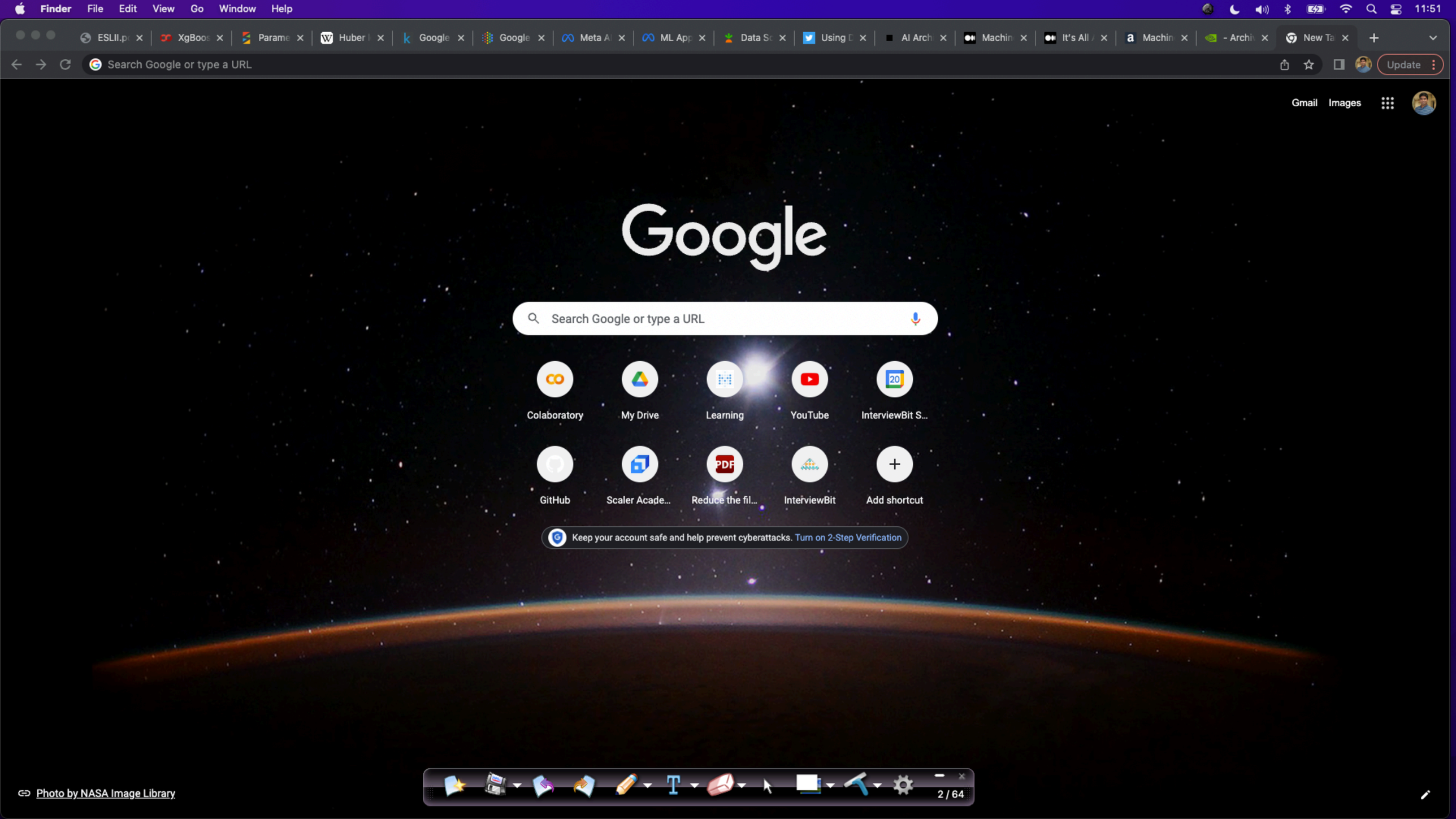
Topics:

- GBDT : classification -loss
  - lightGBM - code
  - pseudo - residual

→
- RF vs GBDT vs Cascading vs Stacking
- Cascading & stacking → Questions
- Kaggle vs real-world → ...

- SVM
  - intuition
  - Math (lots)


Finder File Edit View Go Window Help
11:51
Search Google or type a URL
Update
Gmail Images
Google
Search Google or type a URL
Colaboratory
My Drive
Learning
YouTube
InterviewBit S...
GitHub
Scaler Acade...
Reduce the fil...
InterviewBit
Add shortcut
Keep your account safe and help prevent cyberattacks. Turn on 2-Step Verification
Photo by NASA Image Library
2 / 64

```
[ ] # Domain specific ideas to improve the results: Average data across time.
    ## Source: https://www.researchgate.net/figure/EMG-signal-process-recommended-Green-The-raw-signal-no-treatment-was
```

```
# LightGBM: Sample code
import lightgbm as lgb
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
#Refer: https://lightgbm.readthedocs.io/en/latest/Parameters.html

gridParams = {
    'learning_rate': [0.1, 0.5, 0.8],
    'boosting_type' : ['gbdt'],
    'objective' : ['multiclass'],
    'max_depth' : [5,6,7,8],
    'colsample_bytree' : [0.5,0.7],
    'subsample' : [0.5,0.7],
    'metric':['multi_error'],
    'random_state' : [501]
    }

clf = lgb.LGBMRegressor(num_classes=20)
grid = RandomizedSearchCV(clf,gridParams,verbose=3,cv=3,n_jobs = -1,n_iter=10,)
grid.fit(X_train,Y_train)
```

```
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
#Refer: https://lightgbm.readthedocs.io/en/latest/Parameters.html

gridParams = {
    'learning_rate': [0.1, 0.5, 0.8],
    'boosting_type' : ['gbdt'],
    'objective' : ['multiclass'],
    'max_depth' : [5,6,7,8],
    'colsample_bytree' : [0.5,0.7],
    'subsample' : [0.5,0.7],
    'metric':['multi_error'],
    'random_state' : [501]
    }

clf = lgb.LGBMRegressor(num_classes=20)
grid = RandomizedSearchCV(clf,gridParams,verbose=3,cv=3,n_jobs = -1,n_iter=10,)
grid.fit(X_train,Y_train)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_search.py:972: UserWarning: One or more of the test
  category=UserWarning,
RandomizedSearchCV(cv=3, estimator=LGBMRegressor(num_classes=20), n_jobs=-1,
                   param_distributions={'boosting_type': ['gbdt'],
                                        'colsample_bytree': [0.5, 0.7],
                                        'learning_rate': [0.1, 0.5, 0.8],
                                        'max_depth': [5, 6, 7, 8],
                                        'metric': ['multi_error'],
```

- requires labels in {0, 1}; see `cross-entropy` application for general probability labels in [0, 1]
  - multi-class classification application
    - `multiclass`, softmax objective function, aliases: `softmax`
    - `multiclassova`, One-vs-All binary objective function, aliases: `multiclass_ova`, `ova`, `ovr`
    - `num_class` should be set as well
  - cross-entropy application
    - `cross_entropy`, objective function for cross-entropy (with optional linear weights), aliases: `xentropy`
    - `cross_entropy_lambda`, alternative parameterization of cross-entropy, aliases: `xentlambda`
    - label is anything in interval [0, 1]
  - ranking application
    - `lambdarank`, lambdarank objective. label_gain can be used to set the gain (weight) of `int` label and all values in `label` must be smaller than number of elements in `label_gain`
    - `rank_xendcg`, XE_NDCG_MART ranking objective function, aliases: `xendcg`, `xe_ndcg`, `xe_ndcg_mart`, `xendcg_mart`
    - `rank_xendcg` is faster than and achieves the similar performance as `lambdarank`
    - label should be `int` type, and larger number represents the higher relevance (e.g. 0:bad, 1:fair, 2:good, 3:perfect)
- `boosting` 🔗, default = `gbdt`, type = enum, options: `gbdt`, `rf`, `dart`, `goss`, aliases: `boosting_type`, `boost`
  - `gbdt`, traditional Gradient Boosting Decision Tree, aliases: `gbrt`
  - `rf`, Random Forest, aliases: `random_forest`
  - `dart`, Dropouts meet Multiple Additive Regression Trees
  - `goss`, Gradient-based One-Side Sampling
    - **Note**: internally, LightGBM uses `gbdt` mode for the first `1 / learning_rate` iterations
- `data` 🔗, default = `""`, type = string, aliases: `train`, `train_data`, `train_data_file`, `data_filename`
  - path of training data, LightGBM will train from this data
  - **Note**: can be used only in CLI version
- `valid` 🔗, default = `""`, type = string, aliases: `test`, `valid_data`, `valid_data_file`, `test_data`, `test_data_file`, `valid_filenames`
  - path(s) of validation/test data, LightGBM will output metrics for these data
  - support multiple validation data, separated by `,`
  - **Note**: can be used only in CLI version
- `num_iterations` 🔗, default = `100`, type = int, aliases: `num_iteration`, `n_iter`, `num_tree`, `num_trees`, `num_round`, `num_rounds`, `nrounds`, `num_boost_round`, `n_estimators`
  - number of boosting iterations

```
gridParams = {
    'learning_rate': [0.1, 0.5, 0.8],
    'boosting_type' : ['gbdt'],
    'objective' : ['multiclass'],
    'max_depth' : [5,6,7,8],
    'colsample_bytree' : [0.5,0.7],
    'subsample' : [0.5,0.7],
    'metric':['multi_error'],
    'random_state' : [501]
    }

clf = lgb.LGBMRegressor(num_classes=20)
grid = RandomizedSearchCV(clf,gridParams,verbose=3,cv=3,n_jobs = -1,n_iter=10,)
grid.fit(X_train,Y_train)
```

$P(y_i=1|x_i)$
$P(y_i=2|x_i)$
$\vdots$
$P(y_i=20|x_i)$

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_search.py:972: UserWarning: One or more of the test
  category=UserWarning,
RandomizedSearchCV(cv=3, estimator=LGBMRegressor(num_classes=20), n_jobs=-1,
                   param_distributions={'boosting_type': ['gbdt'],
                                        'colsample_bytree': [0.5, 0.7],
                                        'learning_rate': [0.1, 0.5, 0.8],
                                        'max_depth': [5, 6, 7, 8],
                                        'metric': ['multi_error'],
                                        'objective': ['multiclass'],
```

```
gridParams = {
    'learning_rate': [0.1, 0.5, 0.8],
    'boosting_type' : ['gbdt'],
    'objective' : ['multiclass'],
    'max_depth' : [5,6,7,8],
    'colsample_bytree' : [0.5,0.7],
    'subsample' : [0.5,0.7],
    'metric':['multi_error'],
    'random_state' : [501]
    }

clf = lgb.LGBMRegressor(num_classes=20)
grid = RandomizedSearchCV(clf,gridParams,verbose=3,cv=3,n_jobs = -1,n_iter=10,
grid.fit(X_train,Y_train)
```

3-fold CV

Cores

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_search.py:972: UserWarning: One or more of the test
  category=UserWarning,
RandomizedSearchCV(cv=3, estimator=LGBMRegressor(num_classes=20), n_jobs=-1,
                   param_distributions={'boosting_type': ['gbdt'],
                                        'colsample_bytree': [0.5, 0.7],
                                        'learning_rate': [0.1, 0.5, 0.8],
                                        'max_depth': [5, 6, 7, 8],
                                        'metric': ['multi_error'],
                                        'objective': ['multiclass'],
```

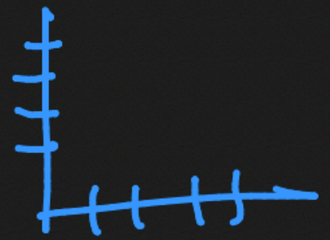
```
gridParams = {
    'learning_rate': [0.1, 0.5, 0.8],
    'boosting_type' : ['gbdt'],
    'objective' : ['multiclass'],
    'max_depth' : [5,6,7,8],
    'colsample_bytree' : [0.5,0.7],
    'subsample' : [0.5,0.7],
    'metric':['multi_error'],
    'random_state' : [501]
    }

clf = lgb.LGBMRegressor(num_classes=20)
grid = RandomizedSearchCV(clf,gridParams,verbose=3,cv=3,n_jobs = -1,n_iter=10,)
grid.fit(X_train,Y_train)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_search.py:972: UserWarning: One or more of the test
  category=UserWarning,
RandomizedSearchCV(cv=3, estimator=LGBMRegressor(num_classes=20), n_jobs=-1,
                   param_distributions={'boosting_type': ['gbdt'],
                                        'colsample_bytree': [0.5, 0.7],
                                        'learning_rate': [0.1, 0.5, 0.8],
                                        'max_depth': [5, 6, 7, 8],
                                        'metric': ['multi_error'],
                                        'objective': ['multiclass'],
```

```
    'objective' : ['multiclass'],
    'max_depth' : [5,6,7,8],
    'colsample_bytree' : [0.5,0.7],
    'subsample' : [0.5,0.7],
    'metric':['multi_error'],
    'random_state' : [501]
    }

clf = lgb.LGBMRegressor(num_classes=20)
grid = RandomizedSearchCV(clf,gridParams,verbose=3,cv=3,n_jobs = -1,n_iter=10,)
grid.fit(X_train,Y_train)
```



```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_search.py:972: UserWarning: One or more of the test
  category=UserWarning,
RandomizedSearchCV(cv=3, estimator=LGBMRegressor(num_classes=20), n_jobs=-1,
                   param_distributions={'boosting_type': ['gbdt'],
                                        'colsample_bytree': [0.5, 0.7],
                                        'learning_rate': [0.1, 0.5, 0.8],
                                        'max_depth': [5, 6, 7, 8],
                                        'metric': ['multi_error'],
                                        'objective': ['multiclass'],
                                        'random_state': [501],
                                        'subsample': [0.5, 0.7]},
                   verbose=1)
```

```
[ ]
```

```
# LightGBM: Sample code
import lightgbm as lgb
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
#Refer: https://lightgbm.readthedocs.io/en/latest/Parameters.html

gridParams = {
    'learning_rate': [0.1, 0.5, 0.8],
    'boosting_type' : ['gbdt'],
    'objective' : ['multiclass'],
    'max_depth' : [5,6,7,8],
    'colsample_bytree' : [0.5,0.7],
    'subsample' : [0.5,0.7],
    'metric':['multi_error'],
    'random_state' : [501]
    }

clf = lgb.LGBMRegressor(num_classes=20)
grid = RandomizedSearchCV(clf,gridParams,verbose=3,cv=3,n_jobs = -1,n_iter=10,)
grid.fit(X_train,Y_train)
```

→ 3

→ 3

→ 2

→ 2

9 x 4 = 36

↓

10 randomly

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_search.py:972: UserWarning: One or more of the test
  category=UserWarning,
```

10 / 10

[ ]

valu	value	value	value =	valu	value = [11, 44, 23, 25, 41, 73, 21, 227, 31, 39, 0, 0 188, 0, 0, 0, 0, 14, 0, 30]

[ ]

```
# Xgboost
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.model_selection import StratifiedKFold

import datetime as dt

params = {
        'learning_rate': [0.1, 0.5, 0.8],
        'subsample': [0.6, 0.8, 1.0],
        'colsample_bytree': [0.6, 0.8, 1.0],
        'max_depth': [3, 4, 5]
        }
xgb = XGBClassifier(n_estimators=100, objective='multi:softmax', num_class=20, silent=True)
```

$3^4 = 81$

[ ]

```
folds = 3
```

Light GBM: (2017 research paper)

↳ :

Input: m: feature dimension
$nodeSet \leftarrow \{0\}$ ▷ tree nodes in current level
$rowSet \leftarrow \{\{0, 1, 2, ...\}\}$ ▷ data indices in tree nodes
for $i = 1$ ***to*** d **do**
  for $node$ ***in*** $nodeSet$ **do**
    usedRows $\leftarrow rowSet[node]$
    for $k = 1$ ***to*** m **do**
      $H \leftarrow$ new Histogram()
      ▷ Build histogram
      for j ***in*** $usedRows$ **do**
        bin $\leftarrow I$.f[k][j].bin
        H[bin].y $\leftarrow H$[bin].y + I.y[j]
        H[bin].n $\leftarrow H$[bin].n + 1
      Find the best split on histogram H.
      ...
  Update $rowSet$ and $nodeSet$ according to the best split points.
  ...

Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: $loss$: loss function, L: weak learner
models $\leftarrow \{\}$, fact $\leftarrow \frac{1-a}{b}$
topN $\leftarrow$ a $\times$ len(I) , randN $\leftarrow$ b $\times$ len(I)
for $i = 1$ ***to*** d **do**
  preds $\leftarrow$ models.predict(I)
  g $\leftarrow loss(I$, preds), w $\leftarrow \{1,1,...\}$
  sorted $\leftarrow$ GetSortedIndices(abs(g))
  topSet $\leftarrow$ sorted[1:topN]
  randSet $\leftarrow$ RandomPick(sorted[topN:len(I)], randN)
  usedSet $\leftarrow$ topSet + randSet
  w[randSet] $\times =$ fact ▷ Assign weight $fact$ to the small gradient data.
  newModel $\leftarrow$ L(I[usedSet], $-$ g[usedSet], w[usedSet])
  models.append(newModel)

3 Gradient-based One-Side Sampling

In this section, we propose a novel sampling method for GBDT that can achieve a good balance between reducing the number of data instances and keeping the accuracy for learned decision trees.

3.1 Algorithm Description

In AdaBoost, the sample weight serves as a good indicator for the importance of data instances. However, in GBDT, there are no native sample weights, and thus the sampling methods proposed for AdaBoost cannot be directly applied. Fortunately, we notice that the gradient for each data instance in GBDT provides us with useful information for data sampling. That is, if an instance is associated with a small gradient, the training error for this instance is small and it is already well-trained. A straightforward idea is to discard those data instances with small gradients. However, the data distribution will be changed by doing so, which will hurt the accuracy of the learned model. To avoid this problem, we propose a new method called Gradient-based One-Side Sampling (GOSS).

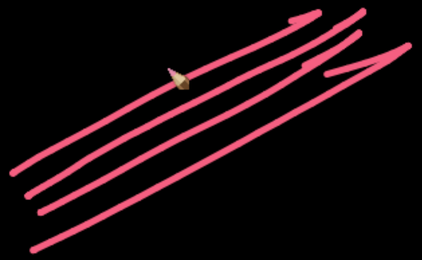


TABLE 10.2. *Gradients for commonly used loss functions.*

Setting	Loss Function	$-\partial L(y_i, f(x_i))/\partial f(x_i)$
Regression	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$
Regression	$\lvert y_i - f(x_i)\rvert$	$\text{sign}[y_i - f(x_i)]$
Regression	Huber	$y_i - f(x_i)$ for $\lvert y_i - f(x_i)\rvert \le \delta_m$ $\delta_m \text{sign}[y_i - f(x_i)]$ for $\lvert y_i - f(x_i)\rvert > \delta_m$ where $\delta_m = \alpha$th-quantile$\{\lvert y_i - f(x_i)\rvert\}$
Classification	Deviance	kth component: $I(y_i = \mathcal{G}_k) - p_k(x_i)$

boosting procedure, and top-down decision tree induction, are themselves approximation procedures. After constructing the tree (10.37), the corresponding constants in each region are given by (10.30).

Table 10.2 summarizes the gradients for commonly used loss functions. For squared error loss, the negative gradient is just the ordinary residual $-g_{im} = y_i - f_{m-1}(x_i)$, so that (10.37) on its own is equivalent standard least squares boosting. With absolute error loss, the negative gradient is

tree to the

Loss: log-loss

binary classfn:-

To show: $-\frac{\partial L}{\partial f_K(x_i)}$ = pseudo residual $\approx$ residual diff(y_i, p_i)

$F_K(x_i) = p_i = P(y_i = 1 \mid x_i)$

y_i = actual class label 0 or 1

$$\checkmark \quad -\frac{\partial L(y_i, p_i)}{\partial p_i} = -\frac{y_i}{p_i} + (1-y_i)(+1)\frac{1}{1-p_i}$$

$$\mathcal{L} = \text{Log-loss}(y_i, p_i) = \boxed{y_i \log p_i + (1-y_i)\log(1-p_i)}$$

$$-\frac{\partial L}{\partial p_i} = \frac{1-y_i}{1-p_i} - \frac{y_i}{p_i}$$

$$= \frac{p_i(1-y_i) - y_i(1-p_i)}{p_i(1-p_i)}$$

$$\frac{-\partial L}{\partial p_i} = \frac{p_i - y_i p_i - y_i + y_i p_i}{p_i(1-p_i)} = \frac{p_i - y_i}{p_i(1-p_i)}$$

Derivative → $p_i - y_i$

y_i : actual label 0 or 1

$p_i = p(y_i = 1 \mid x_i)$

$$\text{pseudo residual} = \frac{p_i - y_i}{(1-p_i)\,p_i}$$

$\hat{y}_i - y_i$

residual

diff(p_i, y_i)

SQ-loss:

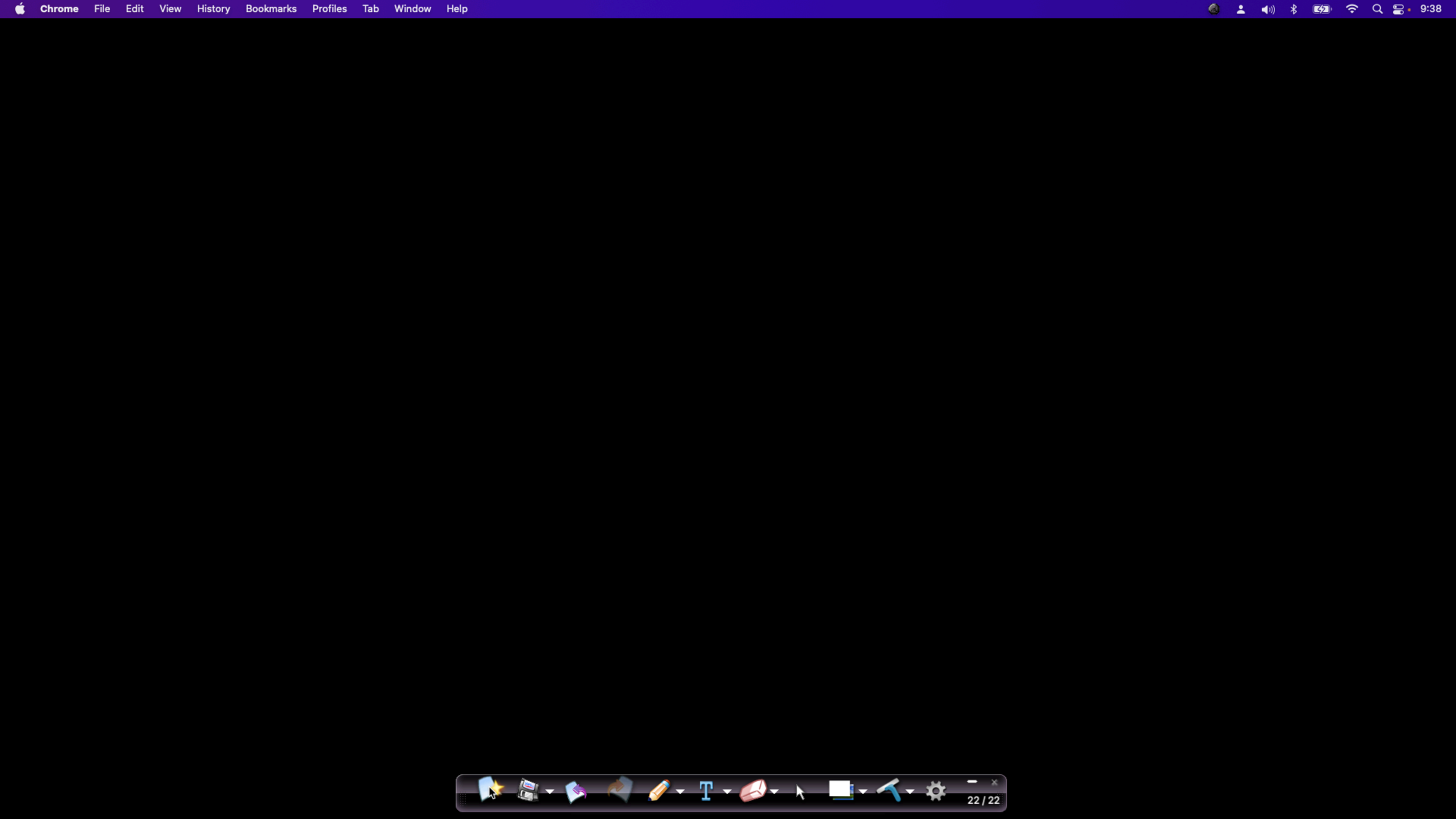
pseudo residual = $2(y_i - \hat{y}_i)$

Ques

pseudo residual $\times$ good estimate of diff loss fn

✓ [pseudo-residual of any diff... loss fn] $\simeq$ sensible residual ✓

diff $(y_i, \hat{y}_i)$

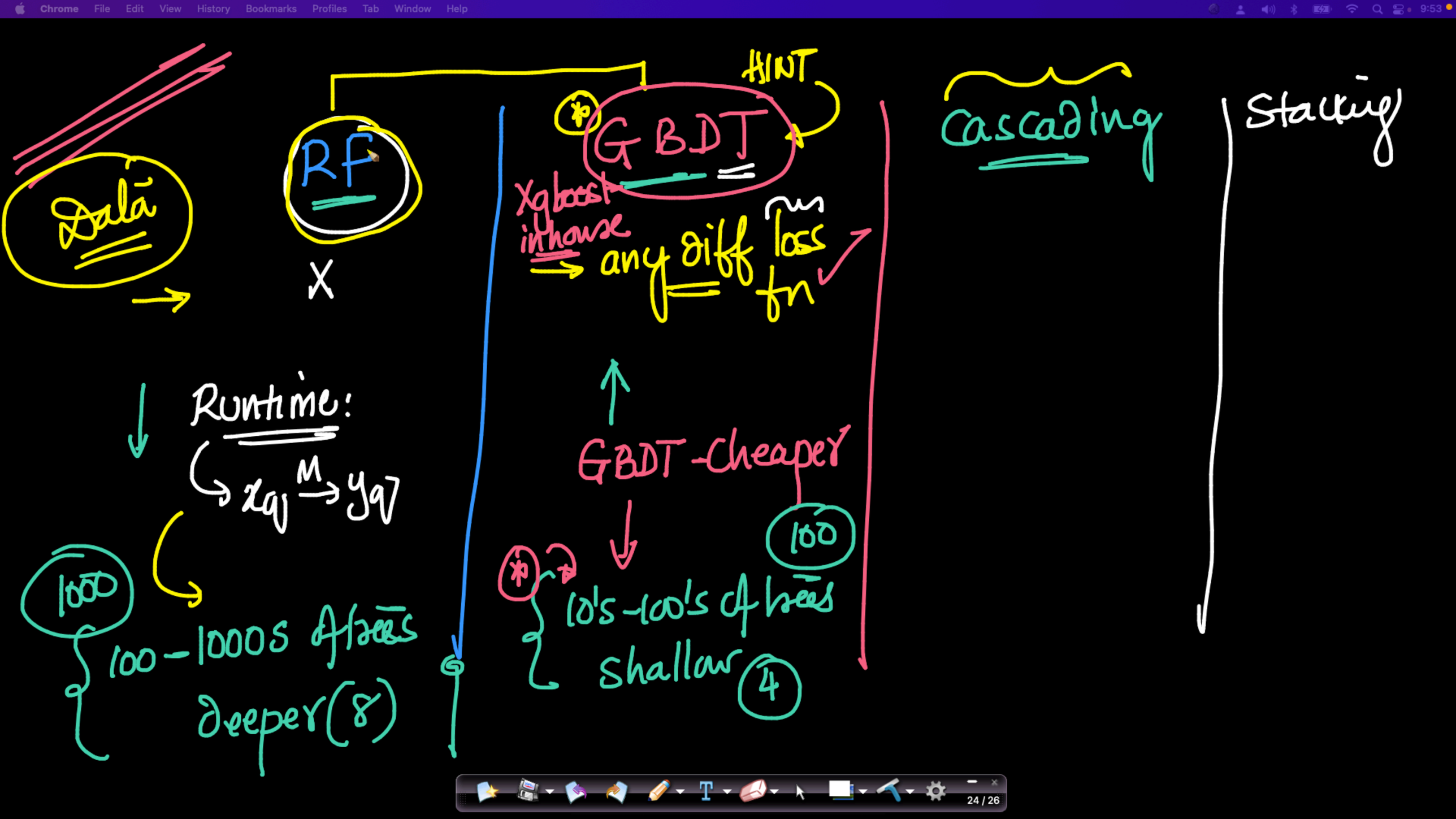

Chrome File Edit View History Bookmarks Profiles Tab Window Help
9:38
22 / 22

Deviance

log-odds → logit formulation

logistic reg: → linear reg on log-odds

$y_i - p_i$


Chrome File Edit View History Bookmarks Profiles Tab Window Help
9:53
Data
RF
X
HINT
GBDT
Xgboost inhouse
any diff loss fn
Cascading
Stacking
Runtime:
x_q → M → y_q
1000
100 - 1000s of trees
deeper (8)
GBDT - Cheaper
100
10's - 100's of trees
shallow
4
24 / 26

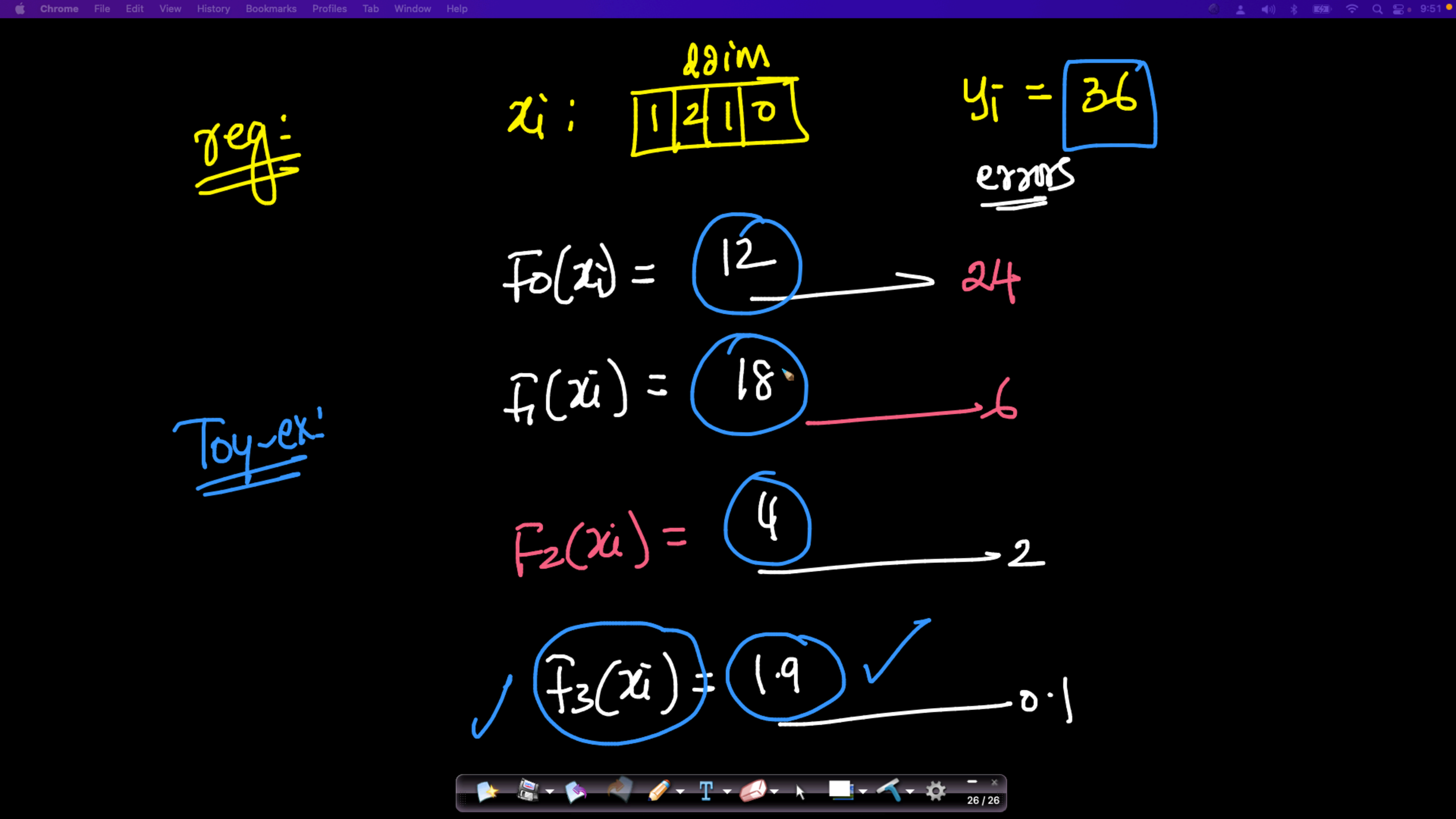
Deployment of GBDT:

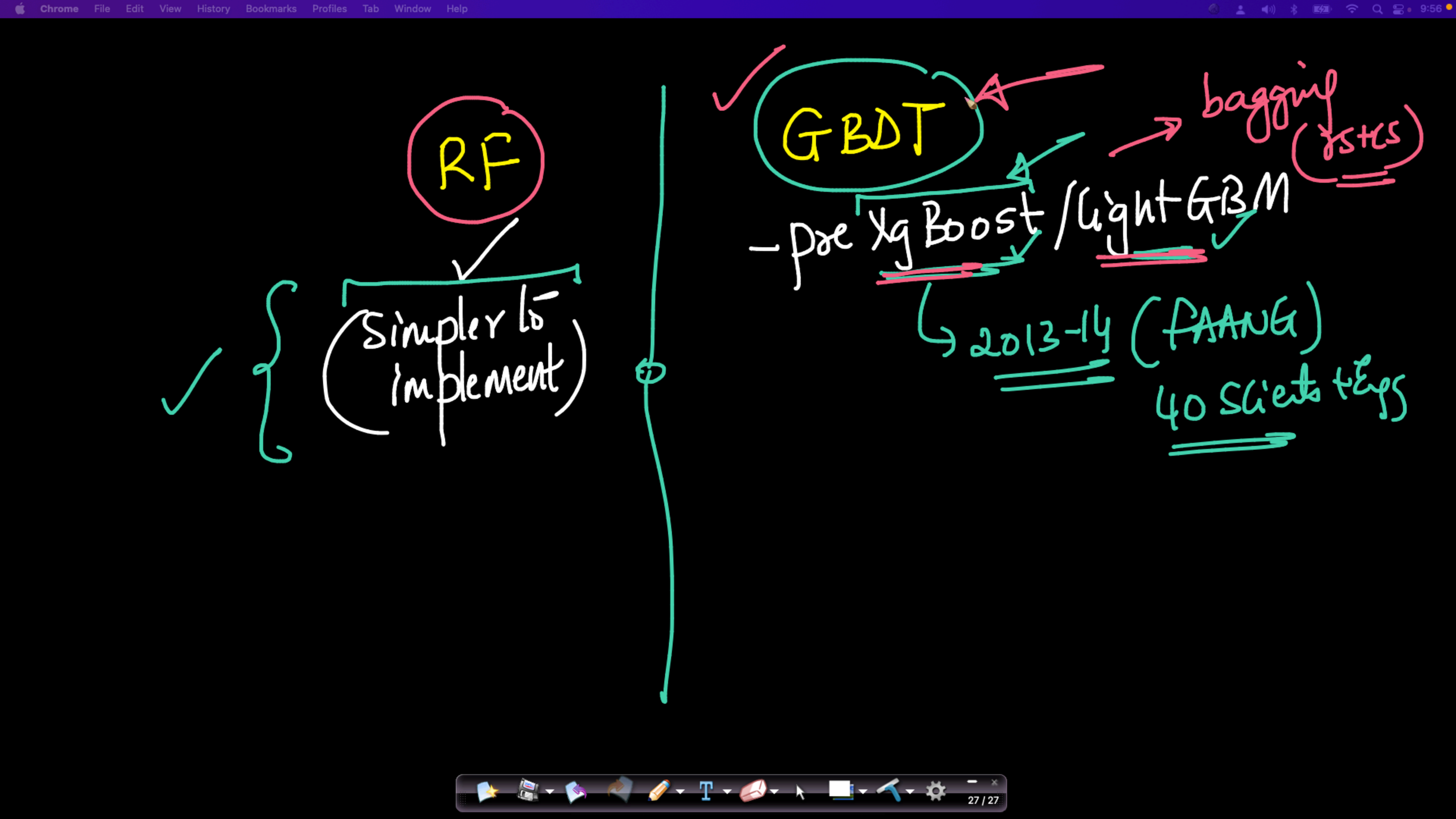
$$F_M(x) = \underbrace{1}_{\alpha_0} \cdot f_0(x) + \overbrace{\alpha_1}^{1.6} f_1(x) + \overbrace{\alpha_2}^{0.2} F_2(x) \quad \overbrace{\alpha_2}^{1.2} \quad \overbrace{\alpha_2}^{1.8}$$

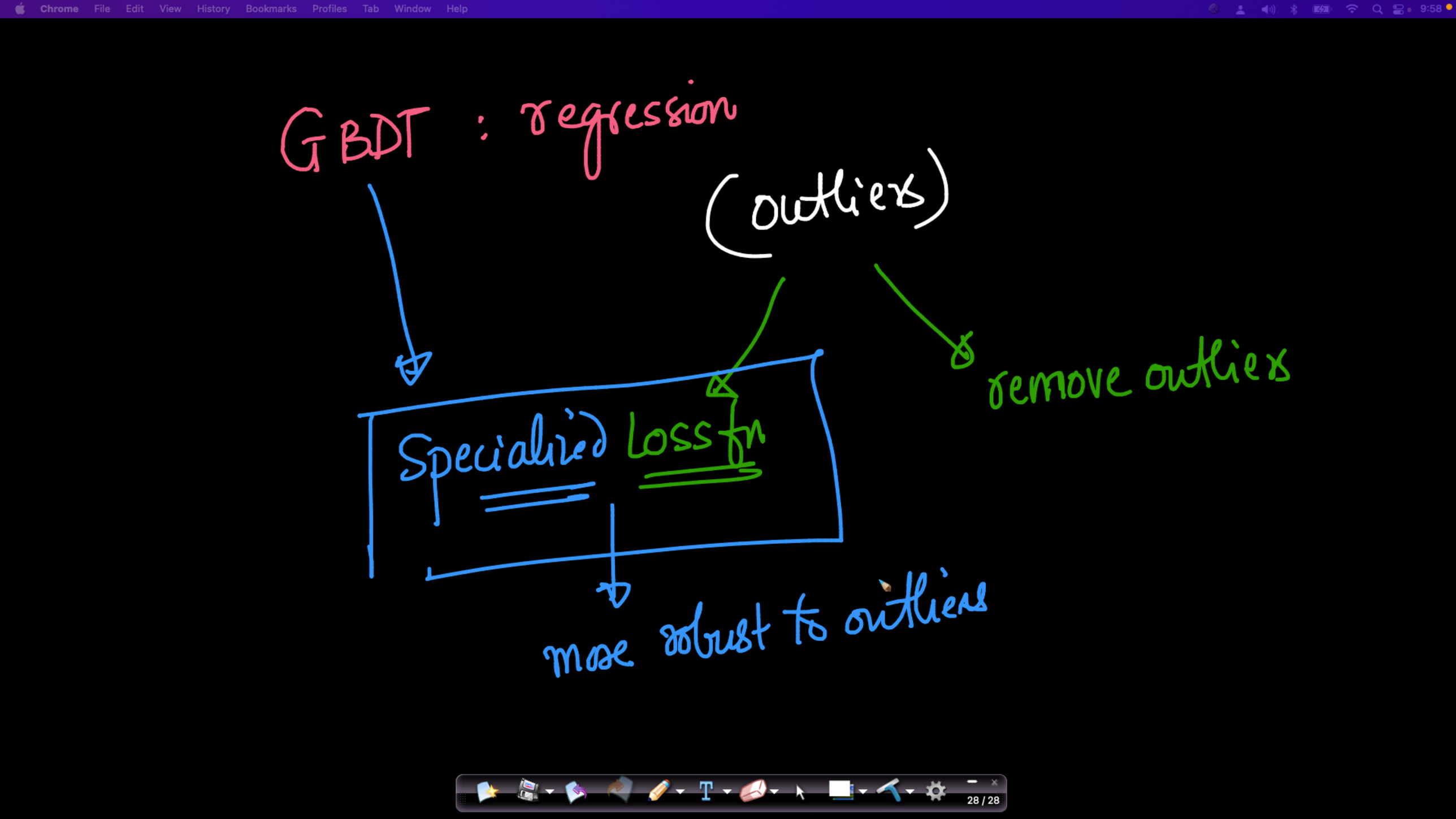
$$\cdots + \alpha_M F_M(x)$$

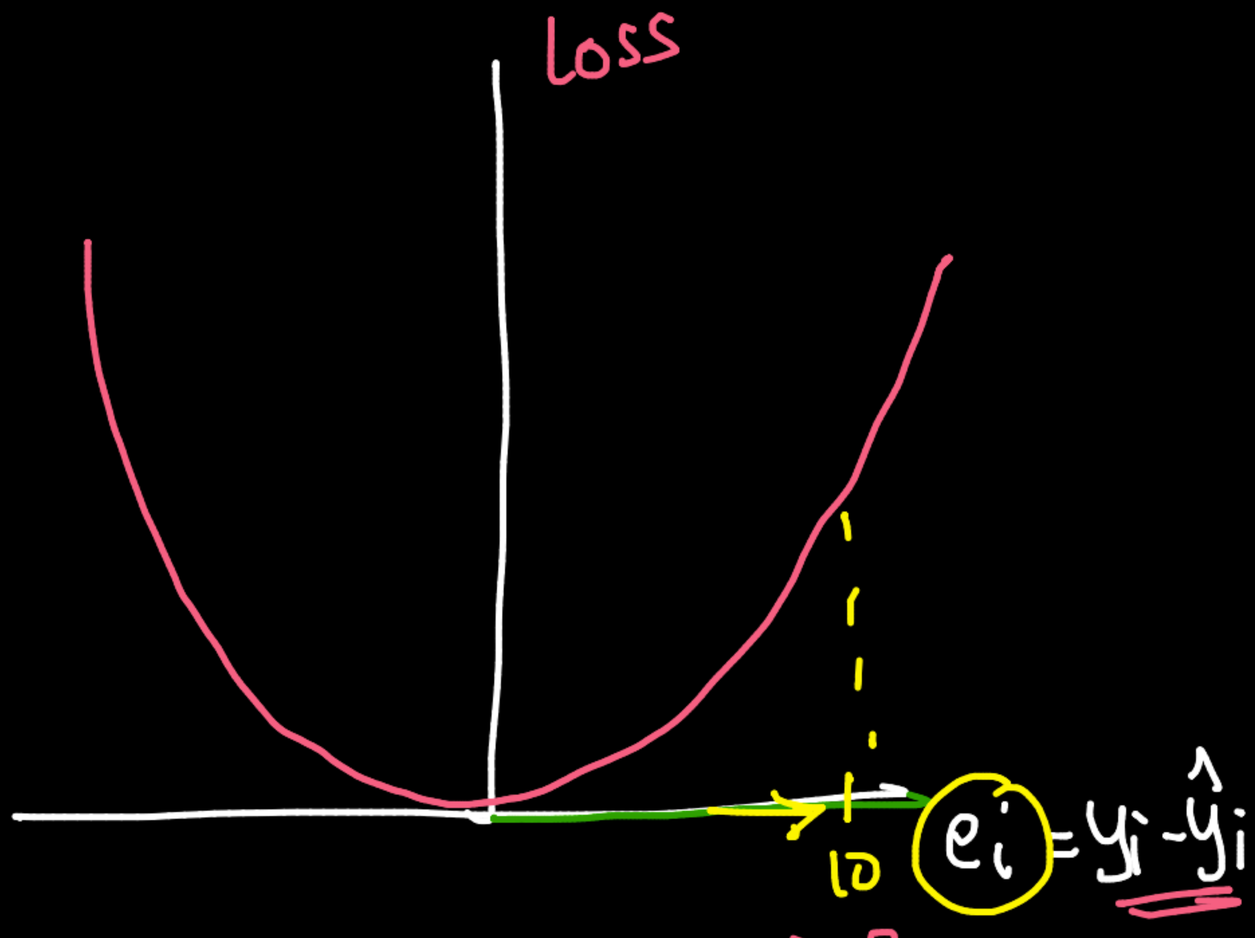
→ each $f_i(x)$

$f_i(x)$: shallow DT


reg:
xi : 2dim
1 2 1 0
yi = 36
errors
Toy-ex:
F0(xi) = 12
24
f1(xi) = 18
6
F2(xi) = 4
2
f3(xi) = 1.9
0.1
26 / 26


RF
Simpler to implement
GBDT
bagging
fasters
- pre XgBoost / LightGBM
2013-14 (FAANG)
40 Scients tEngs


GBDT : regression
(outliers)
remove outliers
Specialized Loss fn
more robust to outliers



SQ-loss: $(y_i - \hat{y}_i)^2$

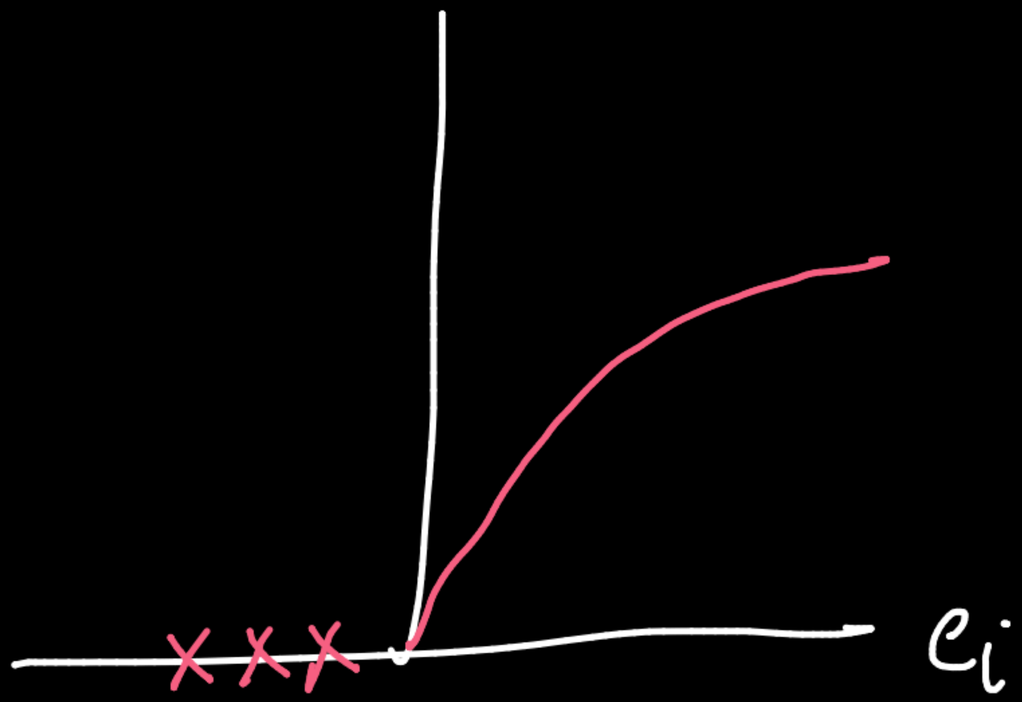
$$\text{loss} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

x_{100}, y_{100} is an outlier ✓

$\hat{y}_{100} - y_{100}$: high

error is accentuated because of the ^{2}

SQRT



doesnt work

✓

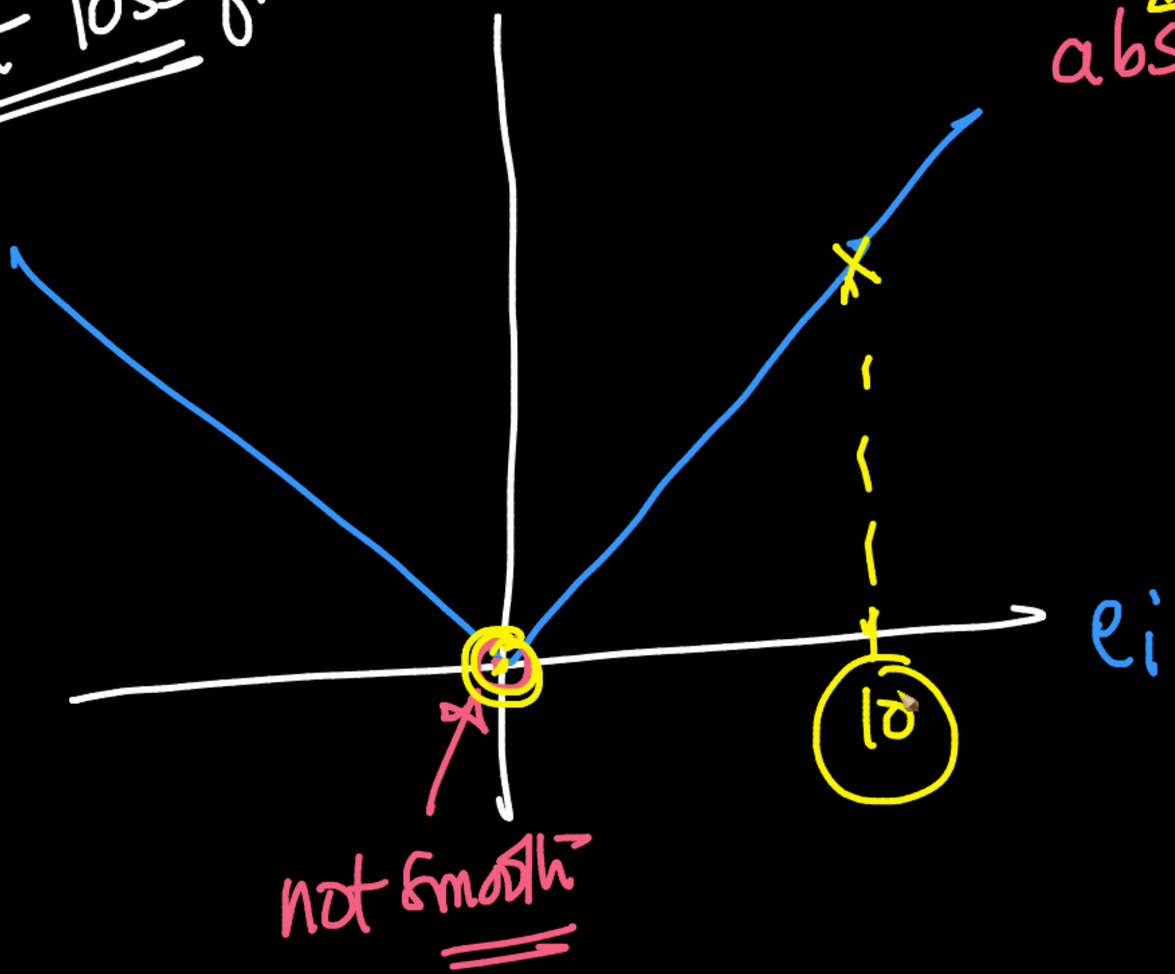
RMSE

$$\sqrt{\frac{1}{n}\left(e_1^2 + e_2^2 + \cdots e_i^2 + \cdots e_n^2\right)}$$

better than Squared loss

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

Robust loss fn:



abs (ei)

$$\min \sum_{i=1}^{n} f(e_i)$$

$$abs(x) = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{cases}$$

hack

$$\frac{d\,abs(x)}{dx} = \begin{cases} +1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}$$

Math: XD

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item

Print/export
Download as PDF
Printable version

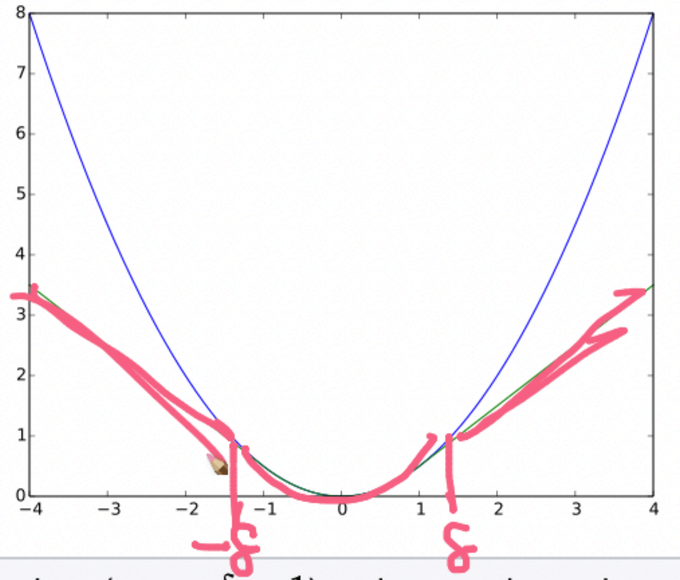
Languages
فارسی
Italiano
日本語
Русский
Edit links

Definition [edit]

The Huber loss function describes the penalty incurred by an estimation procedure f. Huber (1964) defines the loss function piecewise by[1]

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta \cdot \left(|a| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases}$$

This function is quadratic for small values of a, and linear for large values, with equal values and slopes of the different sections at the two points where $|a| = \delta$. The variable a often refers to the residuals, that is to the difference between the observed and predicted values $a = y - f(x)$, so the former can be expanded to[2]



Huber loss (green, $\delta = 1$) and squared error loss (blue) as a function of $y - f(x)$

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta \cdot \left(|y - f(x)| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases}$$

Motivation [edit]

improve this section by

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item

Print/export
Download as PDF
Printable version

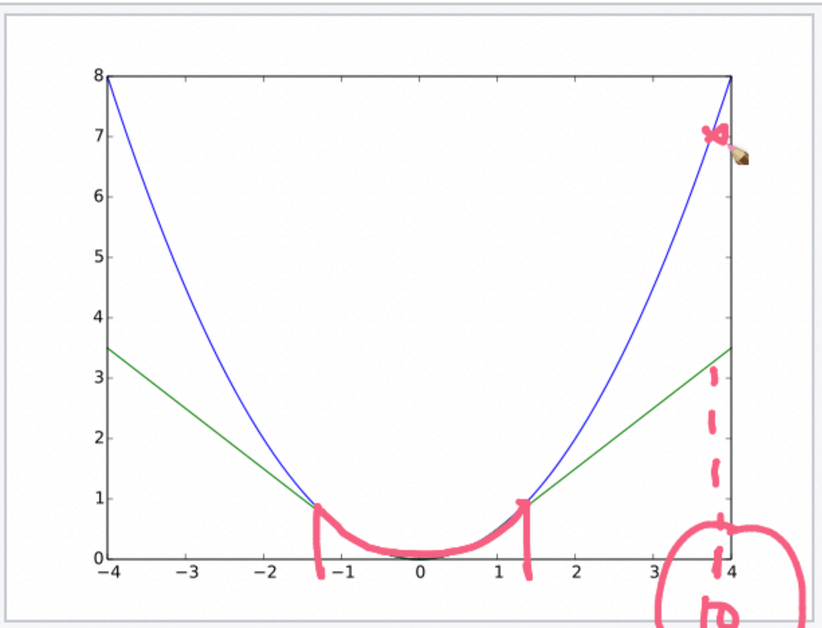
Languages
فارسی
Italiano
日本語
Русский
Edit links

Definition [edit]

The Huber loss function describes the penalty incurred by an estimation procedure f. Huber (1964) defines the loss function piecewise by[1]

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta \cdot \left(|a| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases}$$

This function is quadratic for small values of a, and linear for large values, with equal values and slopes of the different sections at the two points where $|a| = \delta$. The variable a often refers to the residuals, that is to the difference between the observed and predicted values $a = y - f(x)$, so the former can be expanded to[2]



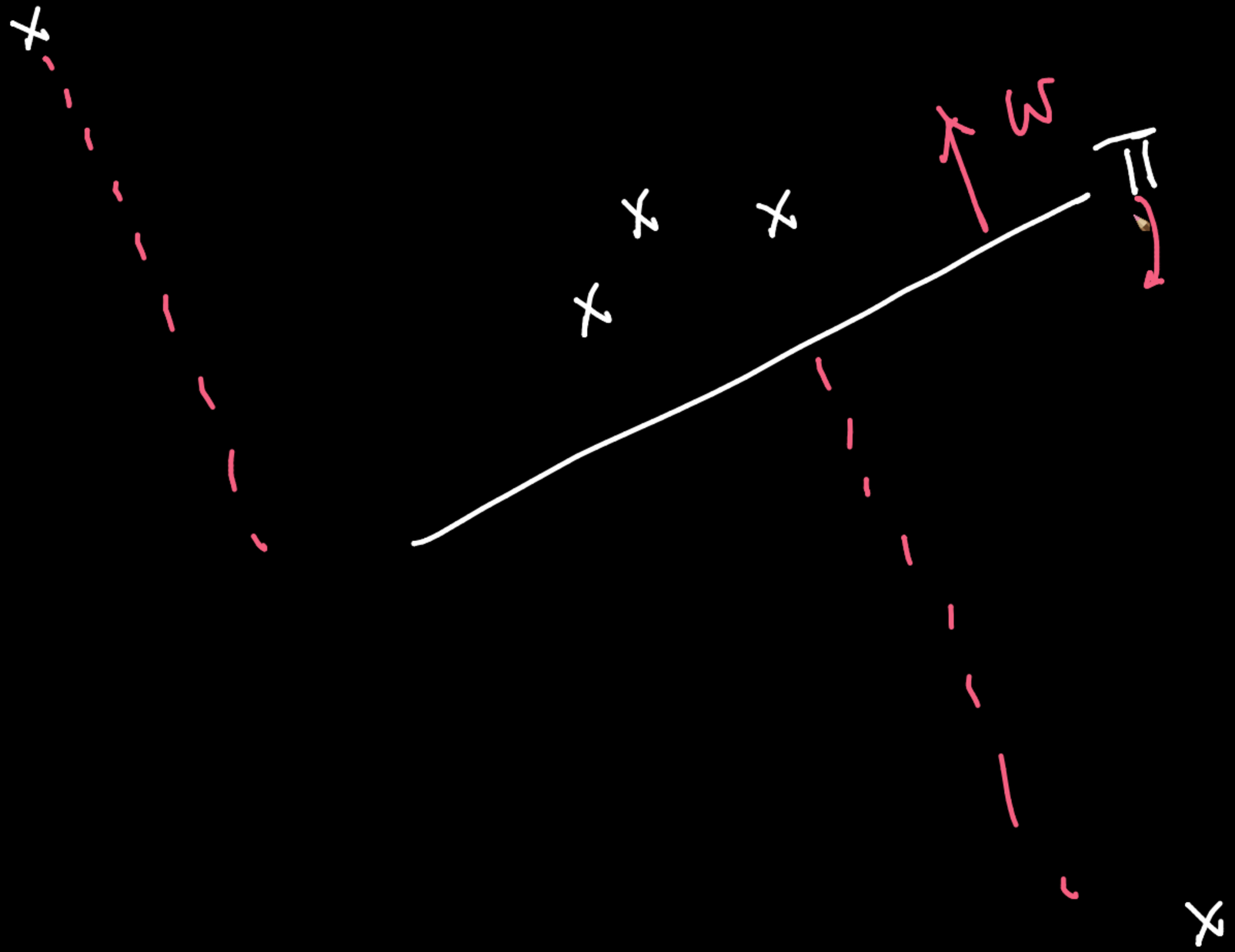
Huber loss (green, $\delta = 1$) and squared error loss (blue) as a function of $y - f(x)$

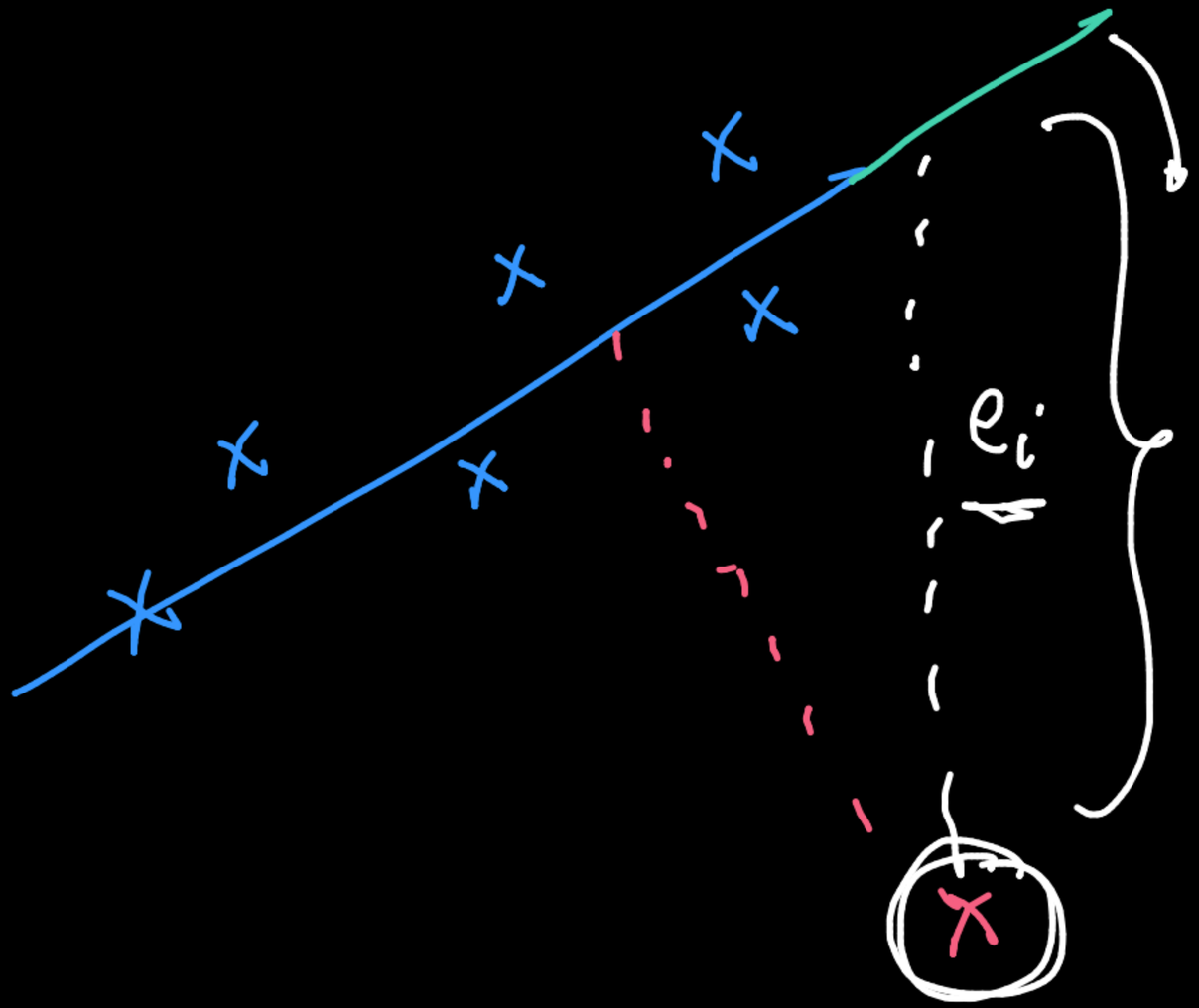
$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta \cdot \left(|y - f(x)| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases}$$

Motivation [edit]

improve this section by

35 / 35


w



huber loss

TABLE 10.2. *Gradients for commonly used loss functions.*

Setting	Loss Function	$-\partial L(y_i, f(x_i))/\partial f(x_i)$
Regression	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$
Regression	$\lvert y_i - f(x_i)\rvert$	$\text{sign}[y_i - f(x_i)]$
Regression	Huber	$y_i - f(x_i)$ for $\lvert y_i - f(x_i)\rvert \le \delta_m$ $\delta_m \text{sign}[y_i - f(x_i)]$ for $\lvert y_i - f(x_i)\rvert > \delta_m$ where $\delta_m = \alpha$th-quantile$\{\lvert y_i - f(x_i)\rvert\}$
Classification	Deviance	kth component: $I(y_i = \mathcal{G}_k) - p_k(x_i)$

boosting procedure, and top-down decision tree induction, are themselves approximation procedures. After constructing the tree (10.37), the corresponding constants in each region are given by (10.30).

Table 10.2 summarizes the gradients for commonly used loss functions. For squared error loss, the negative gradient is just the ordinary residual $-g_{im} = y_i - f_{m-1}(x_i)$, so that (10.37) on its own is equivalent standard least squares boosting. With absolute error loss, the negative gradient is the *sign* of the residual, so at each iteration (10.37) fits the tree to the sign of the current residuals by least squares. For Huber M-regression, the table).

nce (10.22),

Cascading

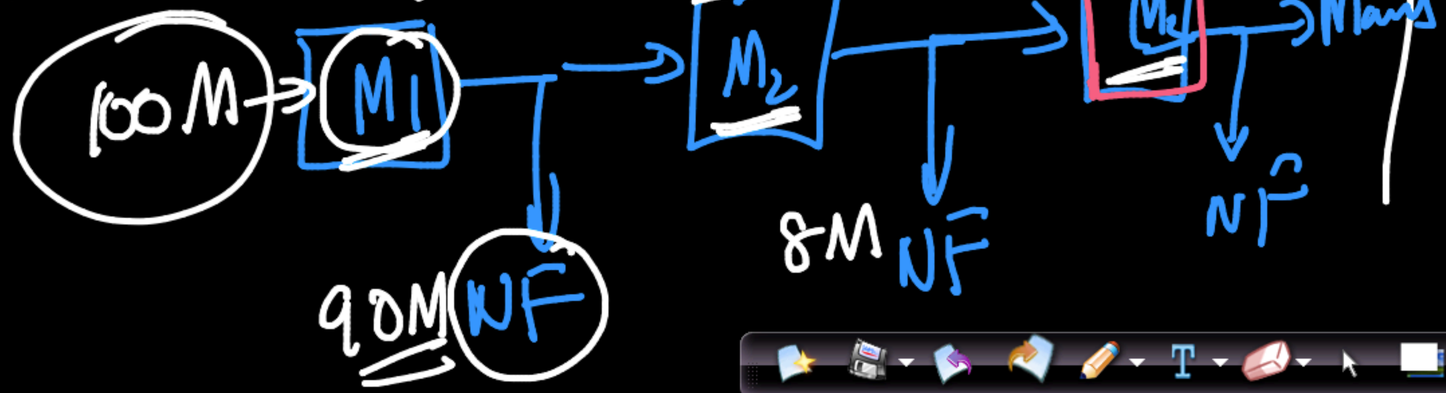
Fraud-detn..

(TRMS)

credit

↳ risk/cost of mistakes is high

imbalanced



Stacking

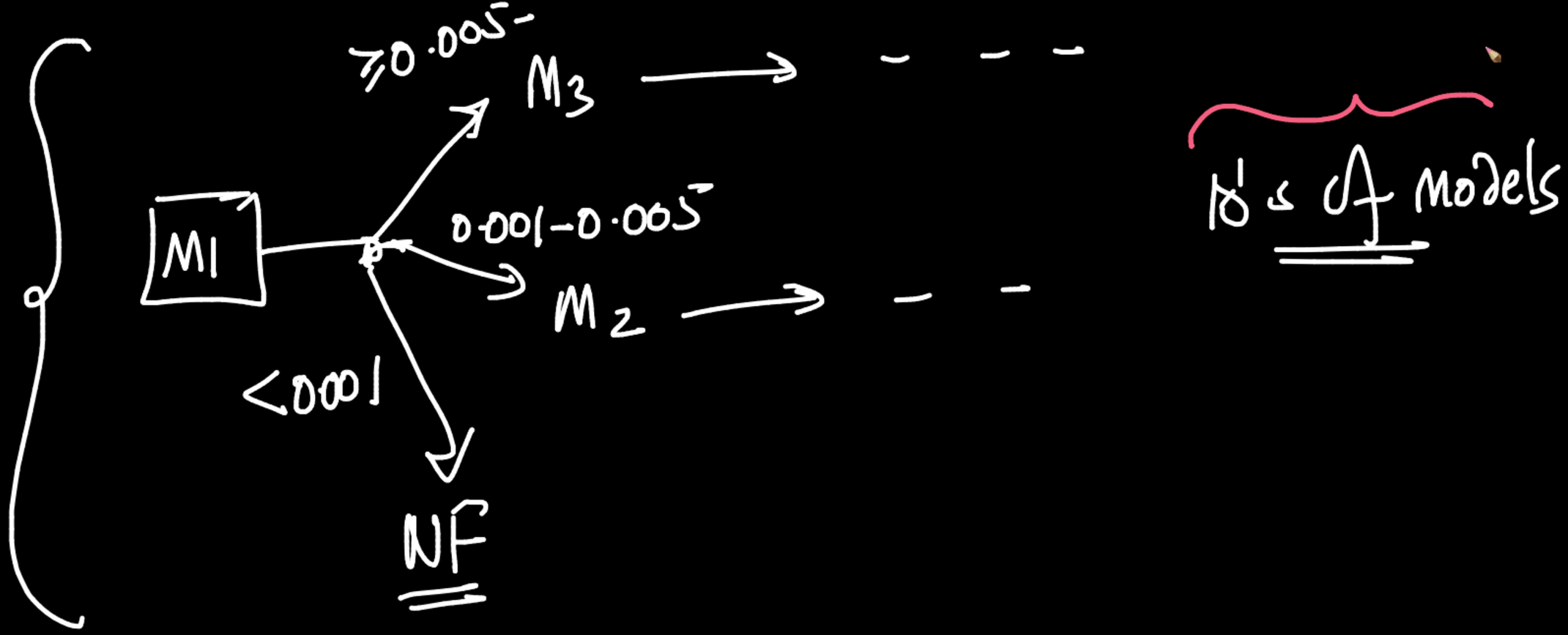
Kaggle - comp

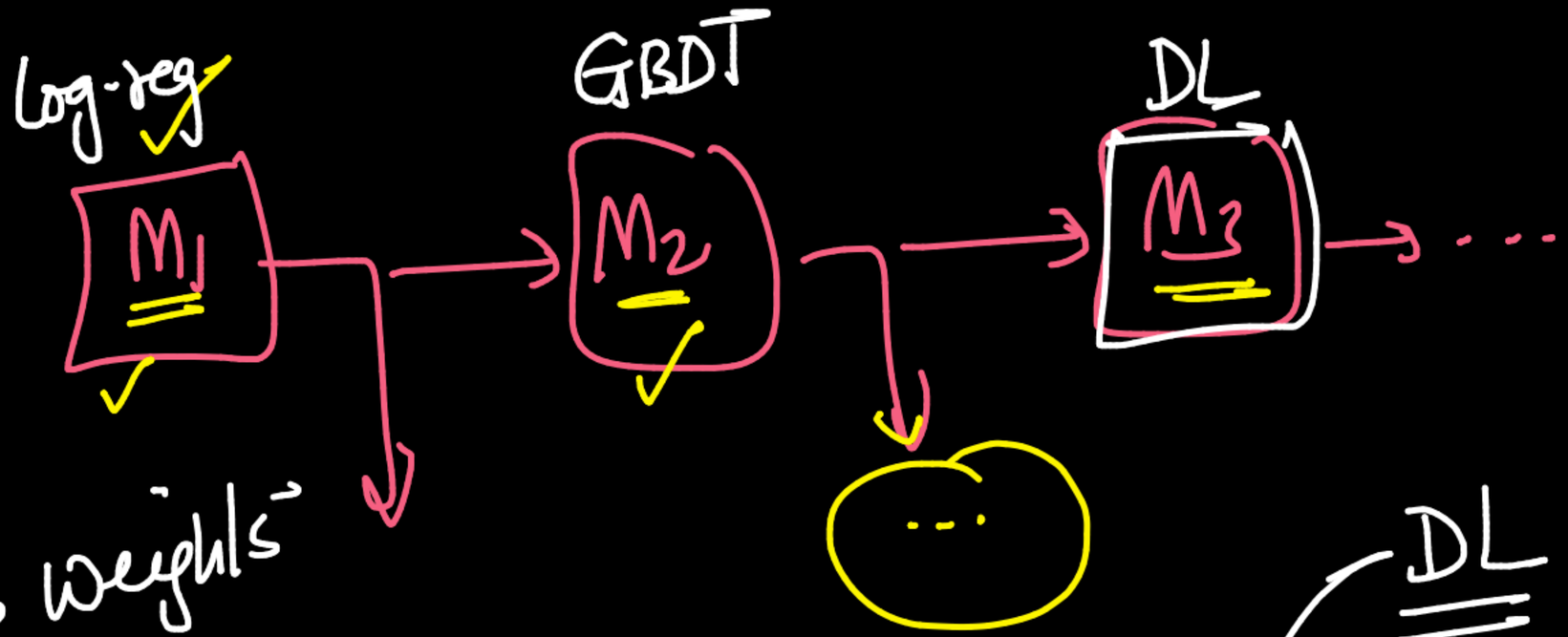
AUC:→

1st : 0.8234

2nd: 0.8231

↳ DL learning


≥0.005
M3
M1
0.001-0.005
M2
<0.001
NF
18 cA models



Log-Reg → weights

DT → if...else

RF/GBDT → tf.!

[explainability → LIME & SHAP] DL

Kaggle vs Real-world

→ Discussion forums & Winners Solns...

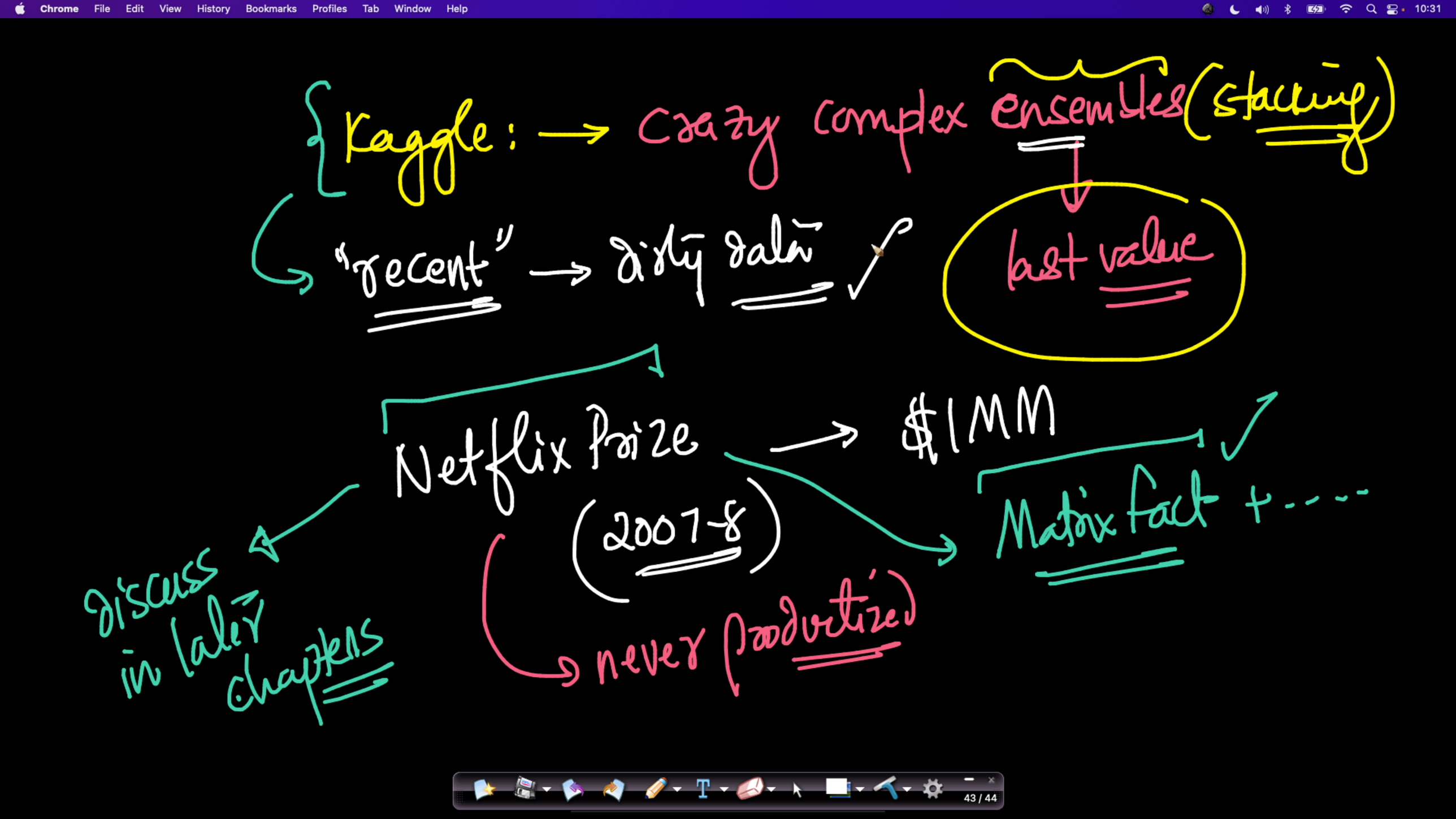
→ One-metric → Squeeze out value

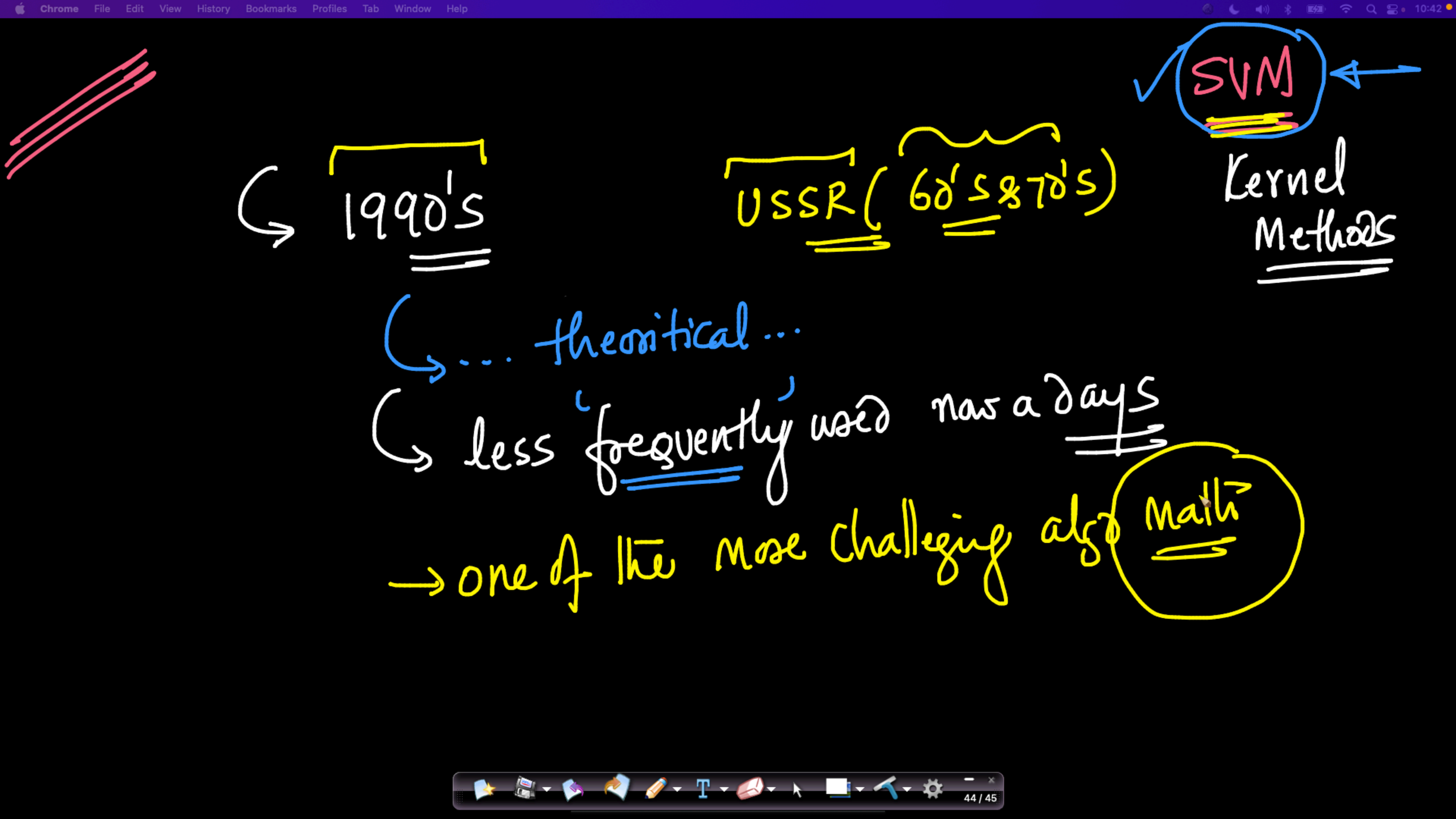
→ multiple things → Interpretability

→ Speed / latency

→ tSNE ; GBDT ; CNN (DL)

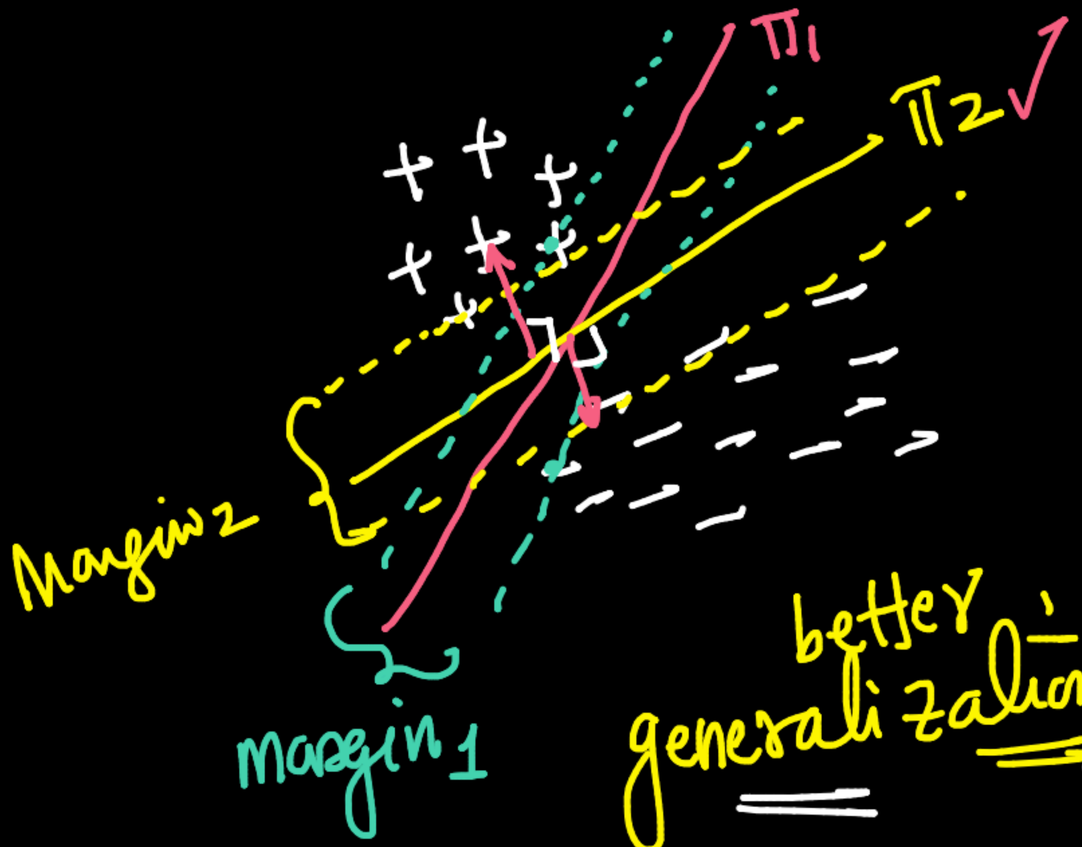
→ metrics { F1 & biz-metrics


Kaggle: → crazy complex ensembles (stacking)
"recent" → dirty data
last value
Netflix Prize → $1MM
(2007-8)
Matrix fact + ----
never productized
discuss in later chapters


SVM
Kernel Methods
1990's
USSR (60's & 70's)
... theoritical ...
less 'frequently' used nowadays
one of the most challenging algo
Math
44 / 45

Geom:

{ pick Π that results in largest margin



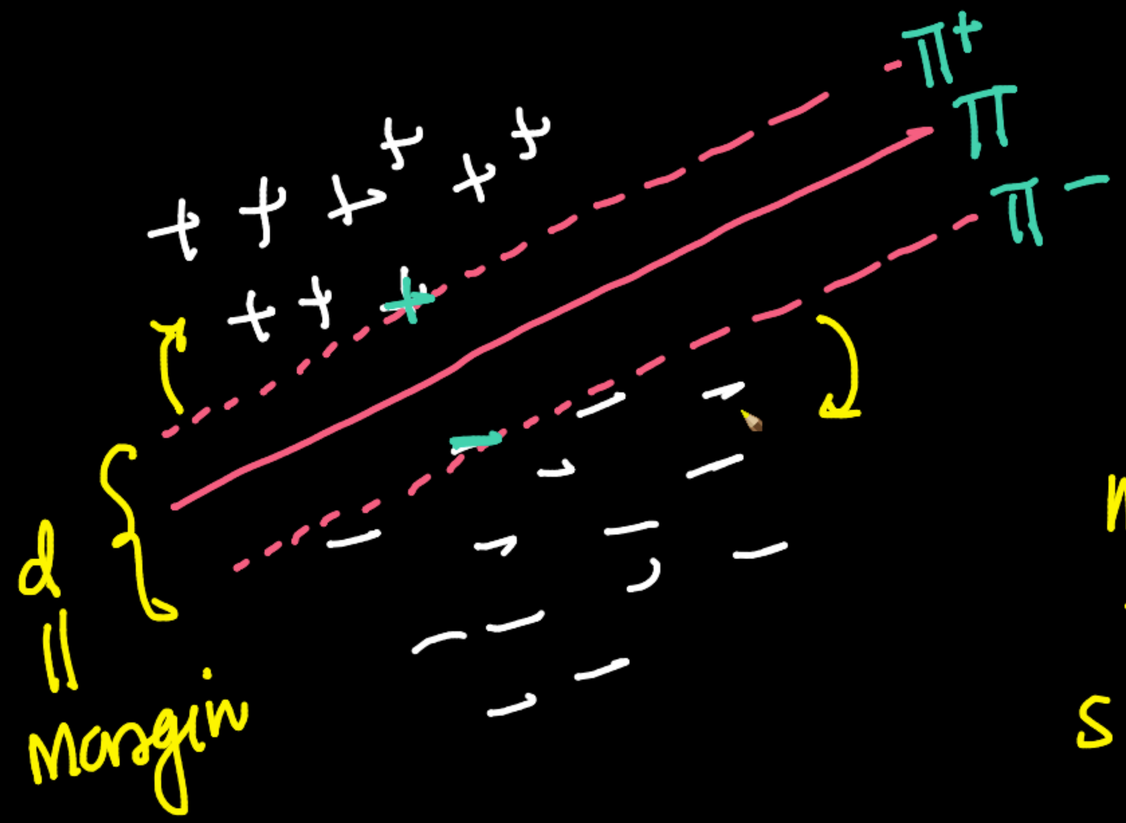
better generalization ✓

{ Margin-Maximising Classifier

π: Seperating hyperplane

$\pi^+ \parallel \pi$

$\pi^- \parallel \pi$



Max. Margin

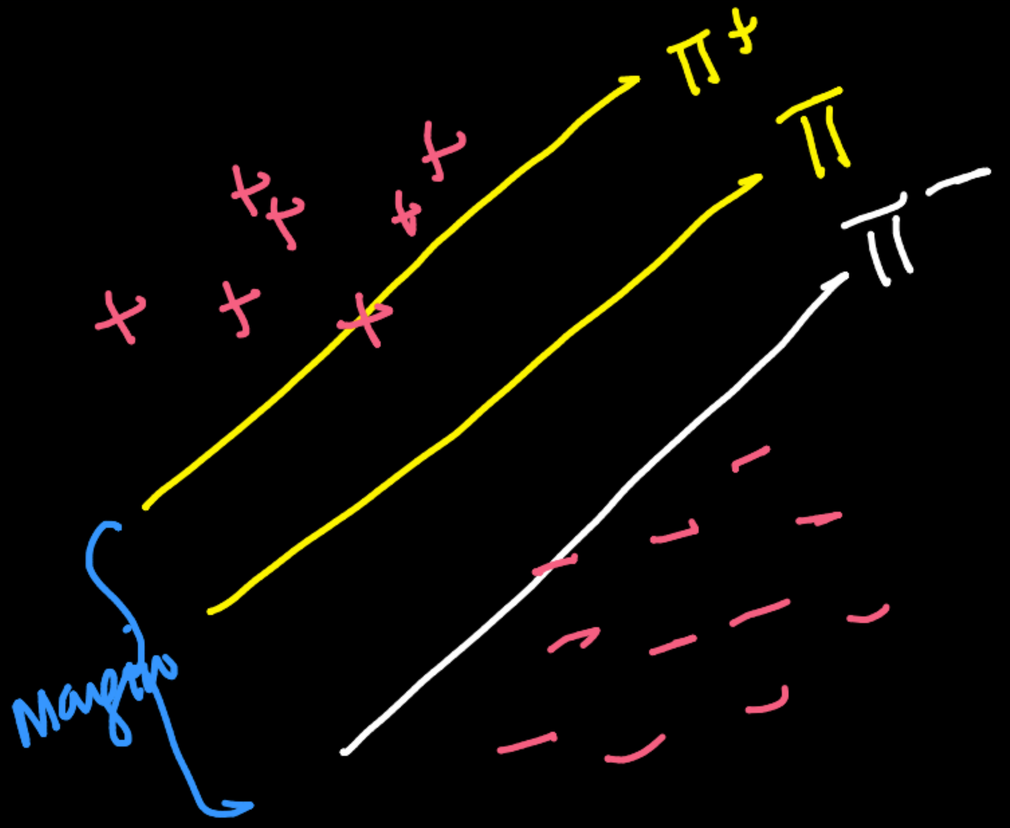
s.t

(Let) $\|w\| \neq 1$

π: $w^T x + b = 0$

π^+: $w^T x + b = 1$

π^-: $w^T x + b = -1$



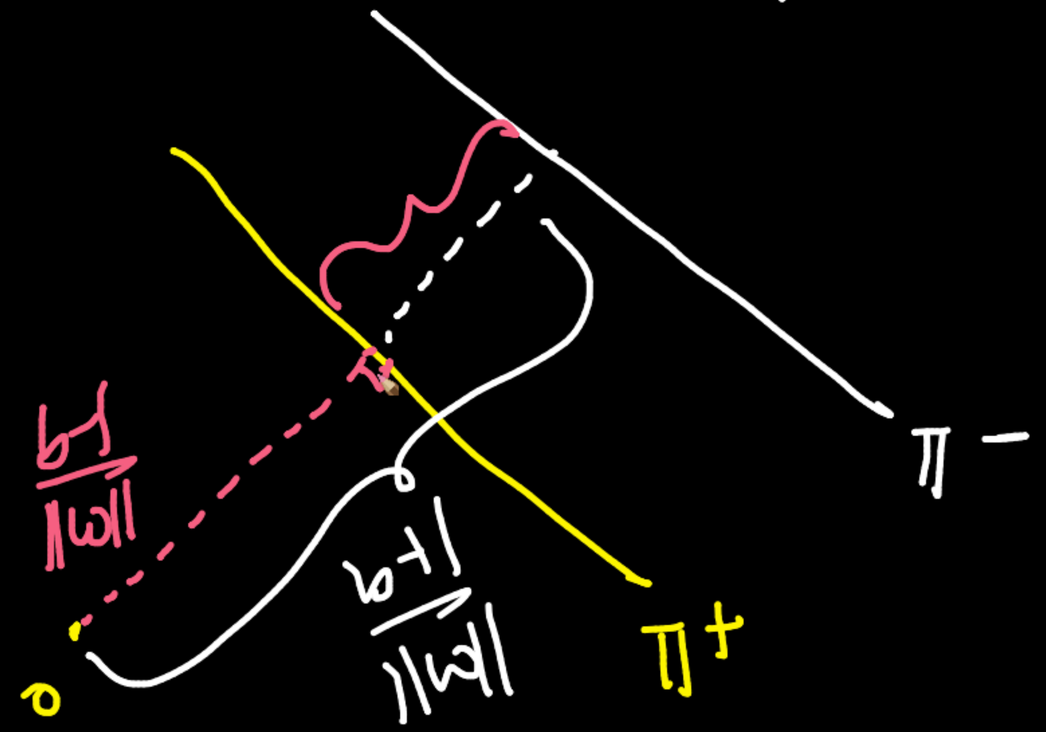
$$\text{margin} = \frac{2}{\|w\|}$$

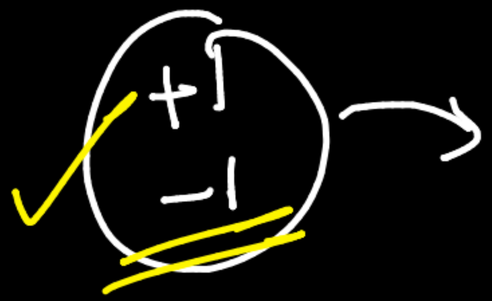
[Simple case: Linearly Seperable data

$$\pi^+ : w^T x + (b-1) = 0$$

$$\pi^- : w^T x + (b+1) = 0$$

$$d(\pi^+, \pi^-) = \frac{2}{\|w\|}$$

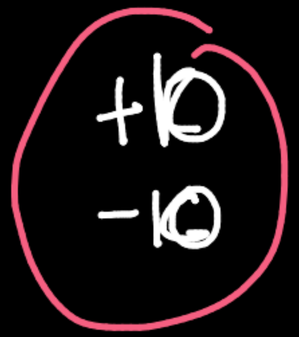


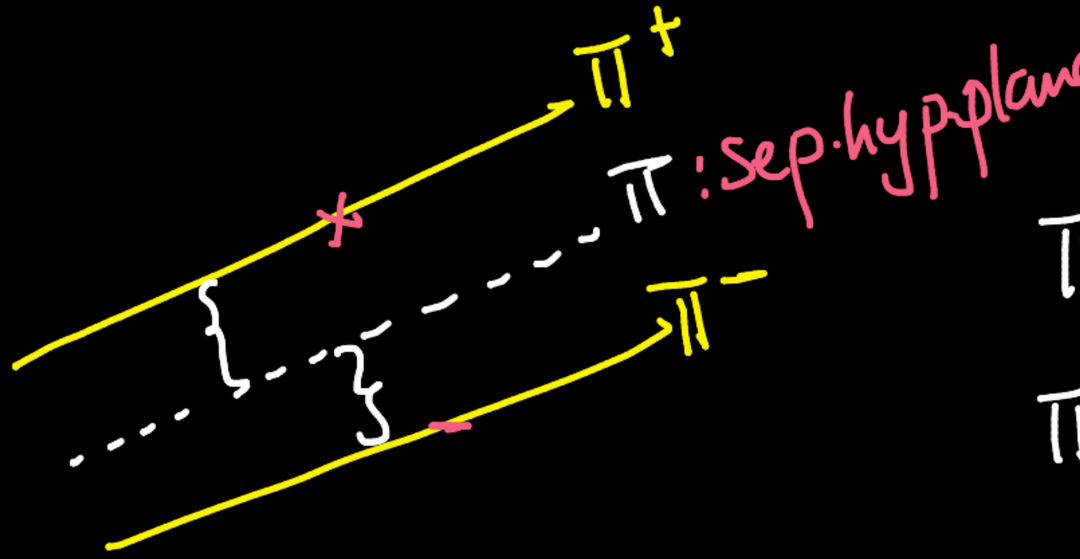


$$\max_{w,b} \frac{2}{\|w\|}$$

$$=$$

$$\max_{w,b} \frac{20}{\|w\|}$$





$\pi^+ : w^T x + b = +10$

$\pi^- : w^T x + b = -10$

$$\text{Margin} = \frac{20}{\|w\|}$$

Hard - Margin

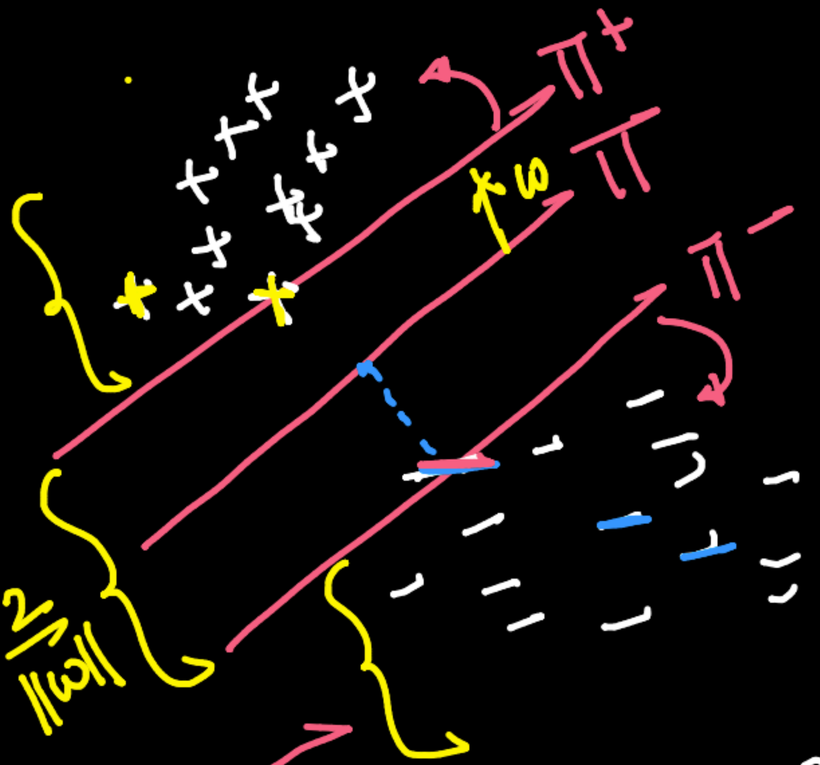
Simplest version

Lr-SVM

$$\max_{w,b} \frac{2}{\|w\|}$$

$$\text{s.t} \quad (w^T x_i + b) y_i \geq 1 \quad \forall i : 1 \to n$$

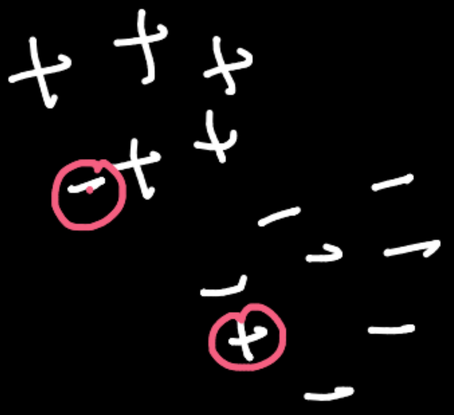
n constraints



$$\mathcal{D} = \{ (x_i, y_i) \}_{i=1}^{n}$$

$+1$ -1

When would this fail : Hard Margin SVM :-



→ data is non-lr. Sep:

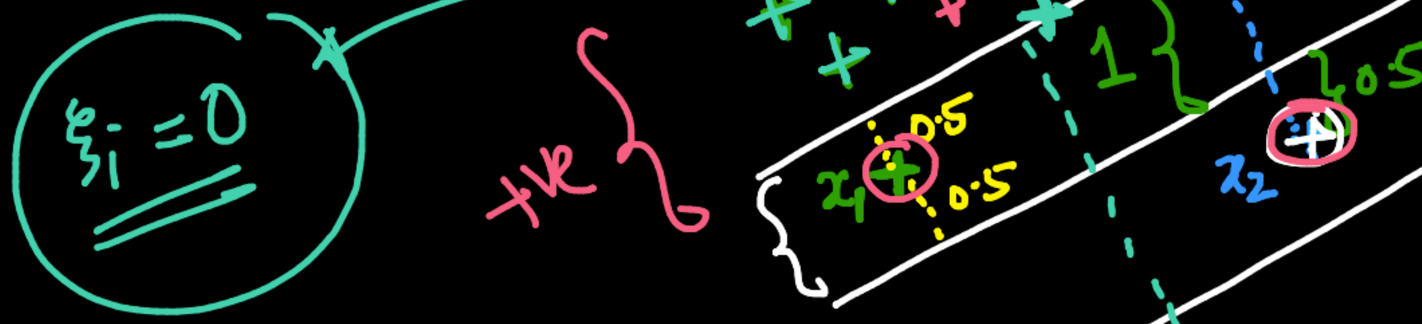
$$\pi : \underline{\theta}^T x + \underline{\theta_0}$$

x_1: $(w^T x_i + b) y_i = 0.5 = 1 - \boxed{0.5} \rightarrow \xi_1$

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

$\pi : w^T x + b = 0$

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



x_2: $(w^T x_i + b) y_i = -0.5$

$= 1 - \boxed{1.5} \rightarrow \xi_2$

x_3: $(w^T x_i + b) y_i$

$= 1 - \boxed{2.5} \rightarrow \xi_3$

$\xi_i = 0$ for all 'correctly' placed pts → hard-margin

$\xi_i > 0$ for all 'incorrectly' placed pts

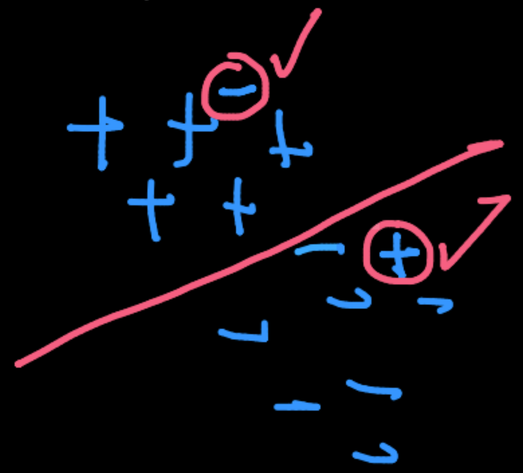
↳ $\xi_i < 1$ → it will still be classified correctly by π

$\xi_i > 1$ → incorrectly classified

Soft-Margin SVM

$$\max \frac{2}{\|w\|} = \sqrt{\min \frac{\|w\|}{2}}$$

not perfectly Lr-seperable



almost linear Seperable

Soft Margin
lr- SVM

$$\min_{w,b} \left[\frac{\|w\|}{2} + C \sum_{i=1}^{n} \xi_i \right]$$

HP

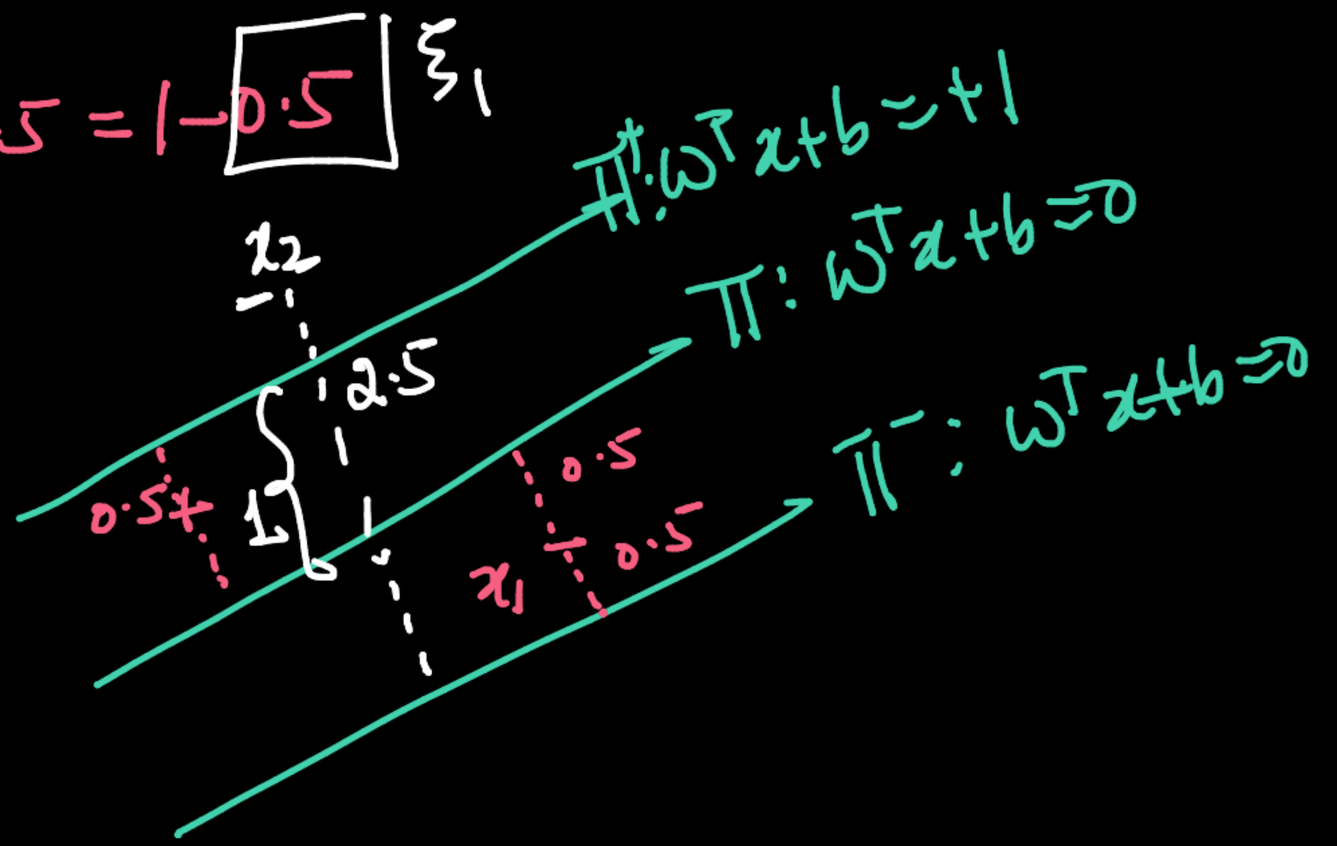
s.t $y_i (w^T x_i + b) \geq 1 - \xi_i$ $\forall i : 1 \to n$

$\xi_i \geq 0$ $\forall i : 1 \to n$

for +ve & -ve pts

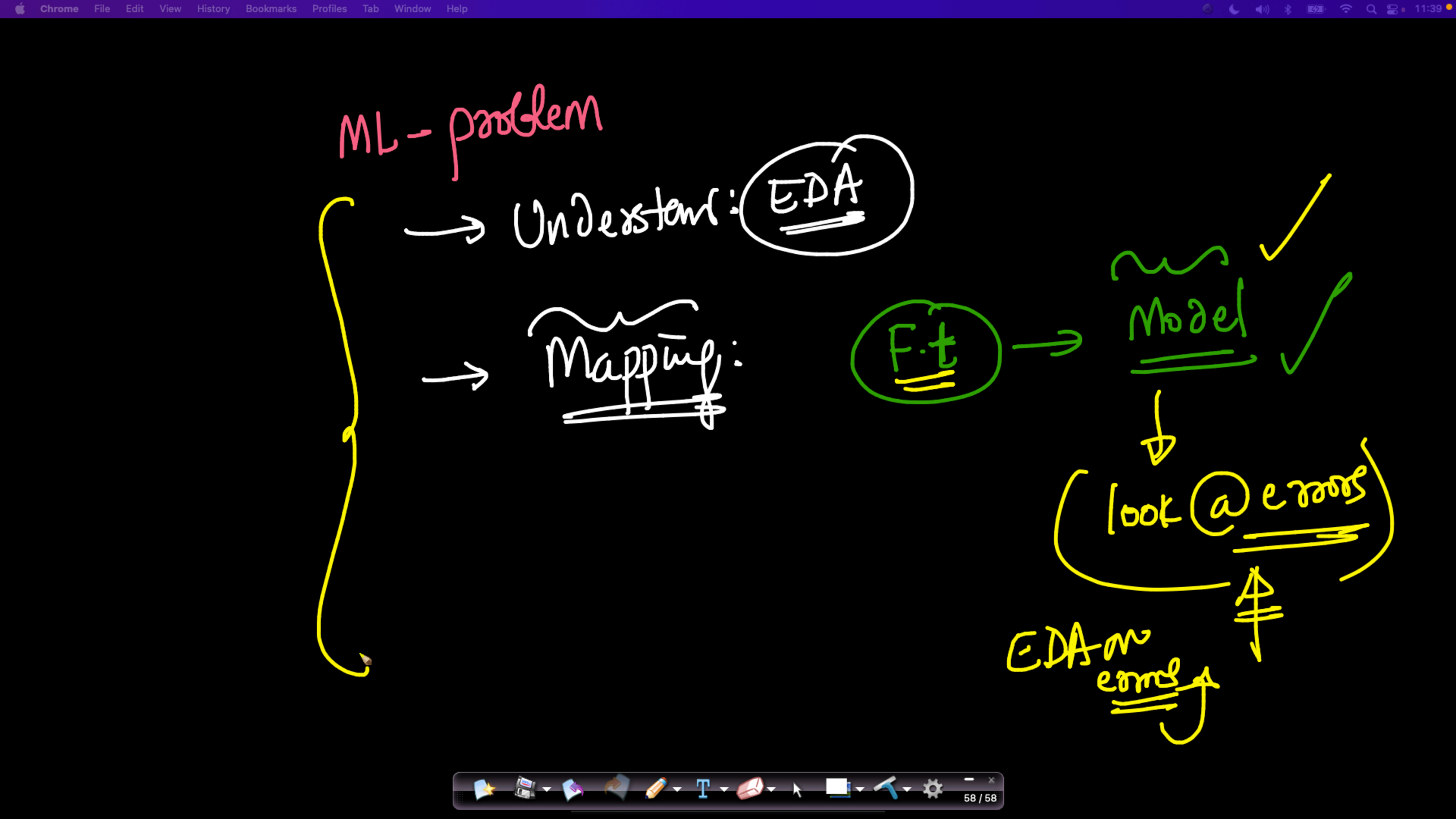
Linear classifn
[almost lr-separable]

x_1: $y_i(w^T x_i + b) = 0.5 = 1 - \boxed{0.5}\ \xi_1$ (-ve, -ve)



x_2: $y_i(w^T x_i + b) = -1.5$

$= 1 - \boxed{2.5}\ \xi_2$


ML - problem
Understand: EDA
Mapping:
Fit
Model
look @ errors
EDA on errors
58 / 58

linear model

$d = (n)$

$n = 1000$

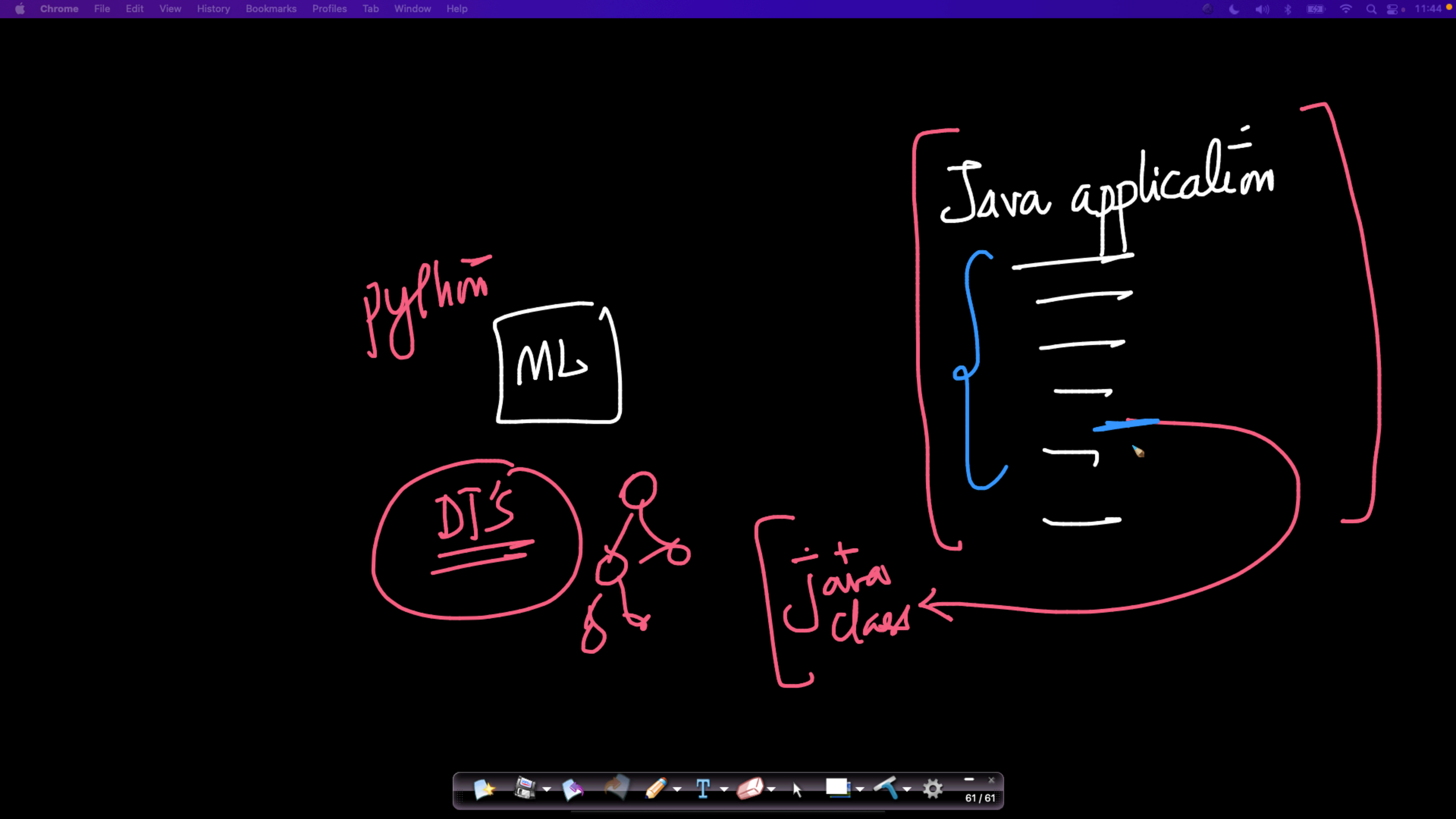
"regularize" - heavily

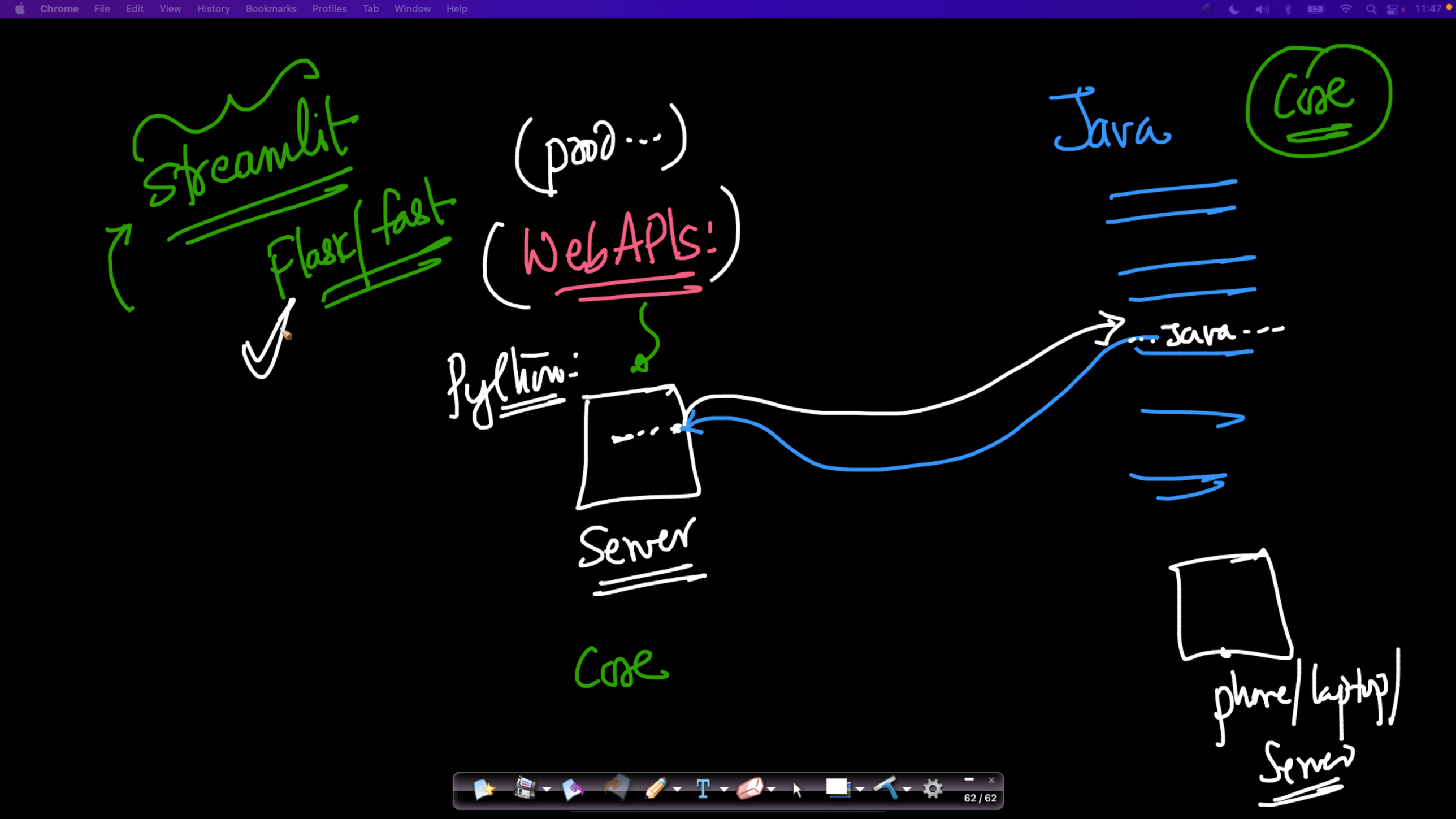
PCA → $n : ?0$ (May not always work)

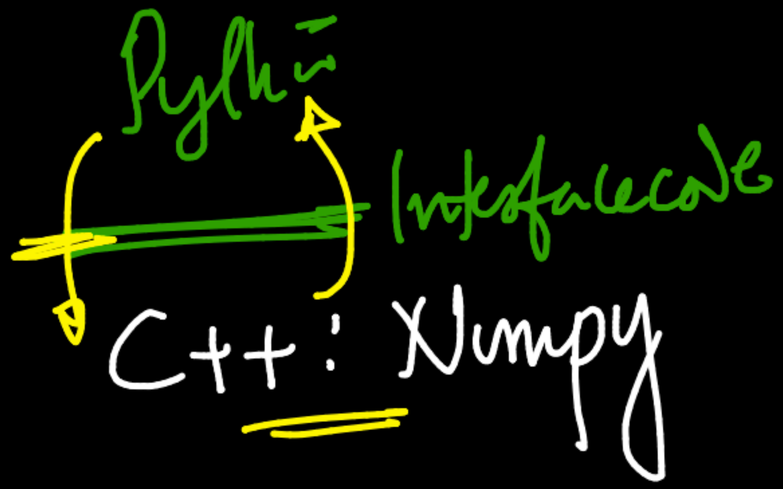
y_i's

ML

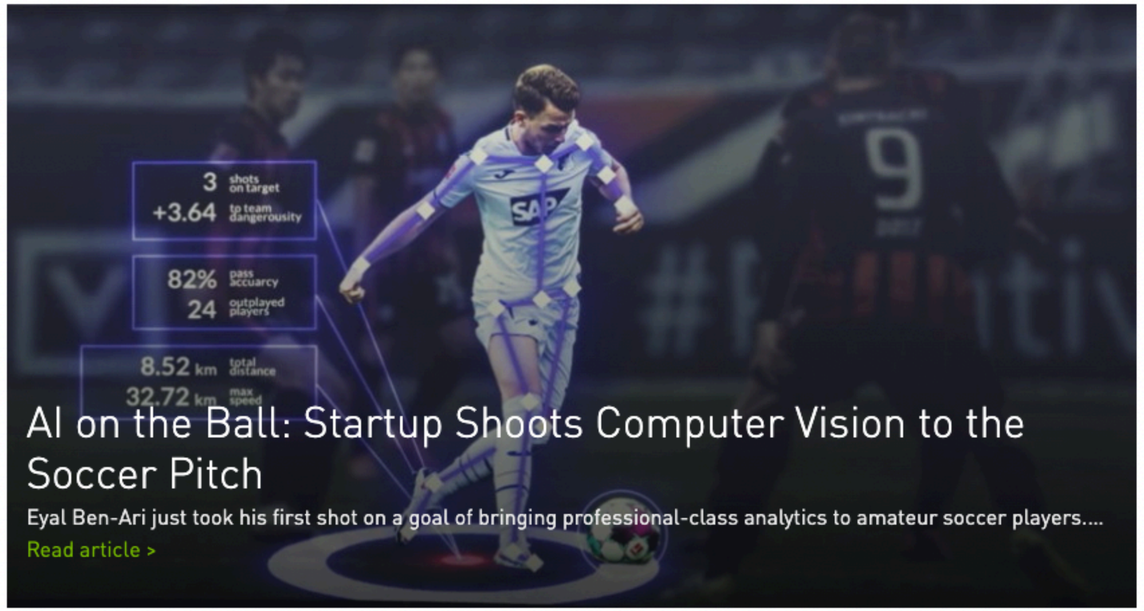

60 / 60


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Java applicalion
Pylhon
ML
DT's
Java class
61 / 61


Chrome File Edit View History Bookmarks Profiles Tab Window Help
11:47
Streamlit
Flask/fast
(pood...)
(Web APIs:)
Java
Case
Python:
...
Server
...Java...
Case
phone/laptop/
Server
62 / 62


Pyth
Interface code
C++ : Numpy

DEEP LEARNING



AI on the Ball: Startup Shoots Computer Vision to the Soccer Pitch

Eyal Ben-Ari just took his first shot on a goal of bringing professional-class analytics to amateur soccer players....

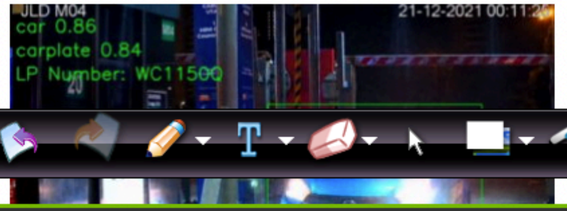
Read article >

MOST POPULAR

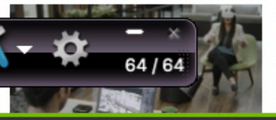
What is Extended Reality?

Mission Made Possible: Real-Time Rendering Helps Studio Create Cinematic Battle Between Characters From 'Diablo Immortal'

AI on the Ball: Startup Shoots Computer Vision to the Soccer Pitch



ALL NVIDIA NEWS



What is Extended Reality?