LinearRegression
reg = LinearRegression()
Reg.fit(x_train, y_train)
reg.score(test_x, y_test)  under fitting
> 0.13
reg.score(x_train, y_train)  overfitting.
> 0.68
from sklearn impont linear_model
Lasso_reg = linear_model.Lasso(alpha = 50, maxiter=50,
                               tol = 0.1)
Lasso_reg.fit(x_train, y_train)
Lasso_net.score(x_test, y_test)
```

L1