Machine Learning notes
Supervised Learning

Unsupervised learning- reading the patter. Able to cluster based on something com

Reinforcement learning- training a dog

SUPERVISED LEARNING
FEATURES:

1. **QUALITATIVE-** CATEGORICAL DATA- finite no of categories. Eg Gender, country
    a. No inherent order
    b. Called **nominal data= No inherent order** use **One hot encoding**
    c. One-Hot Encoding:
    d.

USA	**USA**	**India**	**Franae**

    e. Ordinal Data- which has some order. Eg- Older, Younger. Sad, Happier, Very happier. Can mark them no
2. **QUANTITATIVE FEATURES-**
    a. DISCRETE - LENGTH
    b. CONTINUOUS- TEMPERATURE

TYPES OF PREDICTION IN SUPERVISED LEARNING

1. **CLASSIFICATION-** PREDICT DISCRETE CLASSES.
    a. Binary- Hot Dog or Not Hot Dog, Positive/Neg
    b. Multiclass- its cat,lizard, dolphin. More than 2

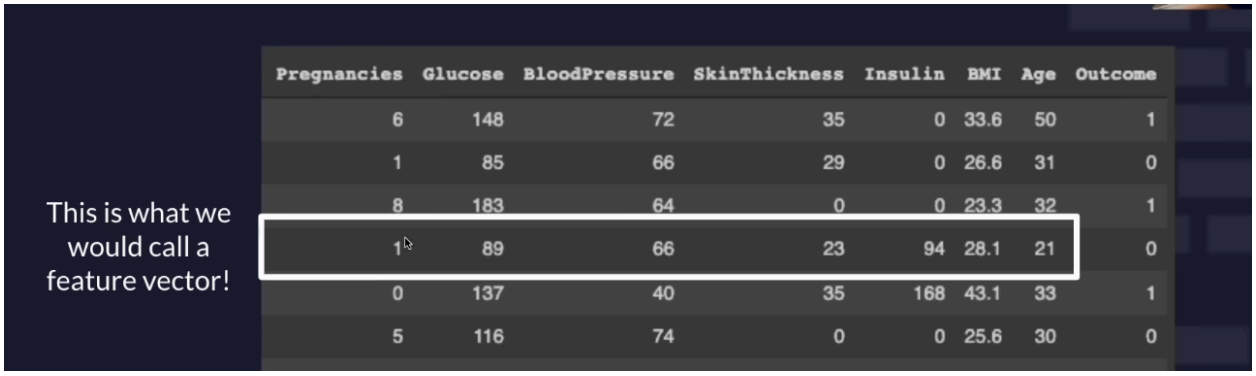
2. **REGRESSION-** predict continuous values
    a. Price of ethereum tomorrow, price of house. Predict to close value

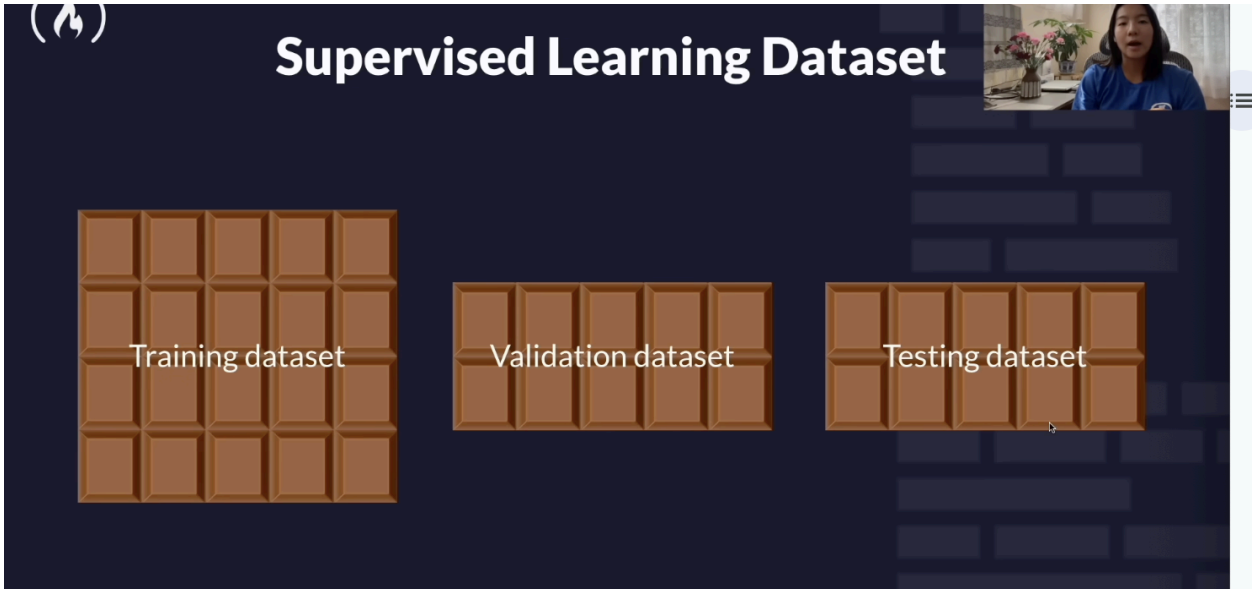
---

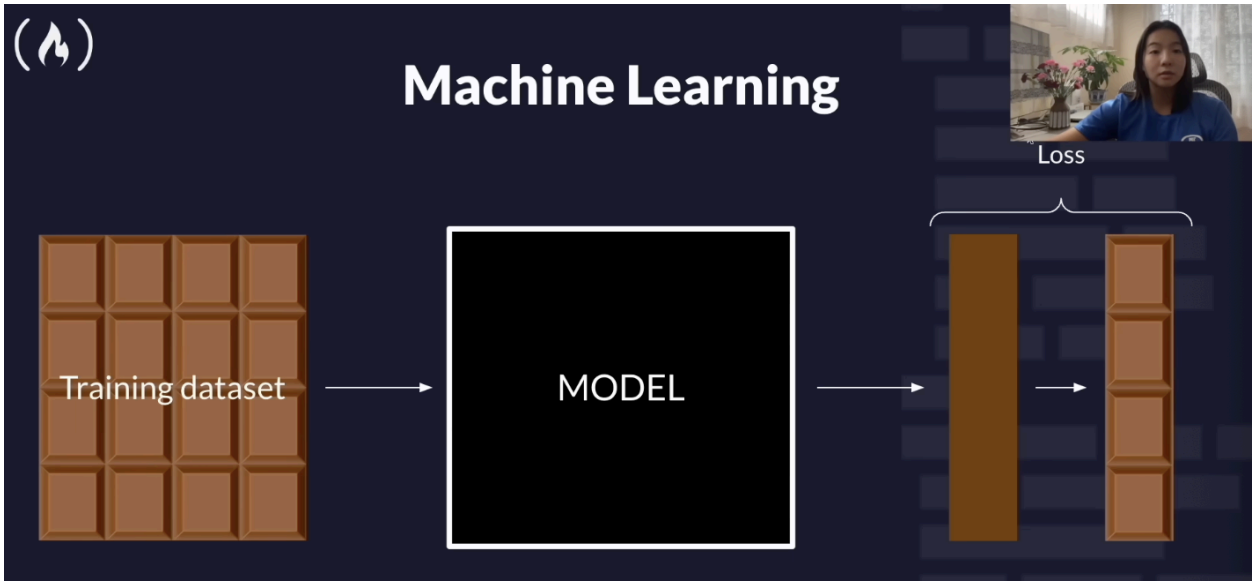
MODEL

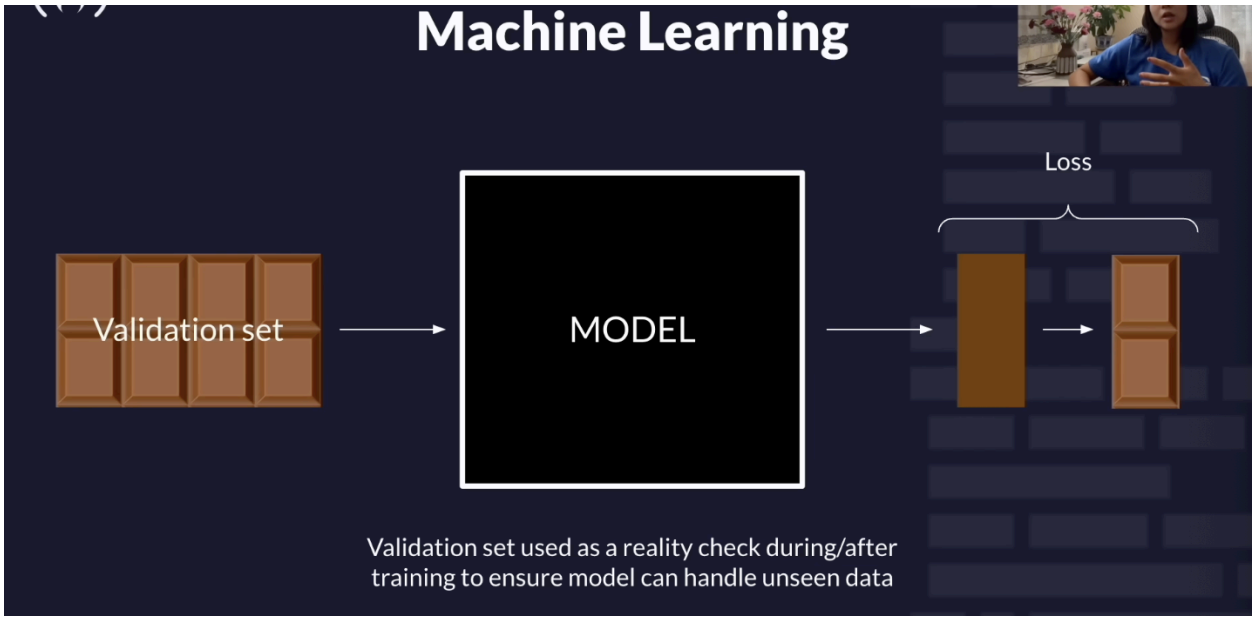
1. **EACH COL REPRESENT DIFF FEATURE**
2. **Feature vector**

out

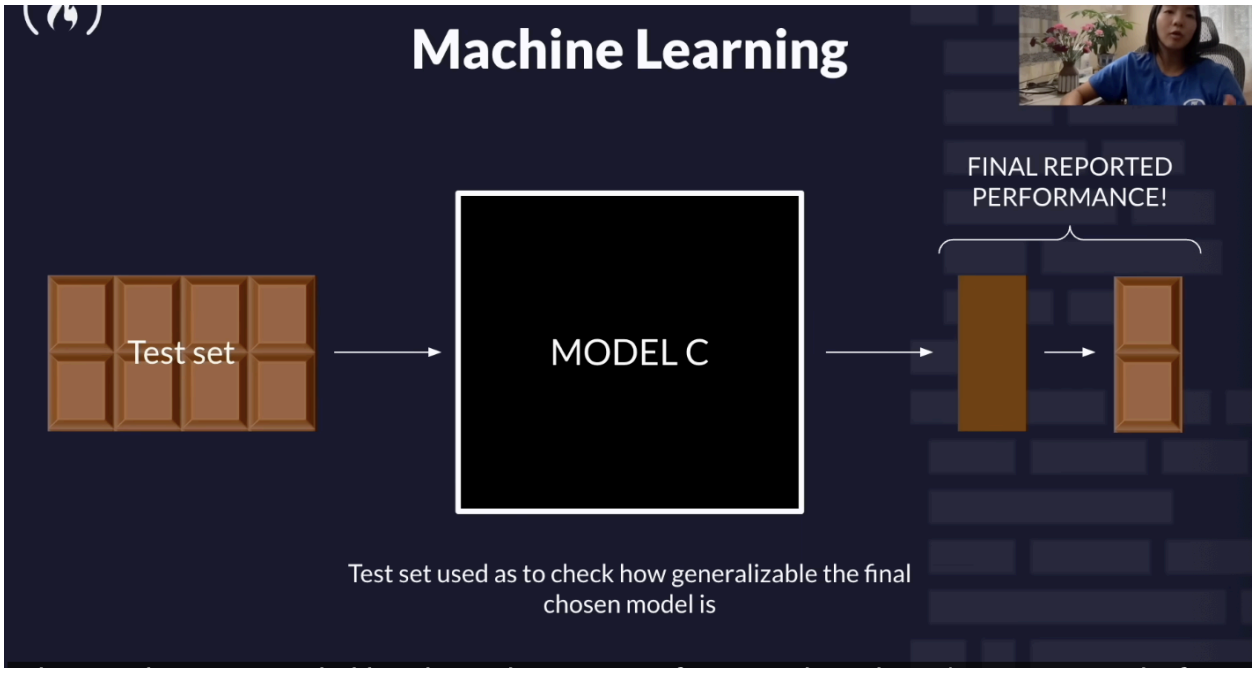


3. **Target for feature vector- what we want to predict [Outcome]**
4. 

5. 

6. 

Loss

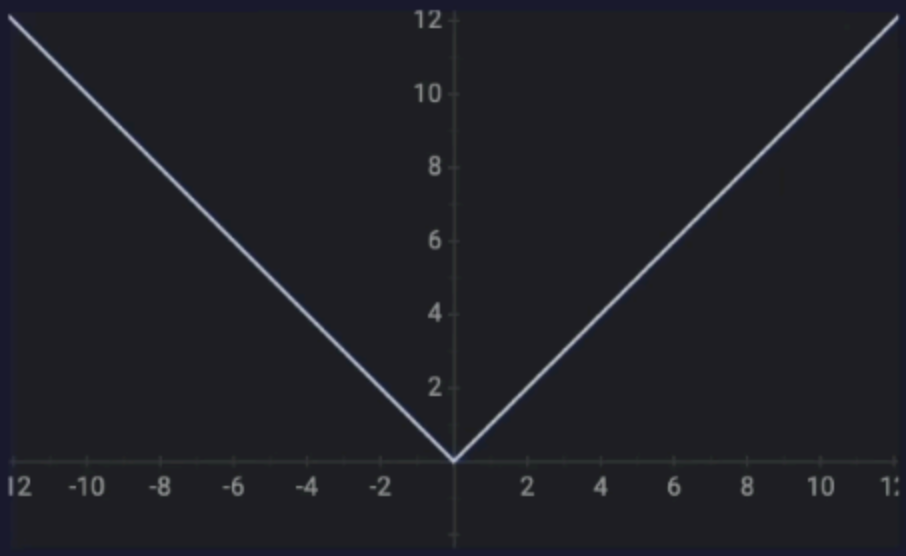


Metric of Performance

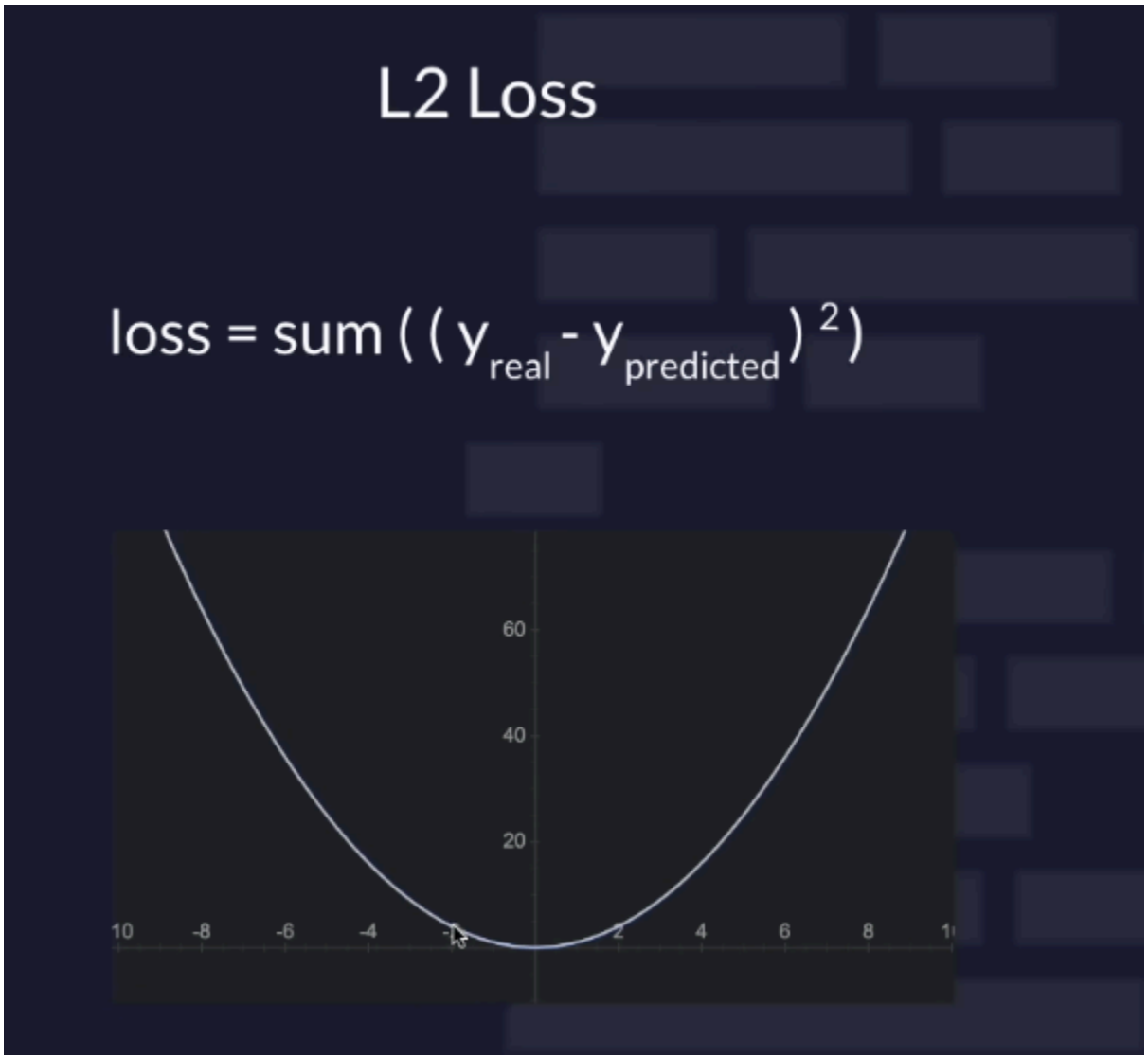
1. L1 loss

L1 Loss

$$loss = sum(| y_{real} - y_{predicted} |)$$



2. L2 Loss



High penalty for higher loss

3. Binary Cross Entropy loss

ACCURACY

[use numpy.split to split the data]

[scale of dataset]

1. **Scale the data so that its relative & SD of specific column**
2. **From sklearn.preprocessing import standardscaller**
3. **sclare=StandardScaler()**
4. **X=scaler.fit_transform(X)**
5. **Data -= np.hstrack-> put side by side**
6. **Y= np.reshape(y,(-1,1)) #-1 infer what would be len of df**
7. **Return data**

[oversample the training data set which are less compared to another sample]

1. From imblearn.over_sampling import RandomOverSampler
2. Ros= RandomOVerSampler|()
3. X,y= ros.fit_resample(X,y)
4. Dont do on validation & test data

[Classification report] to decide which model is better
From sklearn.metrics import classification_report

K-NEAREST NEIGHBOURS

LOOK around what around you and take majority what around you
Euclidean distance- straight line distance from nearest points

Distance- d=(x-x2)2+(y-y2)2^½

K- how many neighbors use to judge the parameter

Majority of points which are close by-> based on ans we decide

Expand distance based on the features

PACKAGE:
From sklearn.neigbhors import KNeighboursClassifier
knn_model= KneightbourClassfier(n_neighbour=1)
knn_model.fit(X_train,y_train)
Y_predict=knn_model.predict(X_test)
classificaiton_report(y_predict,y_actual)

---

ormance metrics that

(also known as

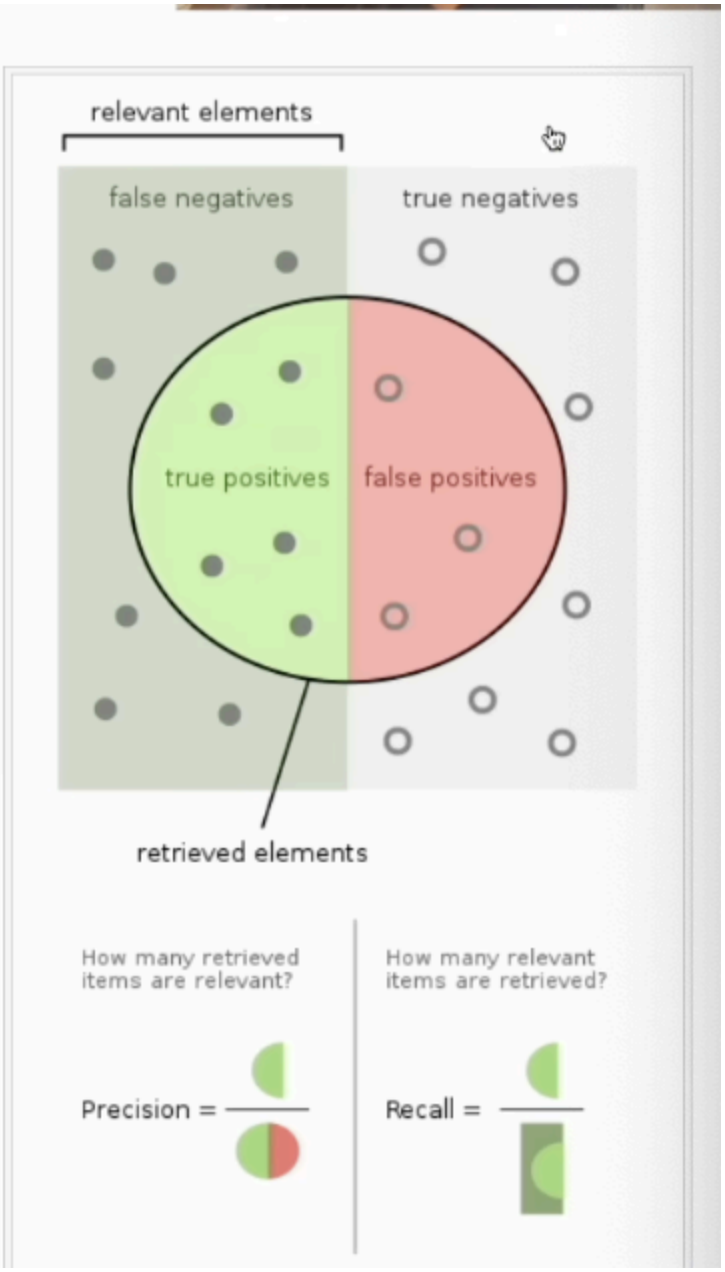
hich contains ten
ıe positives), while
ue negatives). The

ecision is 20/30 =

t, i.e., not a dog,
cision (no false

to the type I error
em.

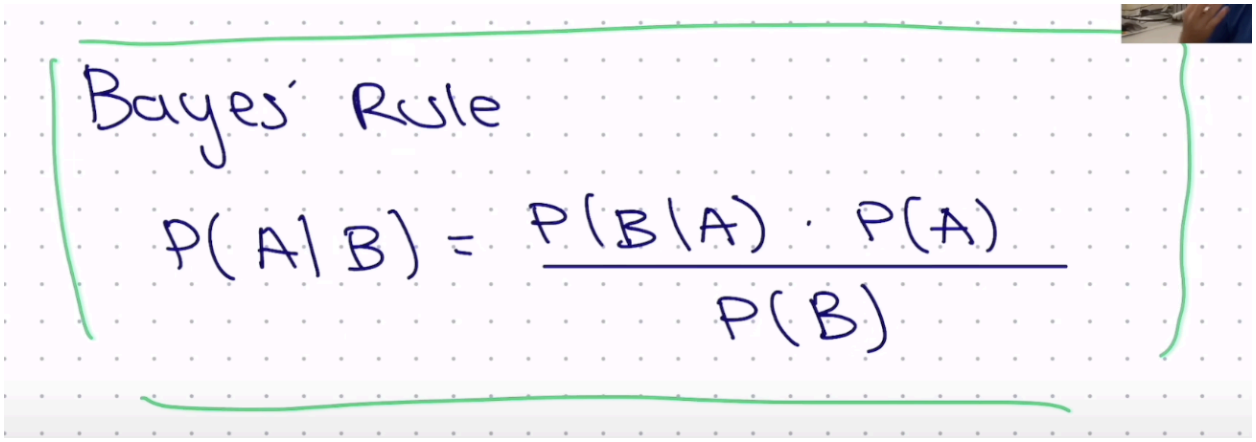
a type II error rate of
hm returns more
t irrelevant ones are

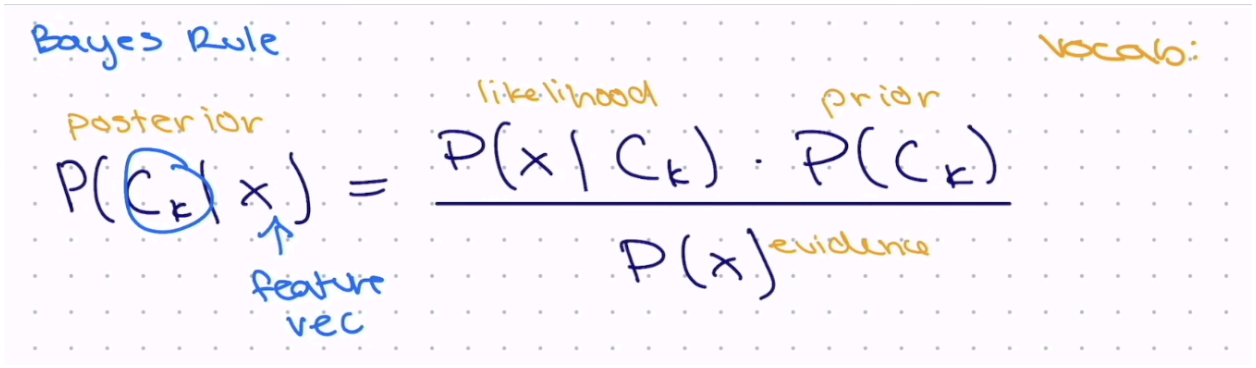


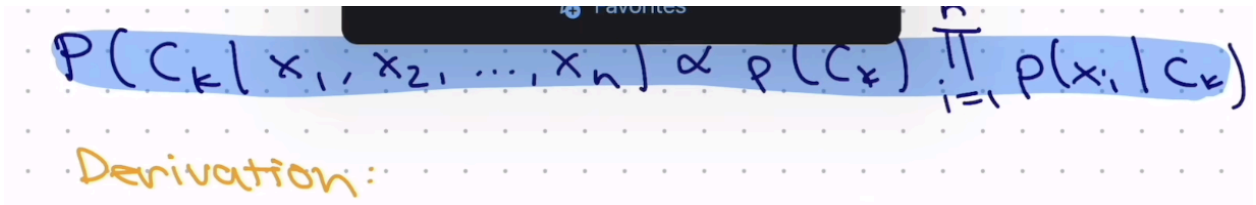
Precision- how many ot them true positive.
Recall- how many we get actually right of true positive
FI-Score- combination of precision & recall

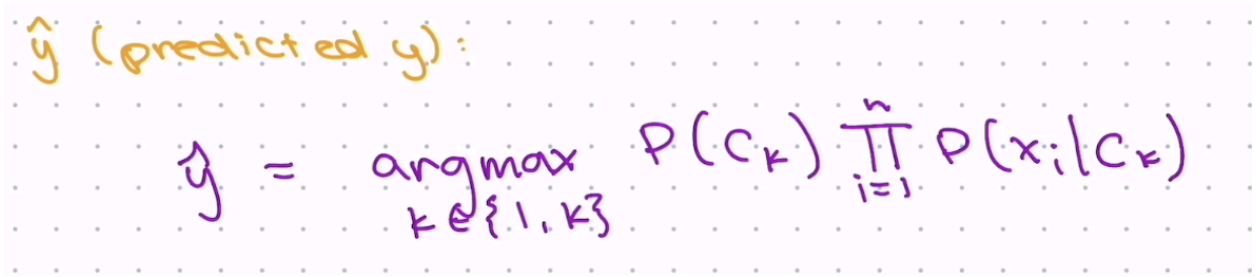
NAIVE BAYES MODEL

CONDITIONAL PROBABILITY









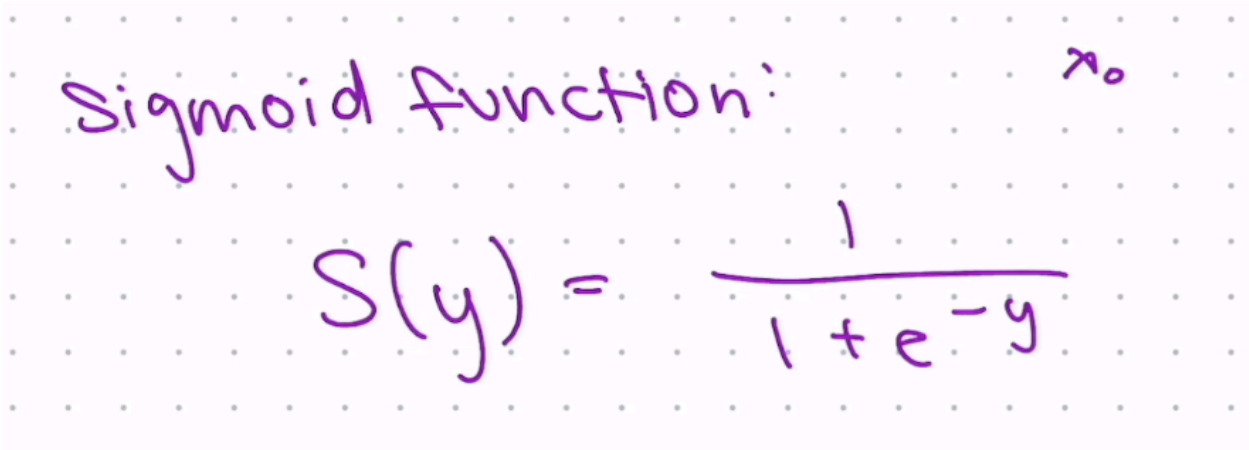
MAP - Max A Posteriori

from sklearn.naive_bayes import GausaianNB

LOGISTICS REGRESSION

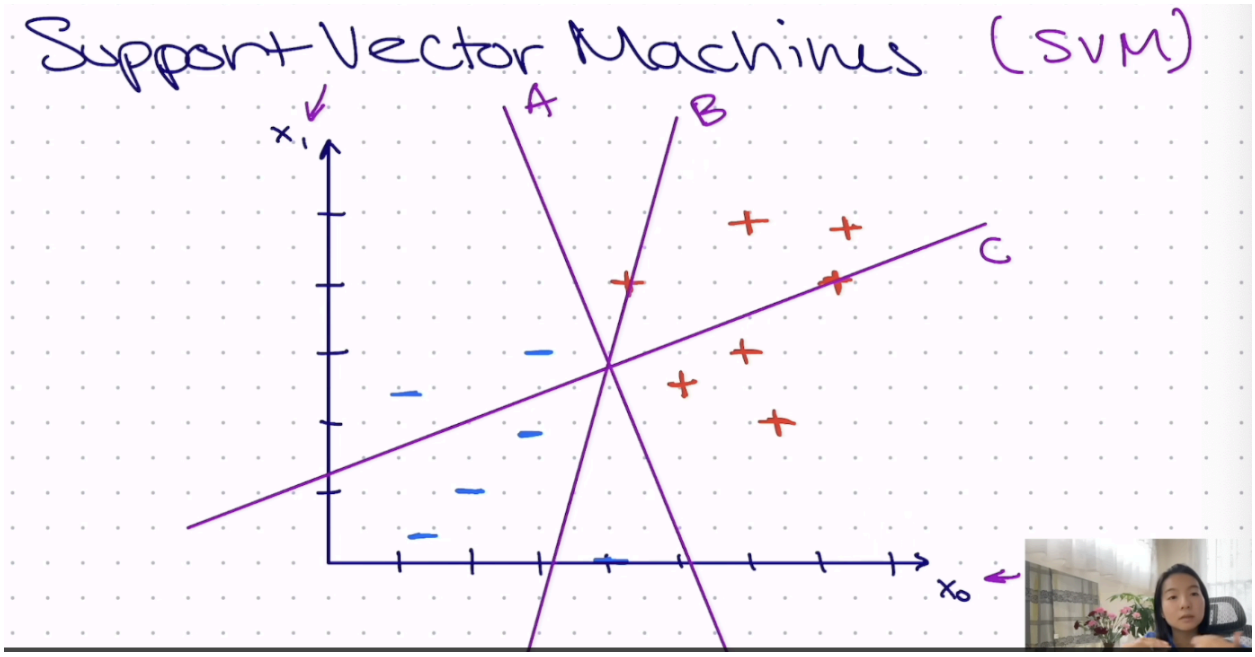
How can we model probability?
ln[p/(1- p)] = mx + b # ratio allows to take infinite value

1. Allows to take x from neg to pos infinity
2. Take exp to obtain e^(mx + b)
3. p= e^mx - pe^mx+b.
4. p =e^mx+b/(1+e^mx +b)= 1/1+e^-(-mx+b)
5. Sigmoid function= 1/1+e^-x



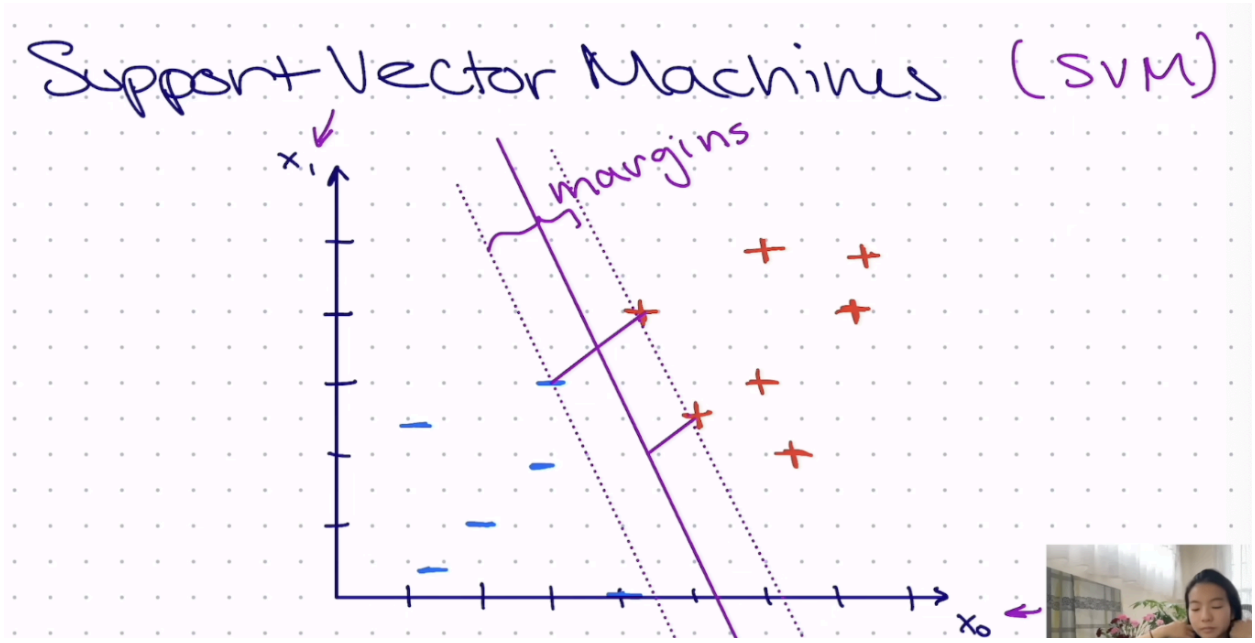
6.
7. Try to fit into sigmoid function
8. #package
   from sklearn.linear_model import LogisticRegression
   lg_model = LogisticsRegression()
   lg_model .fit(X_train,y_train)
   print(classification_report(y_test,y_pred)

SUPPORT VECTOR MACHINES [SVM] FOR CLASSIFICATION



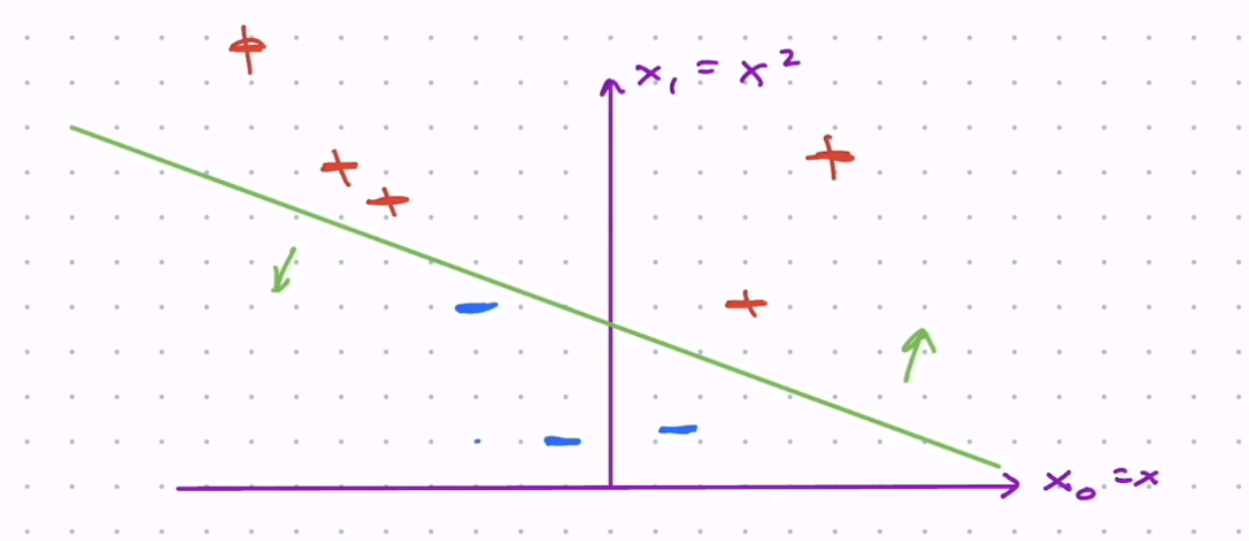
A→ clear distinct line which classifies two

Hyber plan which differentiate two classes



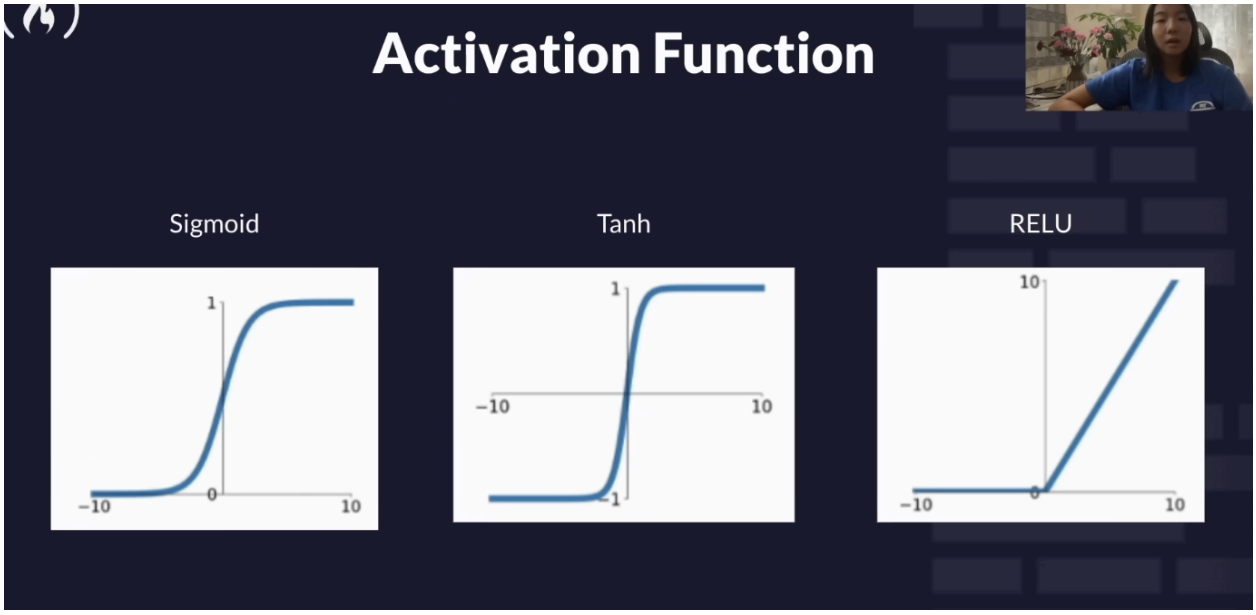
Margins-> Goal is to maximize the margins from two classifier
Issue->> not robust to outliers.
Dataset-> create projection-> taking square

*Transformation -> **Kernel trick -> x⇒ (x,x^2)***

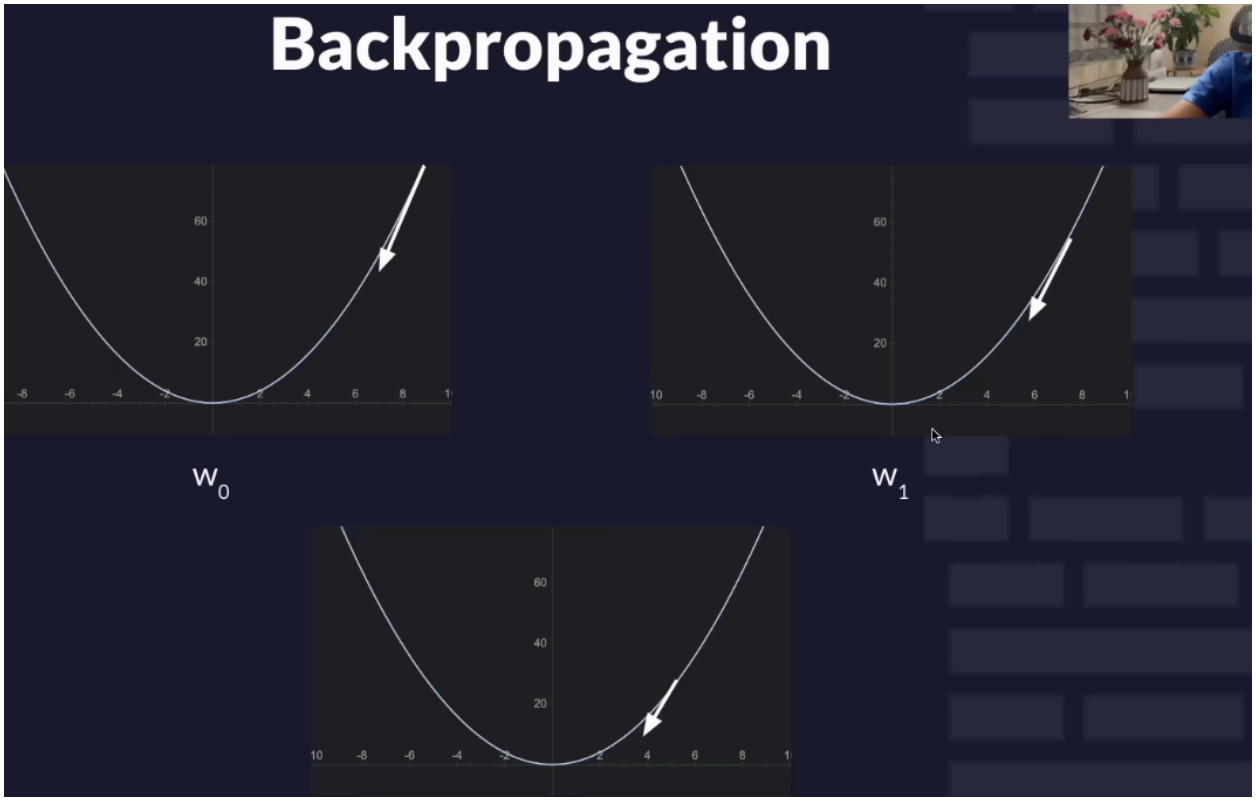



```
from sklean.svm import SVC
svm_model =svm_model.predict(X_test)
Y_pred = svm_model.prdict(x_test)
classification_report(X_train,y_pred)
```

NEURAL NETWORKS



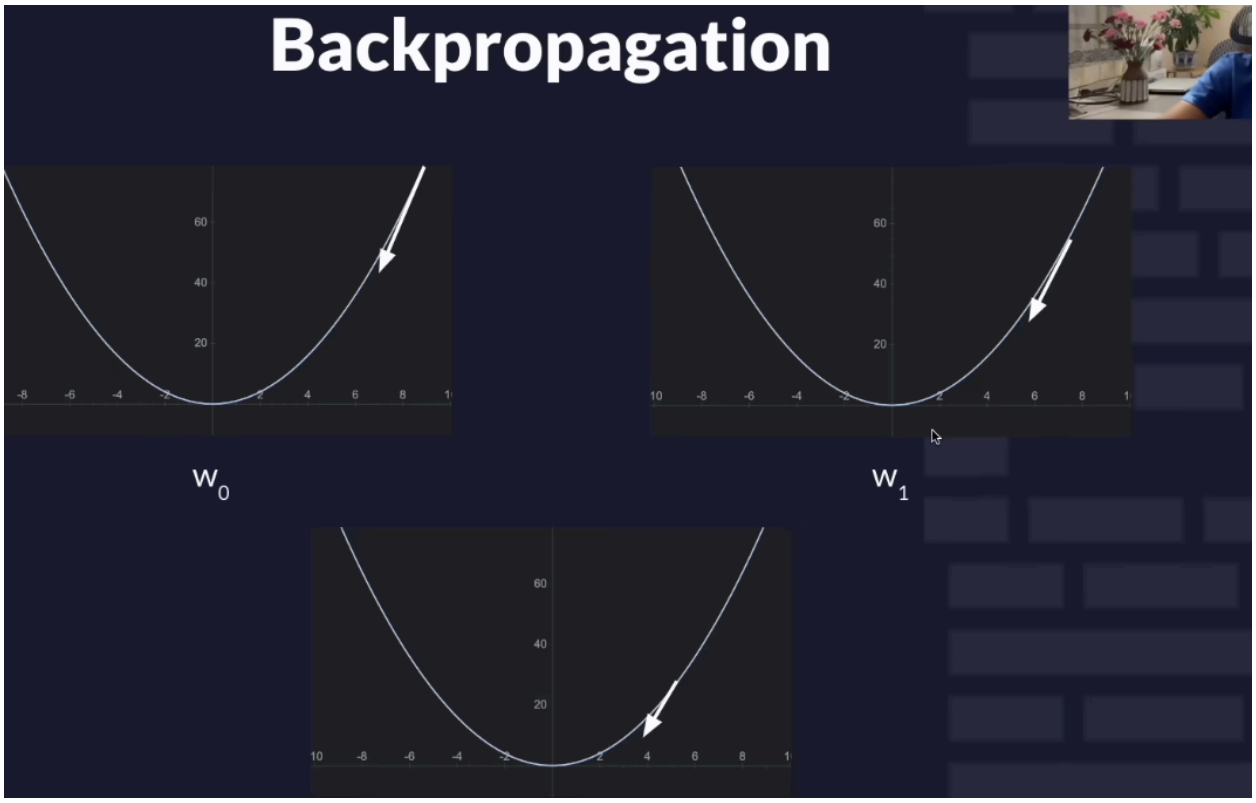
Activation function-< neuron is not a linear combination. Does not collapse of its own



L2 loss function.
Our goal is to go bottom=> go down with help of gradient descent

Also calculus . How much we have to backstrap . Smaller we

weigth_update = weight(old) + alpha*
alpha - learning rate- how long it takes nn to converge



 # Neural network libraries

Tensorflow
tf.keras.Sequential[tf.keras.layers.Dense(16,

Open source lib to create NN

Import tensorflow as tf
nn_model = tf.keras.Sequential([
tf.keras.layers.Dense(64,activation='relu',input_shape=(10,))])
tf.keras.layers.Dense(64,activation='relu',])
tf.keras.layers.Dense(64,activation='sigmoid'])

nn_model.compule(optimiser=tf.keras.optimizers.Admo(0.001),los
s='binary_crossentropy',metrics=['accuracy']

TensorFlow-> each single epoch-> leave 20% and see how it performs,verbose=0

plot_loss(history)--> decreasing loss, and increasing accuracy validation->

Grid search: what happens with different nodes, learning size, batch sise

Dropout= randomly choose certain nodes and dont train them. Avoid overtraining problem

y_pred =(y_prd > 0.5).reshape(-1,1)

REGRESSION- OTHER SETS OF SUPERVISED LEARNING TILL NOW WE LEARN ABOUT CLASSIFICATION MODEL

What would be my prediction for given x value .

Fit a linear model - line of best fit-> model by the equation
y= b0 + b1x
b1-:slope

RESIDUALS
ERROR:
Take some data point and evaluate how far is it from actual data points

Prediction- y^
residual =| y-y^ |
Residual as Yi-Y^ .Distance bet point and its predicted point
Generally trying to decrease the residual for all given points
Sum(abs(y-y^))-> try to decrease by line of best fit. We also decrease the sum of squared residuals . higher penalty for higher errors.
This is called simple LG.

MULTIPLE LINEAR REGRESSION
Multiple values of x vector so our line look like
y= b0+b1x1+b2x2….

ASSUMPTIONS

Linearity
Does my data follow a linear pattern. If y inc or dec then looking linear.
Independence
All samples are independent
Normality
Residual plot should be normally distributed

Homoskedasticity

Distributed such that its variance constant

Homoskedasticity is an assumption in linear regression that the variance of the errors (or residuals) is constant across all values of the independent variable(s). In simpler terms, it means that the "spread" of the errors should be roughly the same no matter what value of the predictor you have.

Simple Example

Imagine you're trying to predict students' exam scores (Y) based on the number of hours they study (X) using a linear regression model:

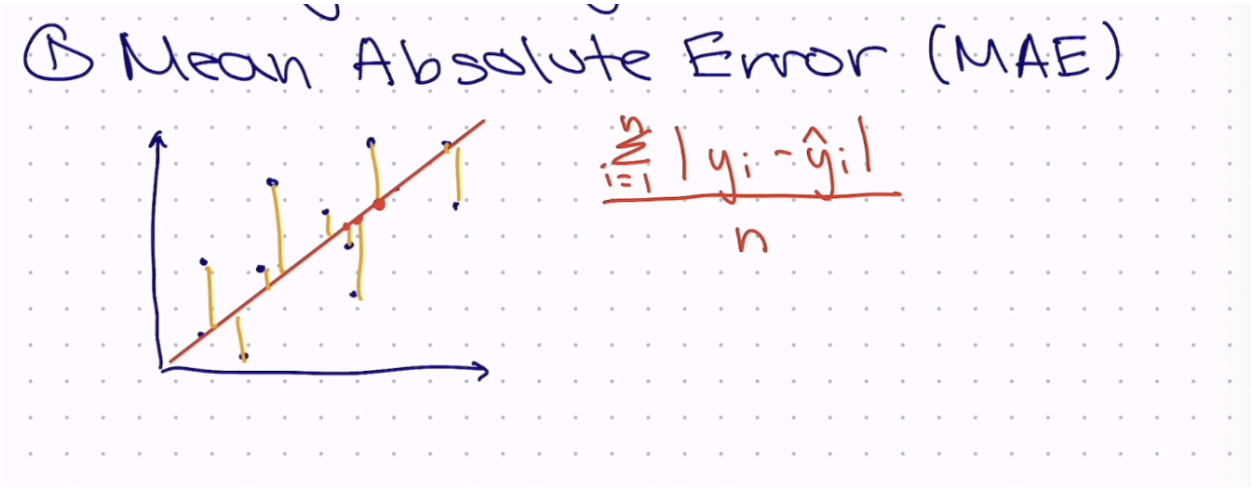
Exam Score=β0+β1×Study Hours+error\text{Exam Score} = \beta_0 + \beta_1 \times \text{Study Hours} + \text{error}Exam Score=β0+β1×Study Hours+error

- **Homoskedasticity:**
  If for students who study 1 hour, the exam scores vary by about ±10 points around the predicted value, and for students who study 5 hours, the exam scores also vary by about ±10 points, then the errors have a constant spread. This means the assumption of homoskedasticity holds.

- **Why It Matters:**
  This constant error spread is important because many statistical tests and confidence intervals in regression rely on the assumption that the error variance is the same across all levels of X. If this isn't true, it can lead to inefficient estimates and unreliable inference.

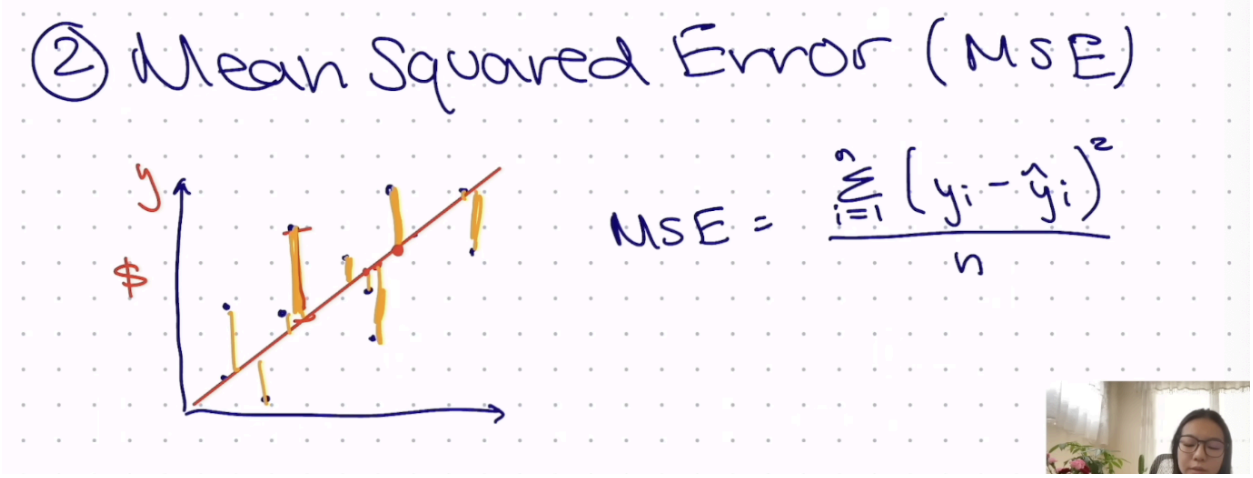
EVALUATION LIN REG MODEL

1. MEAN ABSOLUTE ERROR[MAE]



allows the average price of off is this many dollars

2. Mean Square Error[MSE]

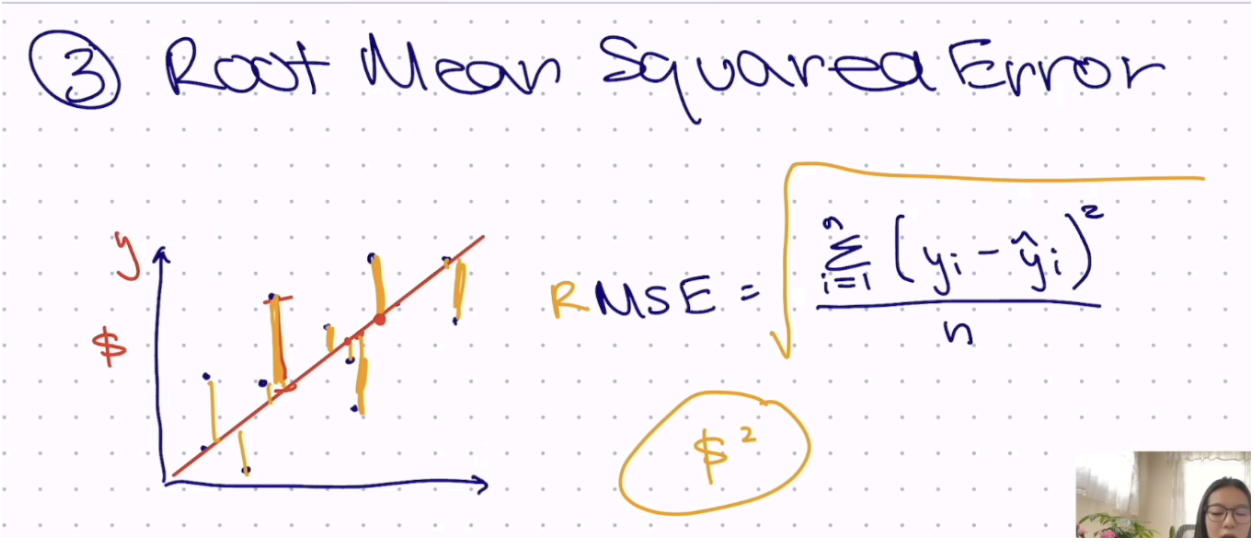


Helps to punish the high errors.

Downsize→ little bit tricky to compare. we

3. Root Mean Squared Error

Now we can say our error is this much dollar

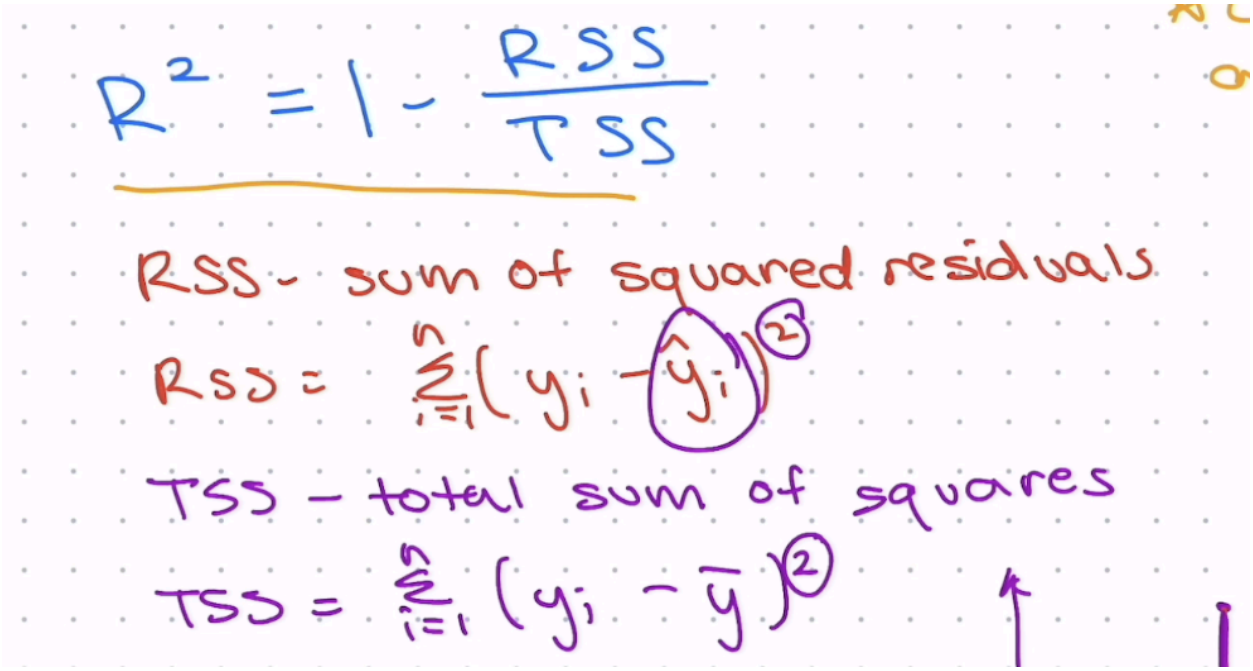


4. **R2->Coefficient of Determination**

   R2= 1-RSS/TSS

   RSS= Sum of sq residual Sum[y-y^]2→ error wrt best fit

   TSS= total sum of squares. Sum(y-ymean)^2→ error wrt mean



R2 towards 1 -> it signs that its good predictor.
Adjusted R

Predict certain values: LG

Regression with neural network

temp_normalizer = tf.keras.layers.Normalizatoin(input_share=(1,)
temp_normalizer.adapt(X_train_temp,
tf.keras.layers.Dense(-1)
}]
temp_nn_model.compile(optimizer=tf.keras.optimizers.
Ada(learning_rate

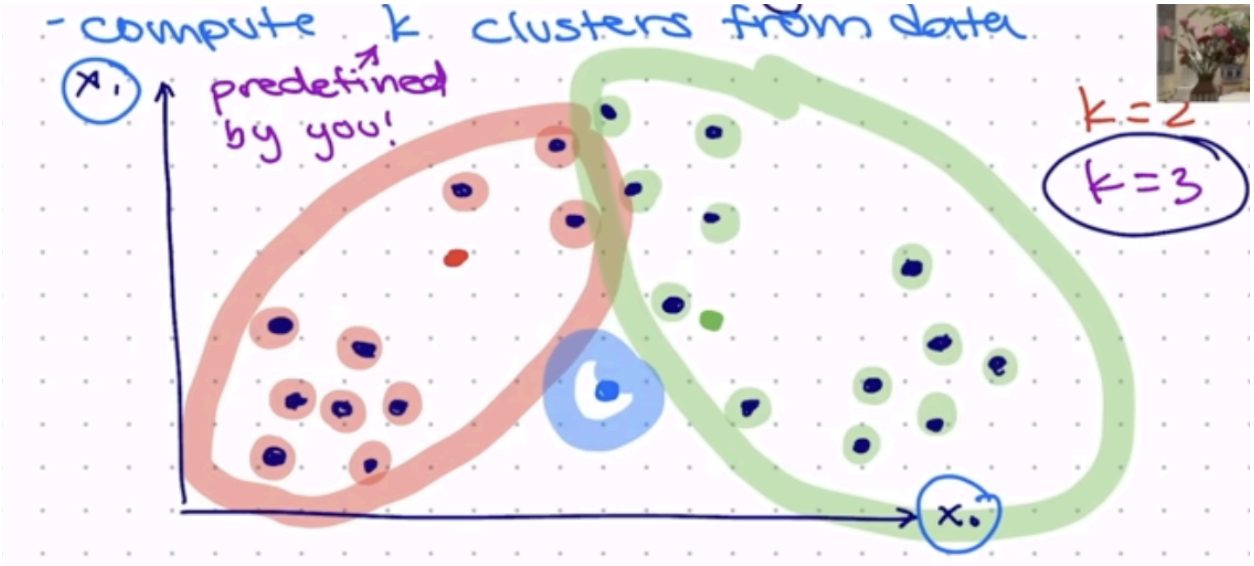
UNSUPERVISED LEARNING

1. **K-means clustering**

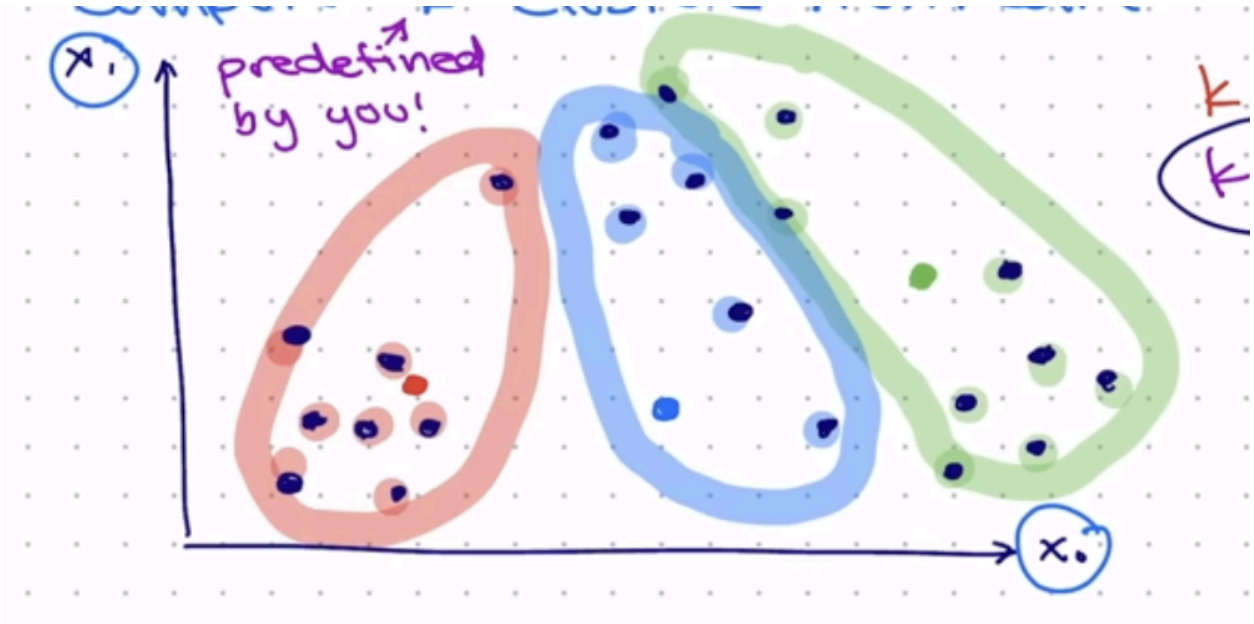
   Compute k clusters from data

   Step 1> choose 3 random points to be centroids

Step 2> Calculate distance b/t points & centroid
Step 3> Assigning points to closed centroid,



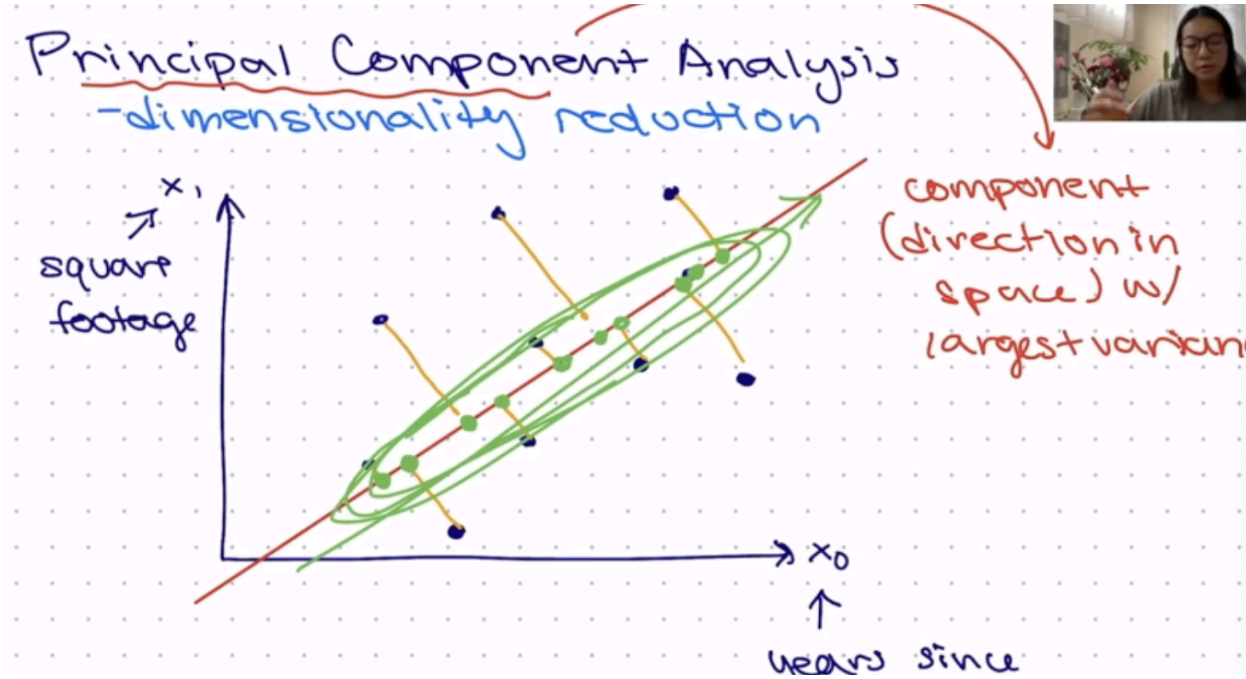
Step 4> recalculate the centroid based on points we have centroid. By taking the average of all points.



Step 5> expectation maximixation until nothing is changing . Finding the pattern -> come up with new point we can surely say that it is closer to some centroid

2. Principal Component Analysis

- **Dime**nsionality reduction
- X0 to X1
- Housing prices
- Find direction in space with largest variance
- We are taking right angle projection
- Map all to one dimensional points
- One dimensional datasets



- F
- Minimize the projection
- dimensions minimize projection residual or maximize the variance
- Taking data which is unlabelled and guess about structure.
- Unsupervised learning -> Kerne

hue-> scatter across multiple groups

Learning machine learning

1. Probability & statistics
    a. Conditional P
    b. Bayes Rule
    c. Statistical distribution
2. Calculus
    a. Optimization→ Gradient Descent
    b. Derivatives
3. Linear Algebra
    a. Vector & matrices
    b. Eigen vectors & Eigen values
4. Programming Skills
    a. Python
    b. Basic concepts→ variables, functions, classes,
    c. Libraries→ Pandas, Numpy, Scikit-learn, Tensorflow, Pytorch
    d. Plotting→ Matplotlib, seaborn
5. Core Concepts
    a. What is ML
        i. Types →
            1. Supervised → Tasks→ Classification & Regression
            2. Unsupervised
            3. Reinforcement
    b. Data
        i. Types→
            1. Qualitative

        2. Quantitative
    ii. Training / Validation / Testing
    iii. Manipulating
        1. Data cleaning →
        2. Feature scaling,
        3. Feature Engineering → stock price→ include return instead of price
c. Models
    i. K-nearest neighbours
    ii. Logistics Regression / Log regression
    iii. SVM (Support Vector Model)
    iv. Linear regression
    v. Neural Net/networks
        1. Perceptrons
    vi. K- Means
    vii. PCA Principal Component Analysis
d. Neural Network
    i. CNN Convolution NN→ running NN on image
    ii. RNN Recurrent→ sequential data
        1. GRU
        2. LSTM

6. Training & Evaluating the model
    a. Metrics
    b. Overfitting
7. Practice + Research
    a. Projects
    b. Youtube / internet
    c. UCI Machine learning repository
    d. Kaggle
    e. Papers→ implementing the papers

f. Internet + experts