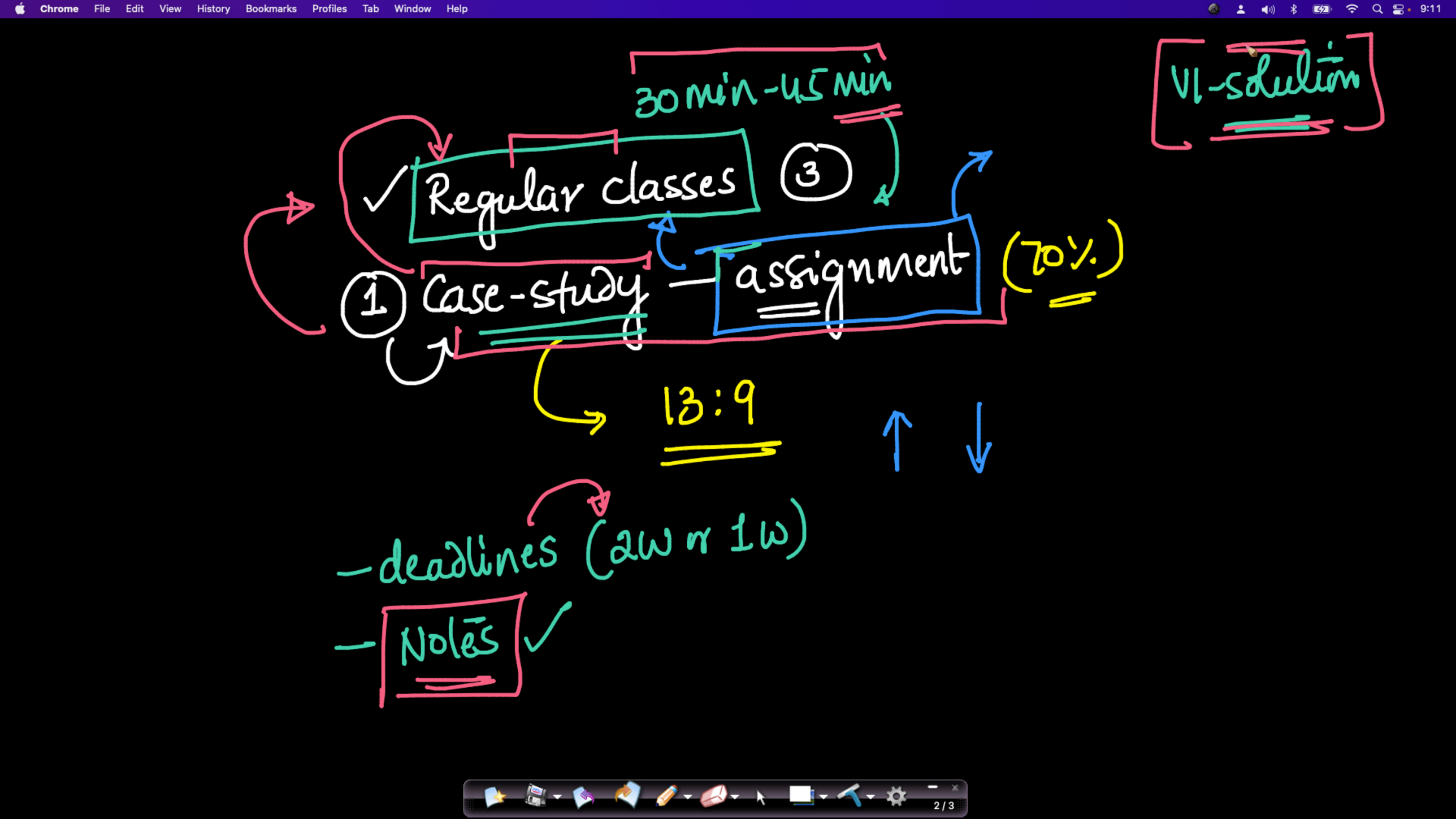
Topics:

1. Case-study: Bio-signals — EMG → DT
2. { Code for XgBoost
3. ✓ K-fold CV
4. ✓ GridSearch & RandomSearch
5. Feature Importance : GBDT
6. Cascading ✓
7. Stacking ✓
8. Q&A

→ $\left[\frac{-\partial L}{\partial f(x)} \approx \text{residuals}\right]$


Chrome File Edit View History Bookmarks Profiles Tab Window Help
9:11
30 min - 45 min
VI-solution
Regular classes
3
1
Case-study
assignment
(70%)
13:9
- deadlines (2w or 1w)
- Notes
2 / 3

Chrome File Edit View History Bookmarks Profiles Tab Window Help 9:12

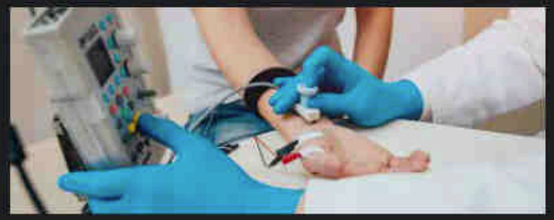
XgBoost.ipynb - Colaboratory | EMG - Google Search | EMG signal process recommen | ESLII.pdf | StackingClassifier: Simple sta | StackingCVClassifier: Stacking

google.com/search?q=EMG&rlz=1C5CHFA_enIN958IN958&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiCqKGTqen3AhVnmFYBHXbHCa8Q_AUoAXoECAIQAw&biw=1745&bih=887&dpr=1.1#imgrc=SQH5vh9ifw7R0M

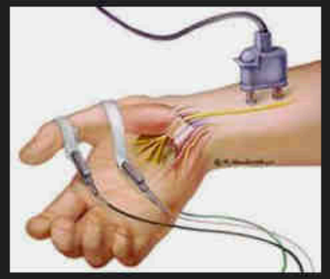
Update

Google EMG

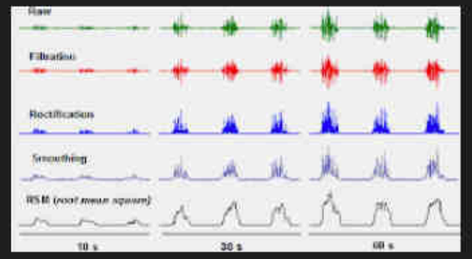
Electromyography (EMG): Purpose ...
healthline.com



Electromyography (EMG) | Johns Hopkins ...
hopkinsmedicine.org

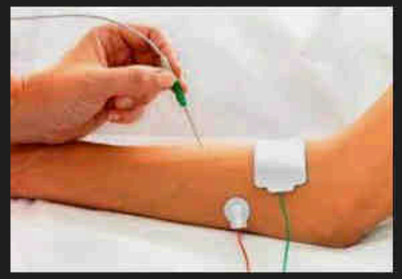


EMG
ascentmc.com



EMG signal process recommended. Green ...
researchgate.net

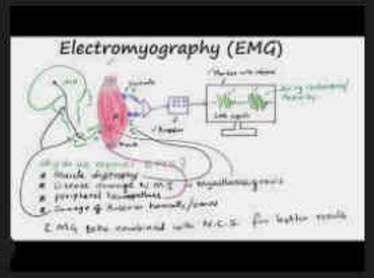
How can EMG and NCV tests help you?: Dr ...
chicagoneurodoc.com



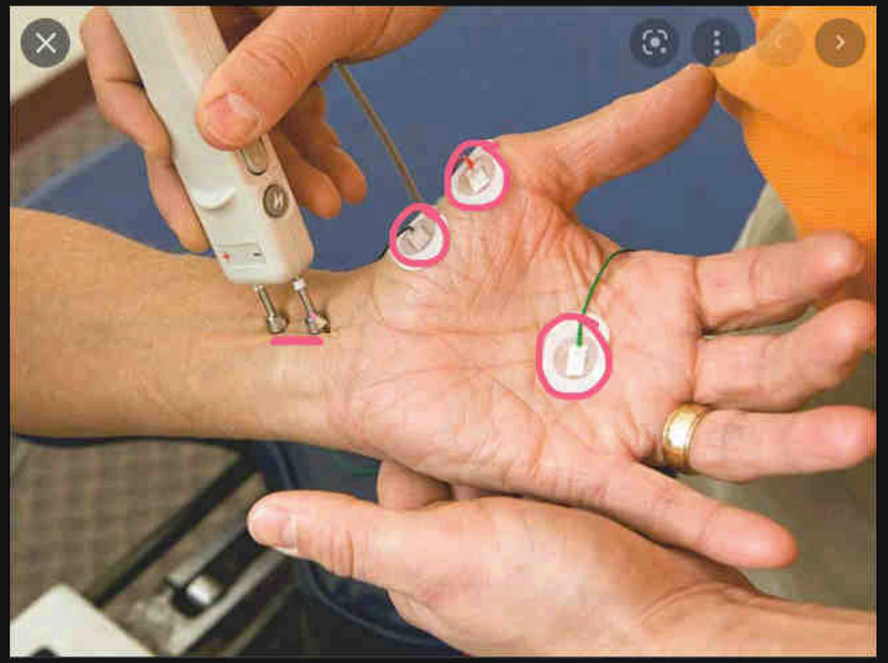
EMG | Technomed
technomed.nl

Electromyography (EMG) | Nerve ...
neurologist-ahmedabad.com

What is EMG? When is it done?
istanbulonkoloji.com



EMG II Electromyography II Muscle ...
youtube.com



Healthline
Electromyography (EMG): Purpose, Procedure, and Results
Visit

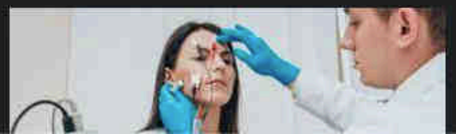
Images may be subject to copyright. Learn More

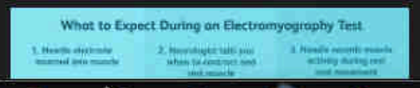
Related images See more

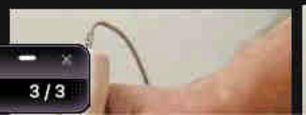
Nerve Conduction Studies - Pre...
precisionhealth.com.au

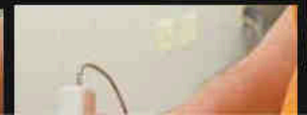
Electromyogram (EMG)
hellodox.com

EMG Full Form - javatpoint









3 / 3

Segment	R-Arm	L-Arm	R-Leg	L-Leg
Channel	ch1	ch2	ch3	ch4	ch5	ch6	ch7	ch8
Muscle	R-Bic	R-Tri	L-Bic	L-Tri	R-Thi	R-Ham	L-Thi	L-Ham
Column	0	1	2	3	4	5	6	7

Segment: A segment defines a body segment or limb.
- Right arm (R-Arm)
- Left arm (L-Arm)
- Right leg (R-Leg)
- Left leg (L-Leg)

Channel: A channel corresponds to an electrode attached on a muscle.

Muscle: A pair of muscles that corresponds to a segment.
- R-Bic: right bicep (C1)
- R-Tri: right tricep (C2)
- L-Bic: left bicep (C3)
- L-Tri: left tricep (C4)
- R-Thi: right thigh (C5)
- R-Ham: right hamstring (C6)
- L-Thi: left thigh (C7)
- L-Ham: left hamstring (C8)

Relevant Papers:

N/A

Citation Request:

```
# EMG Data Analysis
# Source: https://archive.ics.uci.edu/ml/datasets/EMG+Physical+Action+Data+Set
```

```
#How was the data collected ?
## Person/Subject was asked to perform specific physical actions
## Signals produced due to that movement were recorded over time.
## 8 channels were used to record the signals
## Channels here correspond to muscles\ For eg: Right-hand bicep
## Frequency : 10 per ms
```

```
!wget "https://drive.google.com/uc?export=download&id=1ZgZYFdMlQyccRtHs16AhJUEnLPhgjCc7" -O EMG_data.rar
```

```
--2022-05-18 13:05:36--  https://drive.google.com/uc?export=download&id=1ZgZYFdMlQyccRtHs16AhJUEnLPhgjCc7
Resolving drive.google.com (drive.google.com)... 108.177.119.102, 108.177.119.139, 108.177.119.113, ...
Connecting to drive.google.com (drive.google.com)|108.177.119.102|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://doc-0o-14-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc7l7deffksulhg5h7mbp1/bkumsf47lms7nj
Warning: wildcards not supported in HTTP.
--2022-05-18 13:05:42--  https://doc-0o-14-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc7l7deffksulhg5h7mbp1
Resolving doc-0o-14-docs.googleusercontent.com (doc-0o-14-docs.googleusercontent.com)... 142.250.145.132, 2a00:1450:
Connecting to doc-0o-14-docs.googleusercontent.com (doc-0o-14-docs.googleusercontent.com)|142.250.145.132|:443... co
HTTP request sent, awaiting response... 200 OK
```

```
# EMG Data Analysis
# Source: https://archive.ics.uci.edu/ml/datasets/EMG+Physical+Action+Data+Set
```

```
#How was the data collected ?
## Person/Subject was asked to perform specific physical actions
## Signals produced due to that movement were recorded over time.
## 8 channels were used to record the signals
## Channels here correspond to muscles\ For eg: Right-hand bicep
## Frequency : 10 per ms
```


```
!wget "https://drive.google.com/uc?export=download&id=1ZgZYFdMlQyccRtHs16AhJUEnLPhgjCc7" -O EMG_data.rar
```

```
--2022-05-18 13:05:36--  https://drive.google.com/uc?export=download&id=1ZgZYFdMlQyccRtHs16AhJUEnLPhgjCc7
Resolving drive.google.com (drive.google.com)... 108.177.119.102, 108.177.119.139, 108.177.119.113, ...
Connecting to drive.google.com (drive.google.com)|108.177.119.102|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://doc-0o-14-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc7l7deffksulhg5h7mbp1/bkumsf47lms7nj
Warning: wildcards not supported in HTTP.
--2022-05-18 13:05:42--  https://doc-0o-14-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc7l7deffksulhg5h7mbp1
Resolving doc-0o-14-docs.googleusercontent.com (doc-0o-14-docs.googleusercontent.com)... 142.250.145.132, 2a00:1450:
Connecting to doc-0o-14-docs.googleusercontent.com (doc-0o-14-docs.googleusercontent.com)|142.250.145.132|:443... co
HTTP request sent, awaiting response... 200 OK
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help

XgBoost.ipynb - Colaboratory | EMG - Google Search | EMG signal process recommen | ESLII.pdf | StackingClassifier: Simple sta | StackingCVClassifier: Stacking | UCI Machine Learning Reposit

colab.research.google.com/drive/1oOTcIDDRBX1mB3EZTsoYhv4S6Pk7TbjN#scrollTo=J9y5YFVFw8t6

+ Code + Text Reconnect

```
# EMG Data Analysis
# Source: https://archive.ics.uci.edu/ml/datasets/EMG+Physical+Action+Data+Set
```

```
#How was the data collected ?
## Person/Subject was asked to perform specific physical actions
## Signals produced due to that movement were recorded over time.
## 8 channels were used to record the signals
## Channels here correspond to muscles\ For eg: Right-hand bicep
## Frequency : 10 per ms
```

```
!wget "https://drive.google.com/uc?export=download&id=1ZgZYFdMlQyccRtHs16AhJUEnLPhgjCc7" -O EMG_data.rar
```

```
--2022-05-18 13:05:36--  https://drive.google.com/uc?export=download&id=1ZgZYFdMlQyccRtHs16AhJUEnLPhgjCc7
Resolving drive.google.com (drive.google.com)... 108.177.119.102, 108.177.119.139, 108.177.119.113, ...
Connecting to drive.google.com (drive.google.com)|108.177.119.102|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://doc-0o-14-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc717deffksulhg5h7mbp1/bkumsf471ms7nj
Warning: wildcards not supported in HTTP.
--2022-05-18 13:05:42--  https://doc-0o-14-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc717deffksulhg5h7mbp1
Resolving doc-0o-14-docs.googleusercontent.com (doc-0o-14-docs.googleusercontent.com)... 142.250.145.132, 2a00:1450:
Connecting to doc-0o-14-docs.googleusercontent.com (doc-0o-14-docs.googleusercontent.com)|142.250.145.132|:443... co
HTTP request sent, awaiting response... 200 OK
```

7 / 7

```
# EMG Data Analysis
# Source: https://archive.ics.uci.edu/ml/datasets/EMG+Physical+Action+Data+Set
```

4 people

```
#How was the data collected ?
## Person/Subject was asked to perform specific physical actions
## Signals produced due to that movement were recorded over time.
## 8 channels were used to record the signals
## Channels here correspond to muscles\ For eg: Right-hand bicep
## Frequency : 10 per ms
```

8 signals → activity

```
!wget "https://drive.google.com/uc?export=download&id=1ZgZYFdMlQyccRtHs16AhJUEnLPhgjCc7" -O EMG_data.rar
```

```
--2022-05-18 13:05:36--  https://drive.google.com/uc?export=download&id=1ZgZYFdMlQyccRtHs16AhJUEnLPhgjCc7
Resolving drive.google.com (drive.google.com)... 108.177.119.102, 108.177.119.139, 108.177.119.113, ...
Connecting to drive.google.com (drive.google.com)|108.177.119.102|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://doc-0o-14-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc717deffksulhg5h7mbp1/bkumsf471ms7nj
Warning: wildcards not supported in HTTP.
--2022-05-18 13:05:42--  https://doc-0o-14-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc717deffksulhg5h7mbp1
Resolving doc-0o-14-docs.googleusercontent.com (doc-0o-14-docs.googleusercontent.com)... 142.250.145.132, 2a00:1450:
Connecting to doc-0o-14-docs.googleusercontent.com (doc-0o-14-docs.googleusercontent.com)|142.250.145.132|:443... co
HTTP request sent, awaiting response... 200 OK
```

```
EMG_data.rar        100%[===================>]  17.74M  35.3MB/s    in 0.5s

2022-05-18 15:47:22 (35.3 MB/s) - 'EMG_data.rar' saved [18602479/18602479]
```

```
[ ]  !unrar x "./EMG_data.rar" "./"

     Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Hugging.log  OK
     Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Jumping.log  OK
     Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Running.log  OK
     Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Seating.log  OK
     Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Standing.log  OK
     Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Walking.log  OK
     Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Waving.log  OK
     Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Bowing.txt  OK
     Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Clapping.txt  OK
     Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Handshaking.txt  OK
     Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Hugging.txt  OK
     Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Jumping.txt  OK
     Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Running.txt  OK
     Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Seating.txt  OK
     Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Standing.txt  OK
     Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Walking.txt  OK
     Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Waving.txt  OK
     Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Elbowing.log  OK
     Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/FrontKicking.log  OK
     Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Hamering.log  OK
```

```
Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Seating.log  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Standing.log  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Walking.log  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/log/Waving.log  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Bowing.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Clapping.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Handshaking.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Hugging.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Jumping.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Running.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Seating.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Standing.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Walking.txt  OK
Extracting  ./EMG Physical Action Data Set/sub3/Normal/txt/Waving.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Elbowing.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/FrontKicking.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Hamering.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Headering.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Kneeing.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Pulling.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Punching.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Pushing.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/SideKicking.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/log/Slapping.log  OK
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/txt/Elbowing.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/txt/Frontkicking.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/txt/Hamering.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Aggressive/txt/Headering.txt  OK
```

```
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Seating.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Standing.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Walking.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Waving.txt  OK
All OK
```

```
!sudo apt install tree
```

problem

XGBoost

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
tree is already the newest version (1.7.0-5).
The following packages were automatically installed and are no longer required:
  libnvidia-common-460 nsight-compute-2020.2.0
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 67 not upgraded.
```

```
!tree "./EMG Physical Action Data Set/sub1"
```

```
./EMG Physical Action Data Set/sub1
├── Aggressive
│   ├── log
│   │   ├── Elbowing.log
│   │   ├── FrontKicking.log
```

```
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Seating.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Standing.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Walking.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Waving.txt  OK
All OK
```

```
!sudo apt install tree
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
tree is already the newest version (1.7.0-5).
The following packages were automatically installed and are no longer required:
  libnvidia-common-460 nsight-compute-2020.2.0
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 67 not upgraded.
```

```
!tree "./EMG Physical Action Data Set/sub1"
```

```
./EMG Physical Action Data Set/sub1
├── Aggressive
│   ├── log
│   │   ├── Elbowing.log
│   │   ├── FrontKicking.log
```

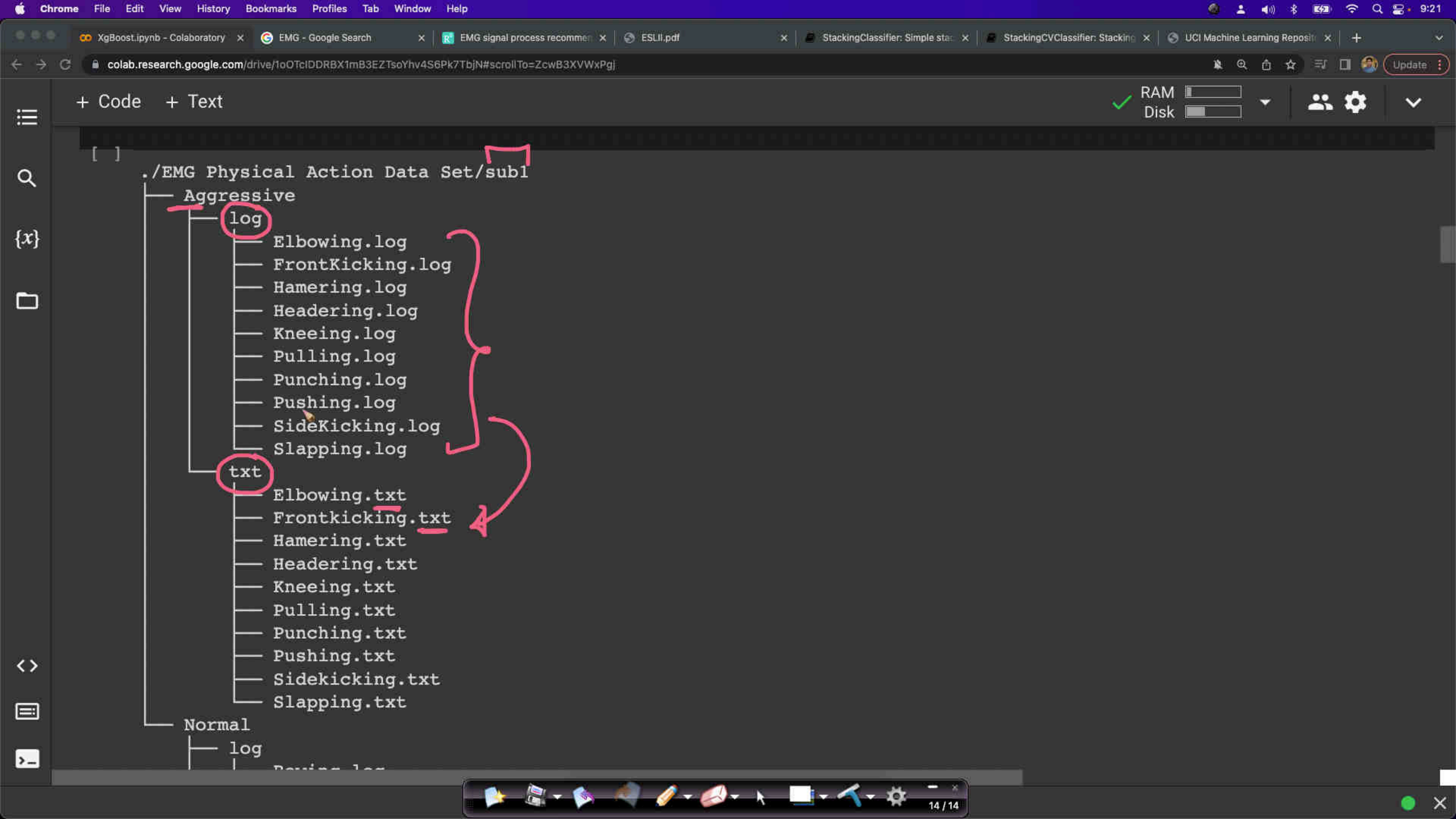
```
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Seating.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Standing.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Walking.txt  OK
Extracting  ./EMG Physical Action Data Set/sub4/Normal/txt/Waving.txt  OK
All OK
```

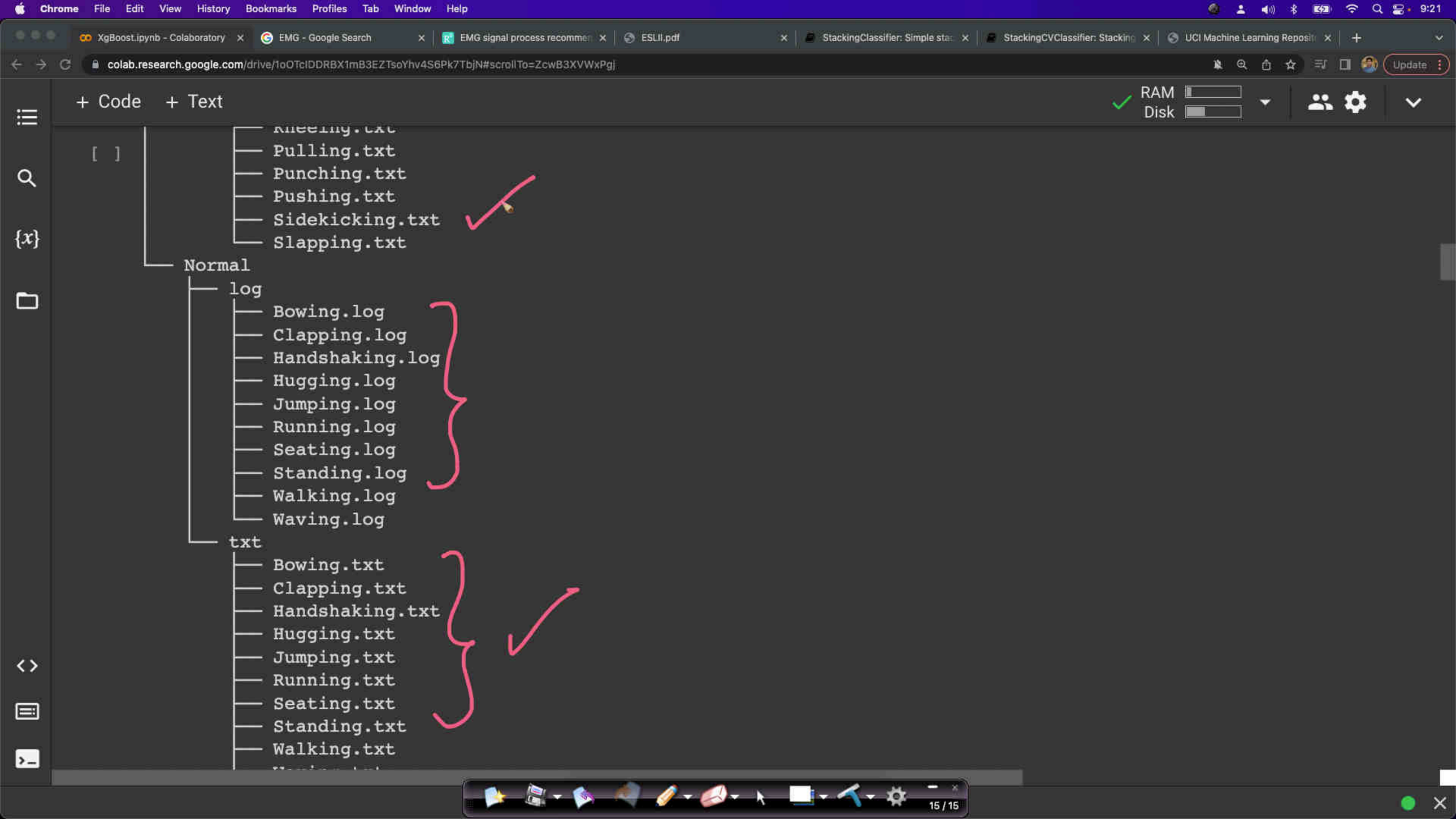
```
!sudo apt install tree
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
tree is already the newest version (1.7.0-5).
The following packages were automatically installed and are no longer required:
  libnvidia-common-460 nsight-compute-2020.2.0
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 67 not upgraded.
```

```
!tree "./EMG Physical Action Data Set/sub1"
```

```
./EMG Physical Action Data Set/sub1
├── Aggressive
│   ├── log
│   │   ├── Elbowing.log
│   │   ├── FrontKicking.log
```


./EMG Physical Action Data Set/sub1
Aggressive
log
Elbowing.log
FrontKicking.log
Hamering.log
Headering.log
Kneeing.log
Pulling.log
Punching.log
Pushing.log
SideKicking.log
Slapping.log
txt
Elbowing.txt
Frontkicking.txt
Hamering.txt
Headering.txt
Kneeing.txt
Pulling.txt
Punching.txt
Pushing.txt
Sidekicking.txt
Slapping.txt
Normal
log


Chrome File Edit View History Bookmarks Profiles Tab Window Help
XgBoost.ipynb - Colaboratory
EMG - Google Search
EMG signal process recommen
ESLII.pdf
StackingClassifier: Simple sta
StackingCVClassifier: Stacking
UCI Machine Learning Reposit
colab.research.google.com/drive/1oOTcIDDRBX1mB3EZTsoYhv4S6Pk7TbjN#scrollTo=ZcwB3XVWxPgj
Update
+ Code
+ Text
RAM
Disk
[ ]
Pulling.txt
Punching.txt
Pushing.txt
Sidekicking.txt
Slapping.txt
Normal
log
Bowing.log
Clapping.log
Handshaking.log
Hugging.log
Jumping.log
Running.log
Seating.log
Standing.log
Walking.log
Waving.log
txt
Bowing.txt
Clapping.txt
Handshaking.txt
Hugging.txt
Jumping.txt
Running.txt
Seating.txt
Standing.txt
Walking.txt
15 / 15

```
        ├── Running.txt
        ├── Seating.txt
        ├── Standing.txt
        ├── Walking.txt
        └── Waving.txt

    6 directories, 40 files
```

```
[ ] !ls -lrt ./EMG\ Physical\ Action\ Data\ Set/sub1/Aggressive/txt/
```

```
total 3768
-rw-r--r-- 1 root root 361096 Feb  7  2010 Slapping.txt
-rw-r--r-- 1 root root 388912 Feb  7  2010 Sidekicking.txt
-rw-r--r-- 1 root root 379428 Feb  7  2010 Pushing.txt
-rw-r--r-- 1 root root 379597 Feb  7  2010 Punching.txt
-rw-r--r-- 1 root root 387656 Feb  7  2010 Pulling.txt
-rw-r--r-- 1 root root 398523 Feb  7  2010 Kneeing.txt
-rw-r--r-- 1 root root 350285 Feb  7  2010 Headering.txt
-rw-r--r-- 1 root root 402363 Feb  7  2010 Hamering.txt
-rw-r--r-- 1 root root 390158 Feb  7  2010 Frontkicking.txt
-rw-r--r-- 1 root root 398095 Feb  7  2010 Elbowing.txt
```

```
[ ] !cat ./EMG\ Physical\ Action\ Data\ Set/sub1/Aggressive/txt/Slapping.txt
```

```
-999    392     38      -51     1409    -90     628     -348
-979    335     30      -95     1995    -71     653     -297
-858    213     -9      -98     2516    -153    419     -305
```

```
-rw-r--r-- 1 root root 379597 Feb  7  2010 Punching.txt
-rw-r--r-- 1 root root 387656 Feb  7  2010 Pulling.txt
-rw-r--r-- 1 root root 398523 Feb  7  2010 Kneeing.txt
-rw-r--r-- 1 root root 350285 Feb  7  2010 Headering.txt
-rw-r--r-- 1 root root 402363 Feb  7  2010 Hamering.txt
-rw-r--r-- 1 root root 390158 Feb  7  2010 Frontkicking.txt
-rw-r--r-- 1 root root 398095 Feb  7  2010 Elbowing.txt
```

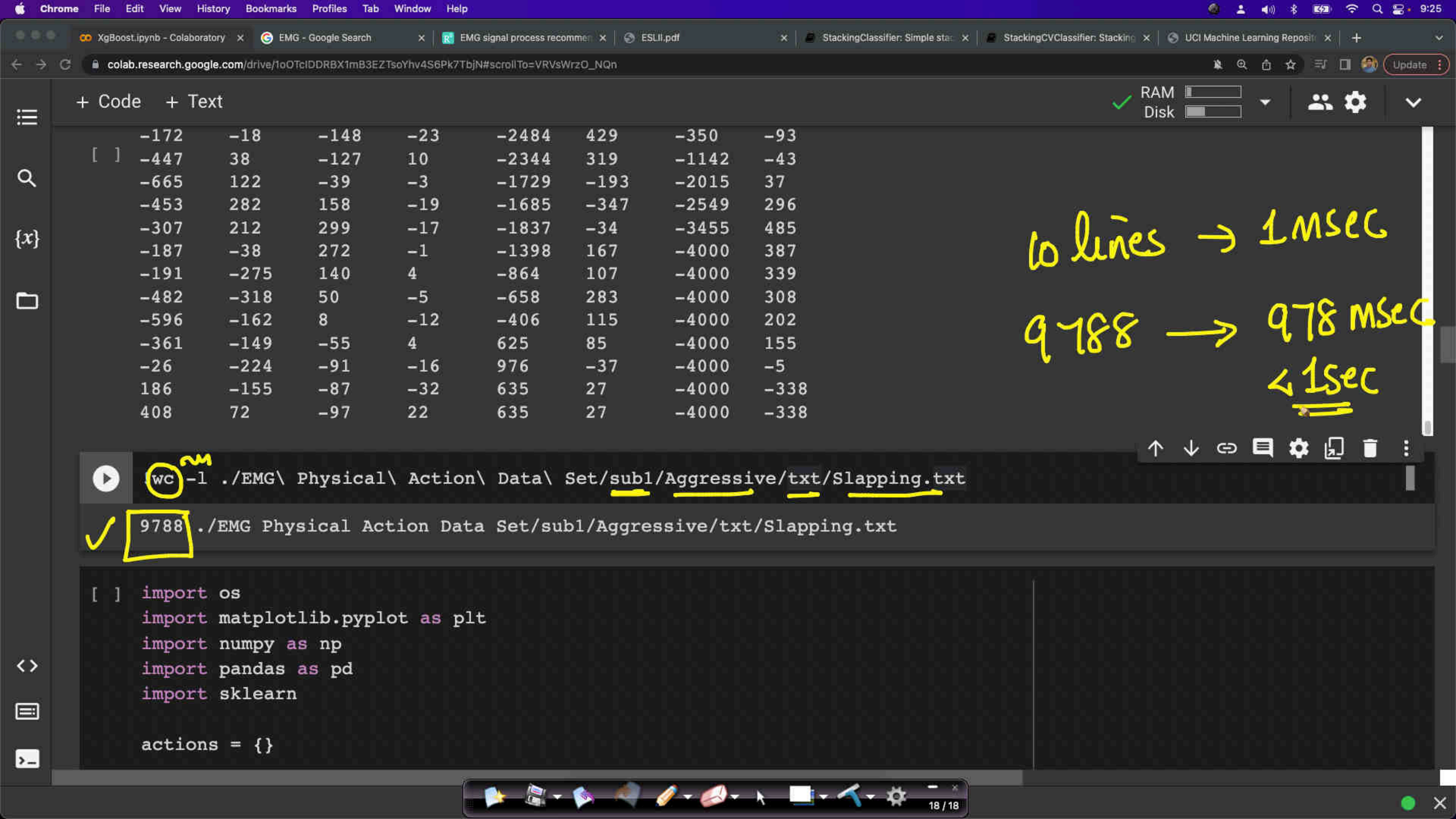
```
!cat ./EMG\ Physical\ Action\ Data\ Set/subj/Aggressive/txt/Slapping.txt
```

```
-999    392     38      -51     1409    -90     628     -348
-979    335     30      -95     1995    -71     653     -297
-858    213     -9      -98     2516    -153    419     -305
-649    93      -63     -62     2822    -11     384     -297
-417    -15     -52     -90     2968    -87     102     -294
-245    -91     -76     -54     2947    -216    -379    -319
-93     -192    -59     -48     2822    -251    -236    -288
-42     -215    -66     -12     2374    -137    329     -276
-58     -136    -37     -5      1771    -140    411     -278
-39     -161    62      -4      1409    -32     329     -314
6       -173    37      18      1017    180     702     -333
38      -167    7       1       519     426     1119    -380
73      -210    19      23      -147    766     1520    -314
12      -161    9       10      -1007   365     1591    -195
-104    43      2       23      -1980   -6      1451    -179
-84     87      21      2       -2395   101     1402    -154
41      -53     8       16      -1945   -113    1320    -105
```

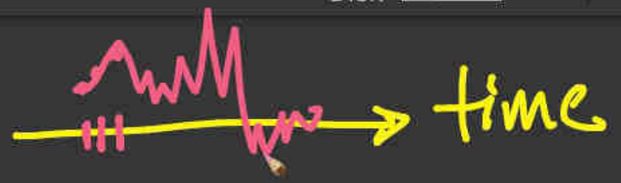
10 times perms

8 - channels/sensors

1ms


Chrome File Edit View History Bookmarks Profiles Tab Window Help
XgBoost.ipynb - Colaboratory
EMG - Google Search
EMG signal process recommen
ESLII.pdf
StackingClassifier: Simple sta
StackingCVClassifier: Stacking
UCI Machine Learning Reposit
colab.research.google.com/drive/1oOTcIDDRBX1mB3EZTsoYhv4S6Pk7TbjN#scrollTo=VRVsWrzO_NQn
Update
+ Code + Text
RAM
Disk
-172 -18 -148 -23 -2484 429 -350 -93
-447 38 -127 10 -2344 319 -1142 -43
-665 122 -39 -3 -1729 -193 -2015 37
-453 282 158 -19 -1685 -347 -2549 296
-307 212 299 -17 -1837 -34 -3455 485
-187 -38 272 -1 -1398 167 -4000 387
-191 -275 140 4 -864 107 -4000 339
-482 -318 50 -5 -658 283 -4000 308
-596 -162 8 -12 -406 115 -4000 202
-361 -149 -55 4 625 85 -4000 155
-26 -224 -91 -16 976 -37 -4000 -5
186 -155 -87 -32 635 27 -4000 -338
408 72 -97 22 635 27 -4000 -338
10 lines → 1 msec
9788 → 978 msec
< 1sec
wc -l ./EMG\ Physical\ Action\ Data\ Set/sub1/Aggressive/txt/Slapping.txt
9788 ./EMG Physical Action Data Set/sub1/Aggressive/txt/Slapping.txt
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn
actions = {}
18 / 18

```
-172    -18     -148    -23     -2484   429     -350    -93
-447    38      -127    10      -2344   319     -1142   -43
-665    122     -39     -3      -1729   -193    -2015   37
-453    282     158     -19     -1685   -347    -2549   296
-307    212     299     -17     -1837   -34     -3455   485
-187    -38     272     -1      -1398   167     -4000   387
-191    -275    140     4       -864    107     -4000   339
-482    -318    50      -5      -658    283     -4000   308
-596    -162    8       -12     -406    115     -4000   202
-361    -149    -55     4       625     85      -4000   155
-26     -224    -91     -16     976     -37     -4000   -5
186     -155    -87     -32     635     27      -4000   -338
408     72      -97     22      635     27      -4000   -338
```




```
!wc -l ./EMG\ Physical\ Action\ Data\ Set/sub1/Aggressive/txt/Slapping.txt
```

```
9788 ./EMG Physical Action Data Set/sub1/Aggressive/txt/Slapping.txt
```

```
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn

actions = {}
```

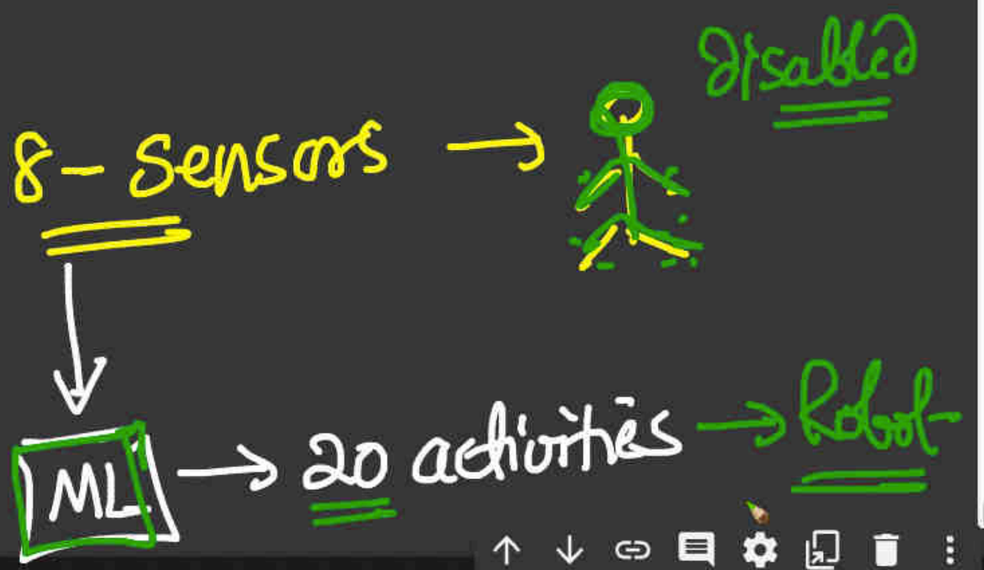
```
-172    -18     -148    -23     -2484   429     -350    -93
-447    38      -127    10      -2344   319     -1142   -43
-665    122     -39     -3      -1729   -193    -2015   37
-453    282     158     -19     -1685   -347    -2549   296
-307    212     299     -17     -1837   -34     -3455   485
-187    -38     272     -1      -1398   167     -4000   387
-191    -275    140     4       -864    107     -4000   339
-482    -318    50      -5      -658    283     -4000   308
-596    -162    8       -12     -406    115     -4000   202
-361    -149    -55     4       625     85      -4000   155
-26     -224    -91     -16     976     -37     -4000   -5
186     -155    -87     -32     635     27      -4000   -338
408     72      -97     22      635     27      -4000   -338
```

```
!wc -l ./EMG\ Physical\ Action\ Data\ Set/sub1/Aggressive/txt/Slapping.txt
```

```
9788 ./EMG Physical Action Data Set/sub1/Aggressive/txt/Slapping.txt
```

```
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn

actions = {}
```

```
-172   -18   -148   -23   -2484   429   -350   -93
-447    38   -127    10   -2344   319  -1142   -43
-665   122    -39    -3   -1729  -193  -2015    37
-453   282    158   -19   -1685  -347  -2549   296
-307   212    299   -17   -1837   -34  -3455   485
-187   -38    272    -1   -1398   167  -4000   387
-191  -275    140     4    -864   107  -4000   339
-482  -318     50    -5    -658   283  -4000   308
-596  -162      8   -12    -406   115  -4000   202
-361  -149    -55     4     625    85  -4000   155
 -26  -224    -91   -16     976   -37  -4000    -5
 186  -155    -87   -32     635    27  -4000  -338
 408    72    -97    22     635    27  -4000  -338
```




```
!wc -l ./EMG\ Physical\ Action\ Data\ Set/sub1/Aggressive/txt/Slapping.txt
```

```
9788 ./EMG Physical Action Data Set/sub1/Aggressive/txt/Slapping.txt
```

```
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn

actions = {}
```

+ Code + Text

RAM
Disk

```
-172    -18    -148    -23    -2484    429    -350    -93
-447    38     -127    10     -2344    319    -1142   -43
-665    122    -39     -3     -1729    -193   -2015   37
-453    282    158     -19    -1685    -347   -2549   296
-307    212    299     -17    -1837    -34    -3455   485
-187    -38    272     -1     -1398    167    -4000   387
-191    -275   140     4      -864     107    -4000   339
-482    -318   50      -5     -658     283    -4000   308
-596    -162   8       -12    -406     115    -4000   202
-361    -149   -55     4      625      85     -4000   155
-26     -224   -91     -16    976      -37    -4000   -5
186     -155   -87     -32    635      27     -4000   -338
408     72     -97     22     635      27     -4000   -338
```

raw & signals → activity

```
!wc -l ./EMG\ Physical\ Action\ Data\ Set/sub1/Aggressive/txt/Slapping.txt
```

```
9788 ./EMG Physical Action Data Set/sub1/Aggressive/txt/Slapping.txt
```

```
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn

actions = {}
```

22 / 22

-172	-18	-148	-23	-2484	429	-350	-93
-447	38	-127	10	-2344	319	-1142	-43
-665	122	-39	-3	-1729	-193	-2015	37
-453	282	158	-19	-1685	-347	-2549	296
-307	212	299	-17	-1837	-34	-3455	485
-187	-38	272	-1	-1398	167	-4000	387
-191	-275	140	4	-864	107	-4000	339
-482	-318	50	-5	-658	283	-4000	308
-596	-162	8	-12	-406	115	-4000	202
-361	-149	-55	4	625	85	-4000	155
-26	-224	-91	-16	976	-37	-4000	-5
186	-155	-87	-32	635	27	-4000	-338
408	72	-97	22	635	27	-4000	-338

```
!wc -l ./EMG\ Physical\ Action\ Data\ Set/sub1/Aggressive/txt/Slapping.txt
```

```
9788 ./EMG Physical Action Data Set/sub1/Aggressive/txt/Slapping.txt
```

```
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn

actions = {}
```

Segment	R-Arm	L-Arm	R-Leg	L-Leg
Channel	ch1	ch2	ch3	ch4	ch5	ch6	ch7	ch8
Muscle	R-Bic	R-Tri	L-Bic	L-Tri	R-Thi	R-Ham	L-Thi	L-Ham
Column	0	1	2	3	4	5	6	7

Segment: A segment defines a body segment or limb.
- Right arm (R-Arm)
- Left arm (L-Arm)
- Right leg (R-Leg)
- Left leg (L-Leg)

Channel: A channel corresponds to an electrode attached on a muscle.

Muscle: A pair of muscles that corresponds to a segment.
- R-Bic: right bicep (C1)
- R-Tri: right tricep (C2)
- L-Bic: left bicep (C3)
- L-Tri: left tricep (C4)
- R-Thi: right thigh (C5)
- R-Ham: right hamstring (C6)
- L-Thi: left thigh (C7)
- L-Ham: left hamstring (C8)

Relevant Papers:

N/A

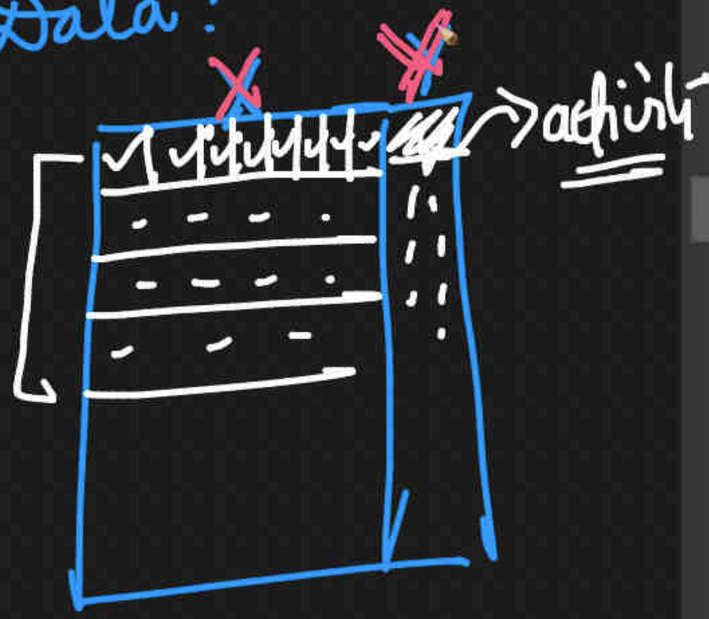
```
[ ]  import os
     import matplotlib.pyplot as plt
     import numpy as np
     import pandas as pd
     import sklearn

     actions = {}

     data_dirs = ["./EMG Physical Action Data Set/sub1/Aggressive/txt",
                  "./EMG Physical Action Data Set/sub1/Normal/txt"]

     ind = 0
     data = pd.DataFrame()

     for dirs in data_dirs :

       for files in os.listdir(dirs):

         with open(os.path.join(dirs, files), "r") as f:

           temp = pd.read_csv(f.name,
                              sep = "\t",
                              header = None,
                              names = ["ch" + str(i) for i in range(1, 9)] # 8 input channels
                              )
```

8-chamels

1 → 1/10 ms

1sec → 10k obs

Data :



```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn

actions = {}

data_dirs = ["./EMG Physical Action Data Set/sub1/Aggressive/txt",
             "./EMG Physical Action Data Set/sub1/Normal/txt"]

ind = 0
data = pd.DataFrame()

for dirs in data_dirs :

  for files in os.listdir(dirs):

    with open(os.path.join(dirs, files), "r") as f:

      temp = pd.read_csv(f.name,
                         sep = "\t",
                         header = None,
                         names = ["ch" + str(i) for i in range(1, 9)] # 8 input channels
                         )
```

```
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn

actions = {}

data_dirs = ["./EMG Physical Action Data Set/sub1/Aggressive/txt",
             "./EMG Physical Action Data Set/sub1/Normal/txt"]

ind = 0
data = pd.DataFrame()

for dirs in data_dirs :

  for files in os.listdir(dirs):

    with open(os.path.join(dirs, files), "r") as f:

      temp = pd.read_csv(f.name,
                         sep = "\t",
                         header = None,
                         names = ["ch" + str(i) for i in range(1, 9)] # 8 input channels
                         )
```

```
[ ]  data_dirs = ["./EMG Physical Action Data Set/sub1/Aggressive/txt",
                  "./EMG Physical Action Data Set/sub1/Normal/txt"]

     ind = 0
     data = pd.DataFrame()

     for dirs in data_dirs :

       for files in os.listdir(dirs):

         with open(os.path.join(dirs, files), "r") as f:

           temp = pd.read_csv(f.name,
                              sep = "\t",
                              header = None,
                              names = ["ch" + str(i) for i in range(1, 9)] # 8 input channels
                              )

           # chunking using Max of every 10 sequential values.
           temp_chunked = pd.DataFrame()

           for i in range(0, len(temp), 10):
             temp_chunked = temp_chunked.append(temp.iloc[i:i+10].max(), ignore_index = True)

           labels = [files[:-4] for i in range(len(temp_chunked))] # remove the last 4 characters=".txt" from the filena
```

```
[ ]  for dirs in data_dirs :

       for files in os.listdir(dirs):

         with open(os.path.join(dirs, files), "r") as f:

           temp = pd.read_csv(f.name,
                              sep = "\t",
                              header = None,
                              names = ["ch" + str(i) for i in range(1, 9)] # 8 input channels
                              )

           # chunking using Max of every 10 sequential values.
           temp_chunked = pd.DataFrame()

           for i in range(0, len(temp), 10):
             temp_chunked = temp_chunked.append(temp.iloc[i:i+10].max(), ignore_index = True)

           labels = [files[:-4] for i in range(len(temp_chunked))] # remove the last 4 characters=".txt" from the filena
           actions[files[:-4]] = ind

           temp_chunked["Action"] = labels

           data = pd.concat([data, temp_chunked])
```

TSV

```
[ ]
    for dirs in data_dirs :

      for files in os.listdir(dirs):

        with open(os.path.join(dirs, files), "r") as f:

          temp = pd.read_csv(f.name,
                             sep = "\t",
                             header = None,
                             names = ["ch" + str(i) for i in range(1, 9)] # 8 input channels
                             )

          # chunking using Max of every 10 sequential values.
          temp_chunked = pd.DataFrame()

          for i in range(0, len(temp), 10):
            temp_chunked = temp_chunked.append(temp.iloc[i:i+10].max(), ignore_index = True)

          labels = [files[:-4] for i in range(len(temp_chunked))] # remove the last 4 characters=".txt" from the filena
          actions[files[:-4]] = ind

          temp_chunked["Action"] = labels

          data = pd.concat([data, temp_chunked])
```

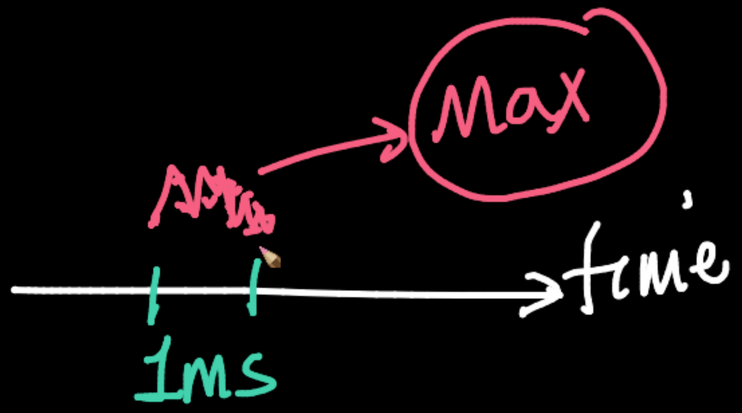
ch1 ch2 - - - ch8

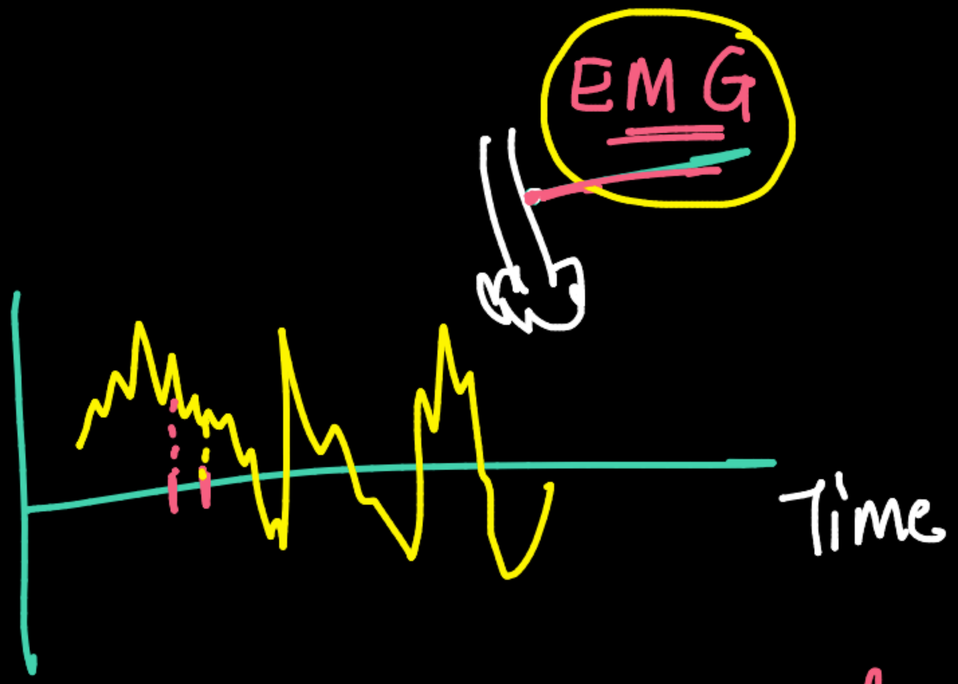
{

Signal-processing

Biomedical-Signal-

Spikes → Mallet





$\frac{1}{10}$th of ms → 10ks

1sec → 10000ks

```
[ ]
    for dirs in data_dirs :

      for files in os.listdir(dirs):

        with open(os.path.join(dirs, files), "r") as f:

          temp = pd.read_csv(f.name,
                             sep = "\t",
                             header = None,
                             names = ["ch" + str(i) for i in range(1, 9)] # 8 input channels
                             )

          # chunking using Max of every 10 sequential values.
          temp_chunked = pd.DataFrame()

          for i in range(0, len(temp), 10):
            temp_chunked = temp_chunked.append(temp.iloc[i:i+10].max(), ignore_index = True)

          labels = [files[:-4] for i in range(len(temp_chunked))] # remove the last 4 characters=".txt" from the filena
          actions[files[:-4]] = ind

          temp_chunked["Action"] = labels

          data = pd.concat([data, temp_chunked])
```

max (----)10-obs values

```
[ ]                     names = ["ch" + str(i) for i in range(1, 9)] # 8 input channels
                        )

        # chunking using Max of every 10 sequential values.
        temp_chunked = pd.DataFrame()

        for i in range(0, len(temp), 10):
          temp_chunked = temp_chunked.append(temp.iloc[i:i+10].max(), ignore_index = True)

        labels = [files[:-4] for i in range(len(temp_chunked))] # remove the last 4 characters=".txt" from the filena
        actions[files[:-4]] = ind

        temp_chunked["Action"] = labels

        data = pd.concat([data, temp_chunked])

        ind+=1

    print(actions)
```

```
{'Frontkicking': 0, 'Slapping': 1, 'Elbowing': 2, 'Pulling': 3, 'Kneeing': 4, 'Hamering': 5, 'Sidekicking': 6, 'Push
```

```
[ ] data.head()
```

```
[ ]                         names = ["ch" + str(i) for i in range(1, 9)] # 8 input channels
                            )

        # chunking using Max of every 10 sequential values.
        temp_chunked = pd.DataFrame()

        for i in range(0, len(temp), 10):
          temp_chunked = temp_chunked.append(temp.iloc[i:i+10].max(), ignore_index = True)

        labels = [files[:-4] for i in range(len(temp_chunked))] # remove the last 4 characters=".txt" from the filena
        actions[files[:-4]] = ind

        temp_chunked["Action"] = labels

        data = pd.concat([data, temp_chunked])

        ind+=1

    print(actions)
```

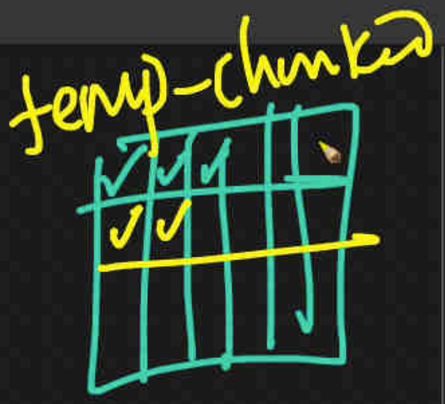
```
{'Frontkicking': 0, 'Slapping': 1, 'Elbowing': 2, 'Pulling': 3, 'Kneeing': 4, 'Hamering': 5, 'Sidekicking': 6, 'Push
```

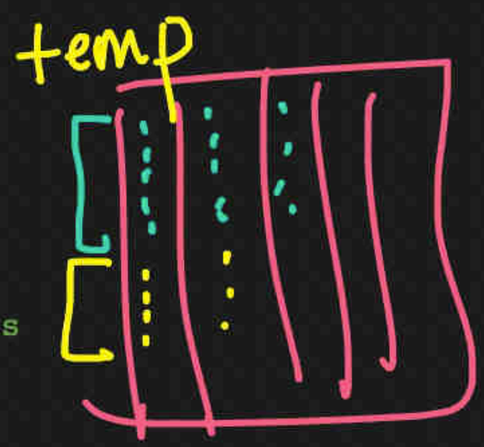
```
[ ] data.head()
```

```
                    sep = "\t",
                    header = None,
                    names = ["ch" + str(i) for i in range(1, 9)] # 8 input channels
                    )

    # chunking using Max of every 10 sequential values.
    temp_chunked = pd.DataFrame()

    for i in range(0, len(temp), 10):
      temp_chunked = temp_chunked.append(temp.iloc[i:i+10].max(), ignore_index = True)

    labels = [files[:-4] for i in range(len(temp_chunked))] # remove the last 4 characters=".txt" from the filena
    actions[files[:-4]] = ind

    temp_chunked["Action"] = labels

    data = pd.concat([data, temp_chunked])

    ind+=1

print(actions)
```

```
{'Frontkicking': 0, 'Slapping': 1, 'Elbowing': 2, 'Pulling': 3, 'Kneeing': 4, 'Hamering': 5, 'Sidekicking': 6, 'Push
```

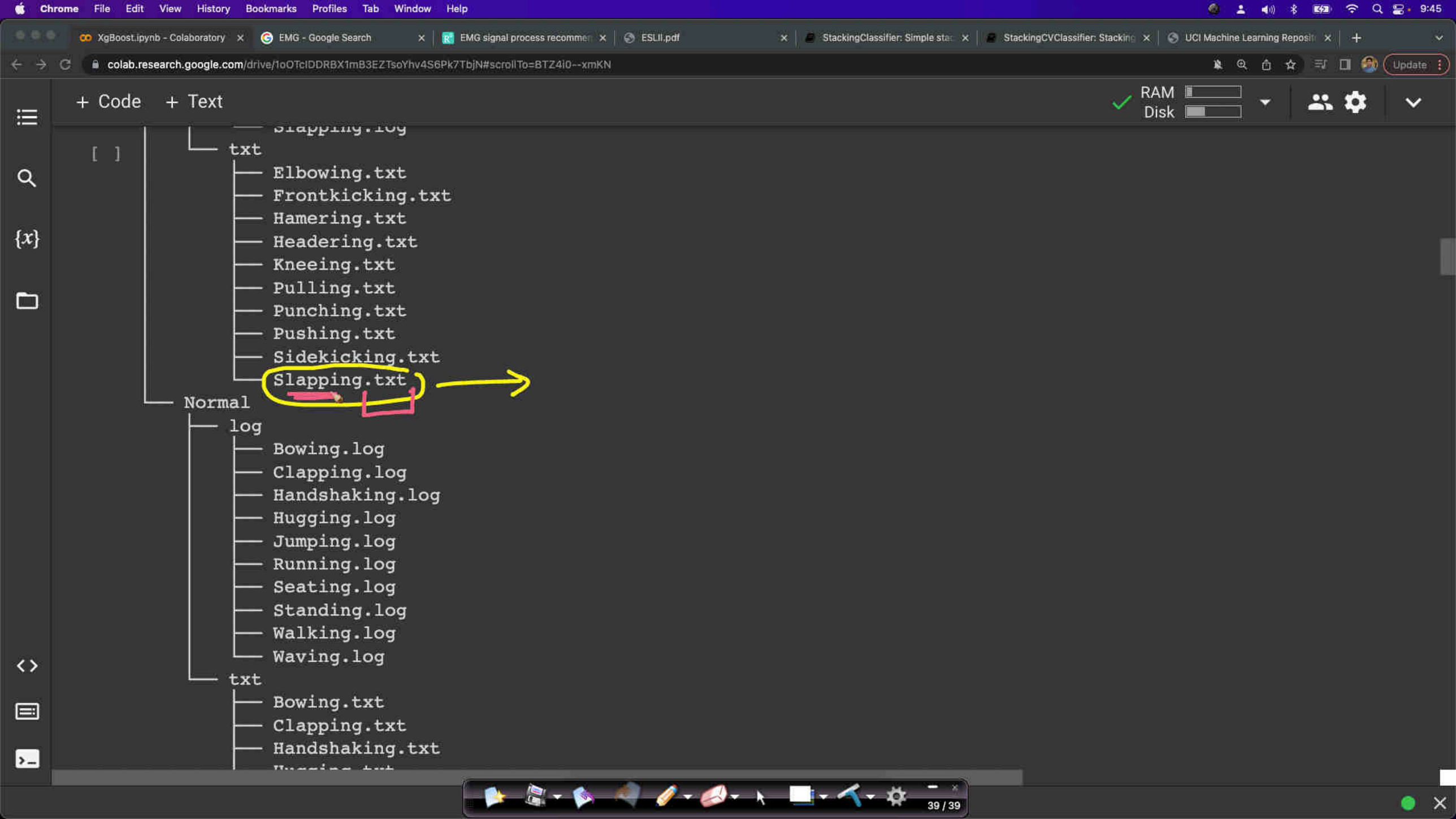
```
for files in os.listdir(dirs):

  with open(os.path.join(dirs, files), "r") as f:

    temp = pd.read_csv(f.name,
                       sep = "\t",
                       header = None,
                       names = ["ch" + str(i) for i in range(1, 9)] # 8 input channels
                       )

    # chunking using Max of every 10 sequential values.
    temp_chunked = pd.DataFrame()

    for i in range(0, len(temp), 10):
        temp_chunked = temp_chunked.append(temp.iloc[i:i+10].max(), ignore_index = True)

    labels = [files[:-4] for i in range(len(temp_chunked))] # remove the last 4 characters=".txt" from the filena
    actions[files[:-4]] = ind

    temp_chunked["Action"] = labels

    data = pd.concat([data, temp_chunked])

    ind+=1
```





```
        temp = pd.read_csv(f.name,
                           sep = "\t",
                           header = None,
                           names = ["ch" + str(i) for i in range(1, 9)] # 8 input channels
                           )

        # chunking using Max of every 10 sequential values.
        temp_chunked = pd.DataFrame()

        for i in range(0, len(temp), 10):
          temp_chunked = temp_chunked.append(temp.iloc[i:i+10].max(), ignore_index = True)

        labels = [files[:-4] for i in range(len(temp_chunked))] # remove the last 4 characters=".txt" from the filena
        actions[files[:-4]] = ind

        temp_chunked["Action"] = labels

        data = pd.concat([data, temp_chunked])

        ind+=1

print(actions)
```

```
{'Frontkicking': 0, 'Slapping': 1, 'Elbowing': 2, 'Pulling': 3, 'Kneeing': 4, 'Hammering': 5, 'Sidekicking': 6, 'Punch
```

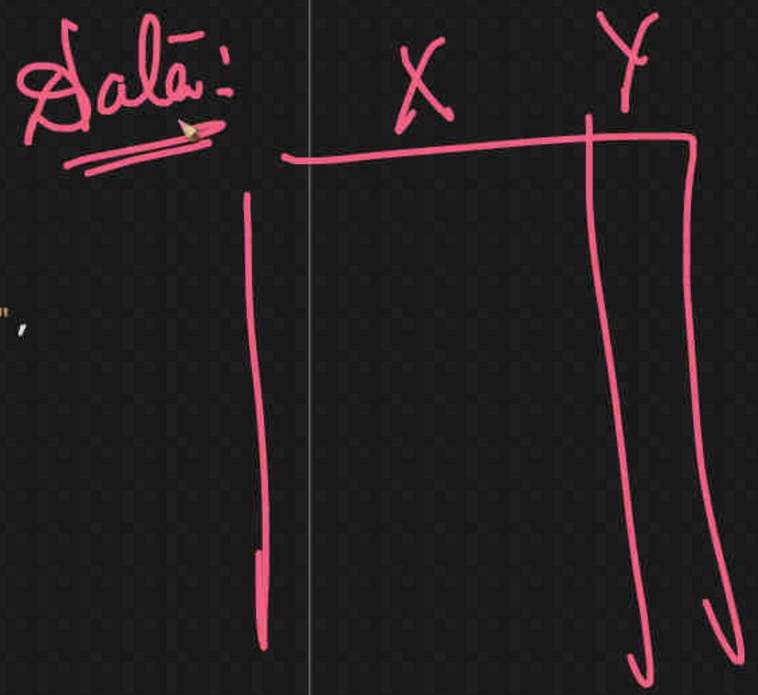

Chrome File Edit View History Bookmarks Profiles Tab Window Help
XgBoost.ipynb - Colaboratory
EMG - Google Search
EMG signal process recommen
ESLII.pdf
StackingClassifier: Simple sta
StackingCVClassifier: Stacking
UCI Machine Learning Reposit
colab.research.google.com/drive/1oOTcIDDRBX1mB3EZTsoYhv4S6Pk7TbjN#scrollTo=BTZ4i0--xmKN
Update
+ Code
+ Text
RAM
Disk
[ ]
txt
Elbowing.txt
Frontkicking.txt
Hamering.txt
Headering.txt
Kneeing.txt
Pulling.txt
Punching.txt
Pushing.txt
Sidekicking.txt
Slapping.txt
Normal
log
Bowing.log
Clapping.log
Handshaking.log
Hugging.log
Jumping.log
Running.log
Seating.log
Standing.log
Walking.log
Waving.log
txt
Bowing.txt
Clapping.txt
Handshaking.txt
39 / 39

```
                    header = None,
                    names = ["ch" + str(i) for i in range(1, 9)] # 8 input channels
                    )

    # chunking using Max of every 10 sequential values.
    temp_chunked = pd.DataFrame()

    for i in range(0, len(temp), 10):
      temp_chunked = temp_chunked.append(temp.iloc[i:i+10].max(), ignore_index = True)

    labels = [files[:-4] for i in range(len(temp_chunked))] # remove the last 4 characters=".txt" from the filena
    actions[files[:-4]] = ind

    temp_chunked["Action"] = labels

    data = pd.concat([data, temp_chunked])

    ind+=1

print(actions)
```

```
{'Frontkicking': 0, 'Slapping': 1, 'Elbowing': 2, 'Pulling': 3, 'Kneeing': 4, 'Hamering': 5, 'Sidekicking': 6, 'Push
```

```
data.head()
```

```
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn

actions = {}

data_dirs = ["./EMG Physical Action Data Set/sub1/Aggressive/txt",
             "./EMG Physical Action Data Set/sub1/Normal/txt"]

ind = 0
data = pd.DataFrame()

for dirs in data_dirs :

  for files in os.listdir(dirs):

    with open(os.path.join(dirs, files), "r") as f:

      temp = pd.read_csv(f.name,
                         sep = "\t",
                         header = None,
                         names = ["ch" + str(i) for i in range(1, 9)] # 8 input channels
```



```
actions = {}

data_dirs = ["./EMG Physical Action Data Set/sub1/Aggressive/txt",
             "./EMG Physical Action Data Set/sub1/Normal/txt"]

ind = 0
data = pd.DataFrame()

for dirs in data_dirs :

  for files in os.listdir(dirs):

    with open(os.path.join(dirs, files), "r") as f:

      temp = pd.read_csv(f.name,
                         sep = "\t",
                         header = None,
                         names = ["ch" + str(i) for i in range(1, 9)] # 8 input channels
                         )

      # chunking using Max of every 10 sequential values.
      temp_chunked = pd.DataFrame()

      for i in range(0, len(temp), 10):
        temp_chunked = temp_chunked.append(temp.iloc[i:i+10].max(), ignore_index = True)
```

```
    for i in range(0, len(temp), 10):
      temp_chunked = temp_chunked.append(temp.iloc[i:i+10].max(), ignore_index = True)

    labels = [files[:-4] for i in range(len(temp_chunked))] # remove the last 4 characters=".txt" from the filena
    actions[files[:-4]] = ind

    temp_chunked["Action"] = labels

    data = pd.concat([data, temp_chunked])

    ind+=1

print(actions)
```

```
{'Frontkicking': 0, 'Slapping': 1, 'Elbowing': 2, 'Pulling': 3, 'Kneeing': 4, 'Hamering': 5, 'Sidekicking': 6, 'Push
```

```
[ ]  data.head()
```

	ch1	ch2	ch3	ch4	ch5	ch6	ch7	ch8	Action
0	328.0	433.0	533.0	210.0	3918.0	961.0	-3193.0	4000.0	Frontkicking
1	577.0	309.0	1550.0	-21.0	-4000.0	707.0	2297.0	3719.0	Frontkicking

```
    for i in range(0, len(temp), 10):
      temp_chunked = temp_chunked.append(temp.iloc[i:i+10].max(), ignore_index = True)

    labels = [files[:-4] for i in range(len(temp_chunked))] # remove the last 4 characters=".txt" from the filena
    actions[files[:-4]] = ind

    temp_chunked["Action"] = labels

    data = pd.concat([data, temp_chunked])

    ind+=1

print(actions)
```

```
{'Frontkicking': 0, 'Slapping': 1, 'Elbowing': 2, 'Pulling': 3, 'Kneeing': 4, 'Hamering': 5, 'Sidekicking': 6, 'Push
```

```
[ ] data.head()
```

	ch1	ch2	ch3	ch4	ch5	ch6	ch7	ch8	Action
0	328.0	433.0	533.0	210.0	3918.0	961.0	-3193.0	4000.0	Frontkicking
1	577.0	309.0	1550.0	-21.0	-4000.0	707.0	2297.0	3719.0	Frontkicking

```
{'Frontkicking': 0, 'Slapping': 1, 'Elbowing': 2, 'Pulling': 3, 'Kneeing': 4, 'Hamering':
```

```
[ ] data.head()
```

	ch1	ch2	ch3	ch4	ch5	ch6	ch7	ch8	Action
0	328.0	433.0	533.0	210.0	3918.0	961.0	-3193.0	4000.0	Frontkicking
1	577.0	309.0	1550.0	-21.0	-4000.0	707.0	2297.0	3719.0	Frontkicking
2	712.0	160.0	632.0	149.0	-3231.0	1515.0	3294.0	4000.0	Frontkicking
3	234.0	68.0	125.0	402.0	2972.0	2428.0	3009.0	1895.0	Frontkicking
4	2471.0	118.0	-263.0	-94.0	-4000.0	4000.0	-3592.0	613.0	Frontkicking

1 MS →

```
[ ] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19711 entries, 0 to 999
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ch1     19711 non-null  float64
 1   ch2     19711 non-null  float64
```

```
{'Frontkicking': 0, 'Slapping': 1, 'Elbowing': 2, 'Pulling': 3, 'Kneeing': 4, 'Hamering':
```

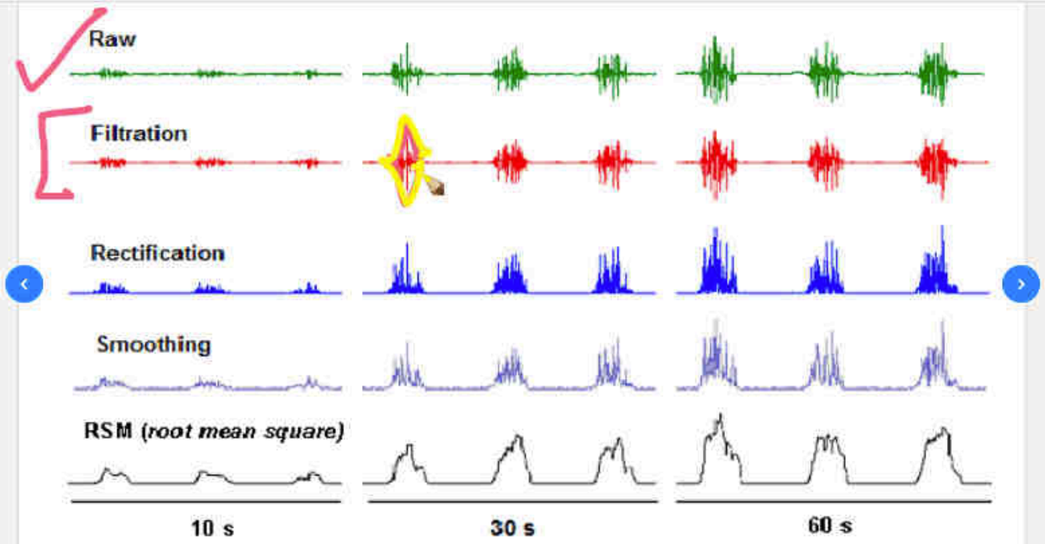
```
[ ] data.head()
```

	ch1	ch2	ch3	ch4	ch5	ch6	ch7	ch8	Action
0	328.0	433.0	533.0	210.0	3918.0	961.0	-3193.0	4000.0	Frontkicking
1	577.0	309.0	1550.0	-21.0	-4000.0	707.0	2297.0	3719.0	Frontkicking
2	712.0	160.0	632.0	149.0	-3231.0	1515.0	3294.0	4000.0	Frontkicking
3	234.0	68.0	125.0	402.0	2972.0	2428.0	3009.0	1895.0	Frontkicking
4	2471.0	118.0	-263.0	-94.0	-4000.0	4000.0	-3592.0	613.0	Frontkicking

averaging

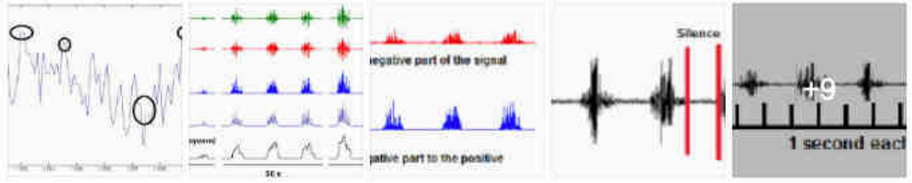
```
[ ] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19711 entries, 0 to 999
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ch1     19711 non-null  float64
 1   ch2     19711 non-null  float64
```



EMG signal process recommended. Green: The raw signal, no treatment was applied until this moment; Red: Filtrated signal, a limit was created for the signal, excluding everything out of it; Blue: Rectified signal, all negative values were transformed in positive ones and added; Purple: the smoothed signal, a linear enveloped was created and the extreme parts of the signal was excluded; Black: The RMS values after all the treatments.

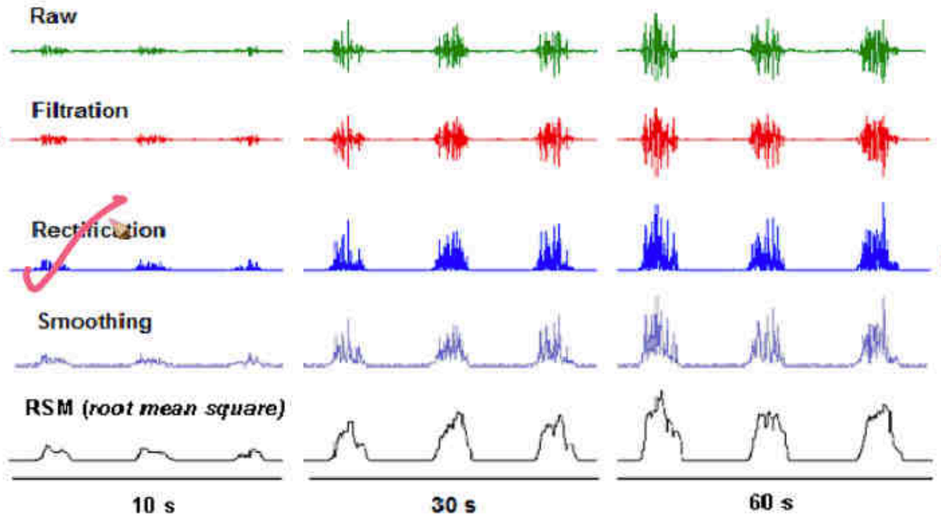
Source publication



Influence of Different Strategies of Treatment Muscle Contraction and Relaxation Phases on EMG Signal Processing and Analysis During Cyclic Exercise

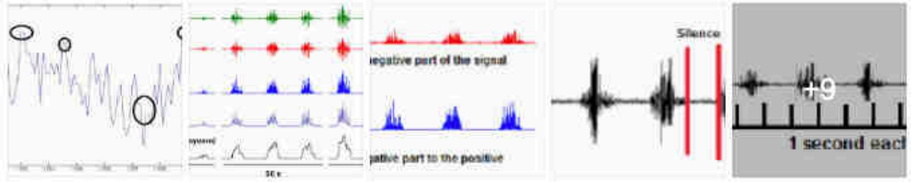
Chapter | Full-text available | Oct 2012

Leandro Altimari · J...



EMG signal process recommended. Green: The raw signal, no treatment was applied until this moment; Red: Filtrated signal, a limit was created for the signal, excluding everything out of it; Blue: Rectified signal, all negative values were transformed in positive ones and added; Purple: the smoothed signal, a linear enveloped was created and the extreme parts of the signal was excluded; Black: The RMS values after all the treatments.

Source publication



Influence of Different Strategies of Treatment Muscle Contraction and Relaxation Phases on EMG Signal Processing and Analysis During Cyclic Exercise

Chapter | Full-text available | Oct 2012

Leandro Altimari ·

4	2471.0	118.0	-263.0	-94.0	-4000.0	4000.0	-3592.0	613.0	Frontkicking

all 4 patients

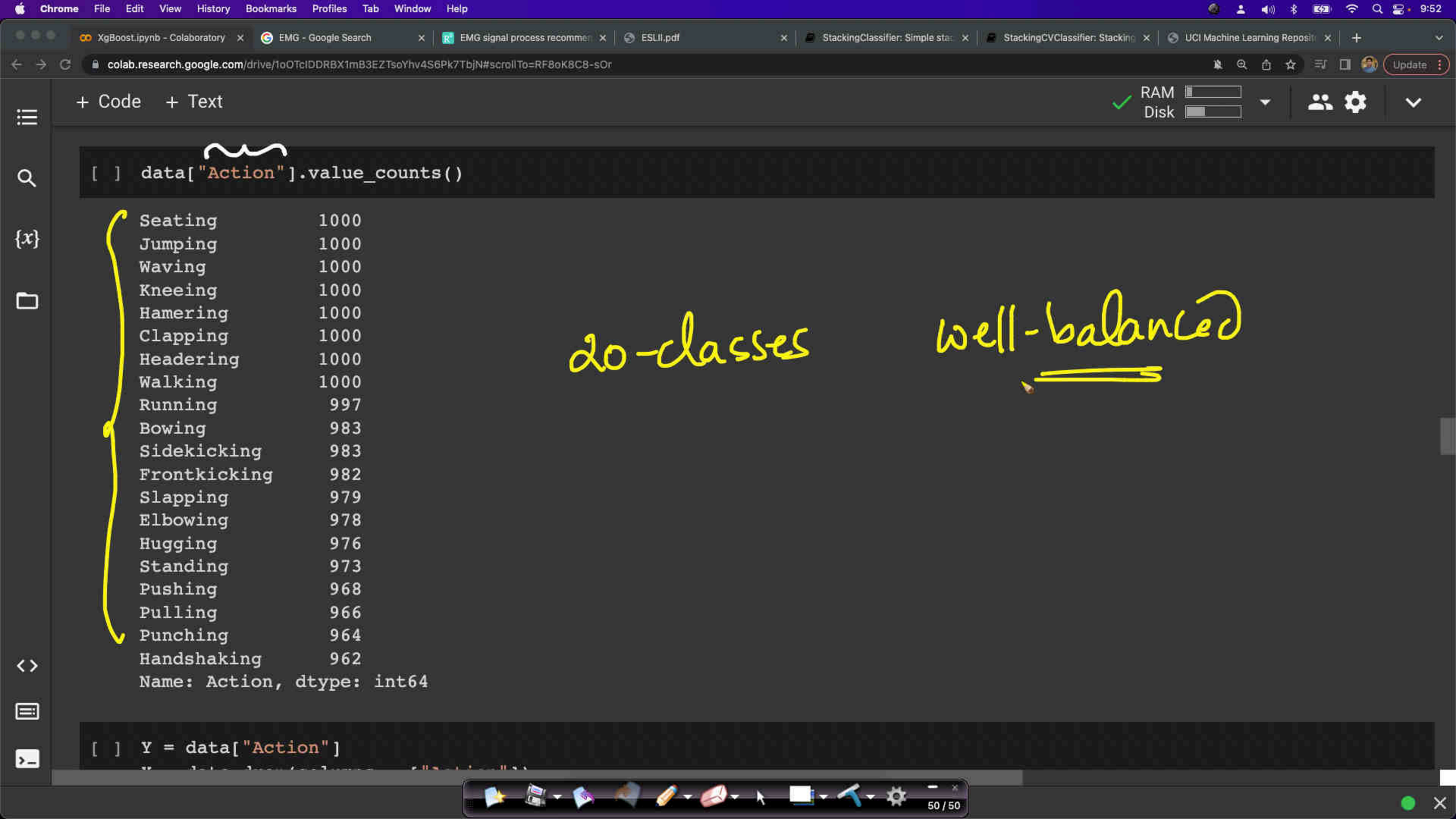
[ ] data.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19711 entries, 0 to 999
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   ch1     19711 non-null  float64
 1   ch2     19711 non-null  float64
 2   ch3     19711 non-null  float64
 3   ch4     19711 non-null  float64
 4   ch5     19711 non-null  float64
 5   ch6     19711 non-null  float64
 6   ch7     19711 non-null  float64
 7   ch8     19711 non-null  float64
 8   Action  19711 non-null  object 
dtypes: float64(8), object(1)
memory usage: 1.5+ MB
```

~20K rows

[ ] data["Action"].value_counts()

```
Seating    1000
Jumping    1000
Waving     1000
```


Chrome File Edit View History Bookmarks Profiles Tab Window Help
XgBoost.ipynb - Colaboratory
EMG - Google Search
EMG signal process recommen
ESLII.pdf
StackingClassifier: Simple sta
StackingCVClassifier: Stacking
UCI Machine Learning Reposit
colab.research.google.com/drive/1oOTcIDDRBX1mB3EZTsoYhv4S6Pk7TbjN#scrollTo=RF8oK8C8-sOr
Update
+ Code + Text
RAM
Disk
[ ] data["Action"].value_counts()
Seating 1000
Jumping 1000
Waving 1000
Kneeing 1000
Hamering 1000
Clapping 1000
Headering 1000
Walking 1000
Running 997
Bowing 983
Sidekicking 983
Frontkicking 982
Slapping 979
Elbowing 978
Hugging 976
Standing 973
Pushing 968
Pulling 966
Punching 964
Handshaking 962
Name: Action, dtype: int64
20-classes
well-balanced
[ ] Y = data["Action"]
50 / 50

```
Handshaking      962
Name: Action, dtype: int64
```

```
Y = data["Action"]
X = data.drop(columns = ["Action"])
```

```
# Label encoding
Y = Y.map(actions)
Y.head()
print(Y.value_counts())
```

```
19    1000
11    1000
18    1000
4     1000
5     1000
14    1000
9     1000
12    1000
13     997
15     983
6      983
0      982
1      979
2      978
```

```
Handshaking      962
Name: Action, dtype: int64
```

```
Y = data["Action"]
X = data.drop(columns = ["Action"])
```

```
# Label encoding
Y = Y.map(actions)
Y.head()
print(Y.value_counts())
```

```
19    1000
11    1000
18    1000
4     1000
5     1000
14    1000
9     1000
12    1000
13     997
15     983
6      983
0      982
1      979
2      978
```

```
Handshaking      962
Name: Action, dtype: int64
```

```
Y = data["Action"]
X = data.drop(columns = ["Action"])
```

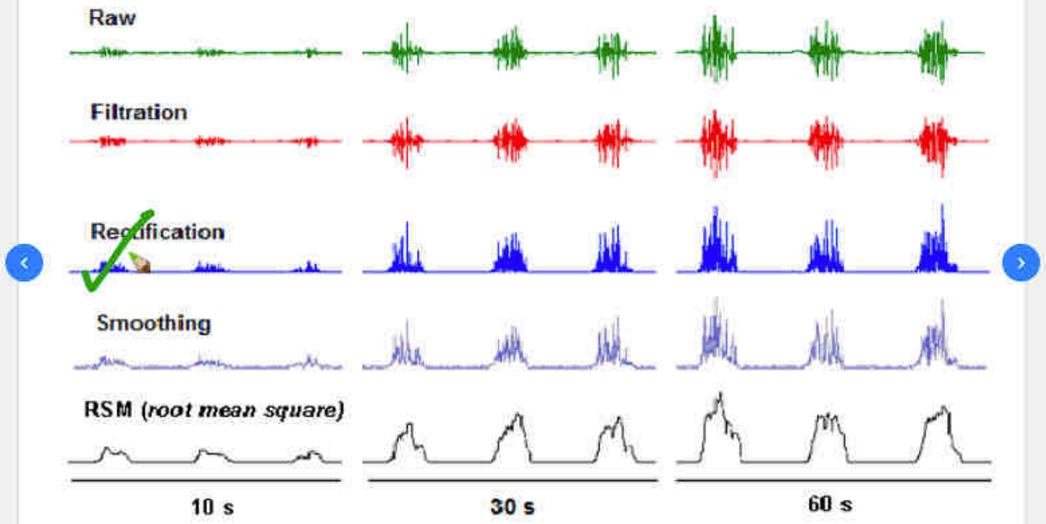
```
# Label encoding
Y = Y.map(actions)
Y.head()
print(Y.value_counts())
```

```
19    1000
11    1000
18    1000
4     1000
5     1000
14    1000
9     1000
12    1000
13     997
15     983
6      983
0      982
1      979
2      978
```

```
1       979
2       978
10      976
16      973
7       968
3       966
8       964
17      962
Name: Action, dtype: int64
```

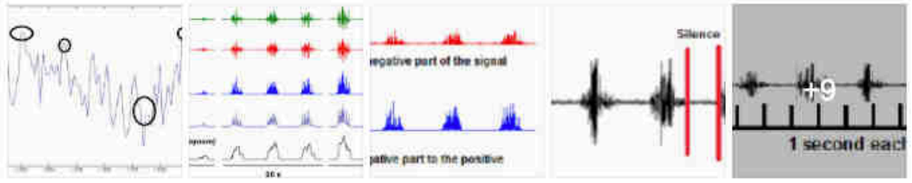
```
# Domain specific pre-processing
X = abs(X)
X.head()
```

	ch1	ch2	ch3	ch4	ch5	ch6	ch7	ch8
0	328.0	433.0	533.0	210.0	3918.0	961.0	3193.0	4000.0
1	577.0	309.0	1550.0	21.0	4000.0	707.0	2297.0	3719.0
2	712.0	160.0	632.0	149.0	3231.0	1515.0	3294.0	4000.0
3	234.0	68.0	125.0	402.0	2972.0	2428.0	3009.0	1895.0
4	2471.0	118.0	263.0	94.0	4000.0	4000.0	3592.0	613.0



EMG signal process recommended. Green: The raw signal, no treatment was applied until this moment; Red: Filtrated signal, a limit was created for the signal, excluding everything out of it; Blue: Rectified signal, all negative values were transformed in positive ones and added; Purple: the smoothed signal, a linear enveloped was created and the extreme parts of the signal was excluded; Black: The RMS values after all the treatments.

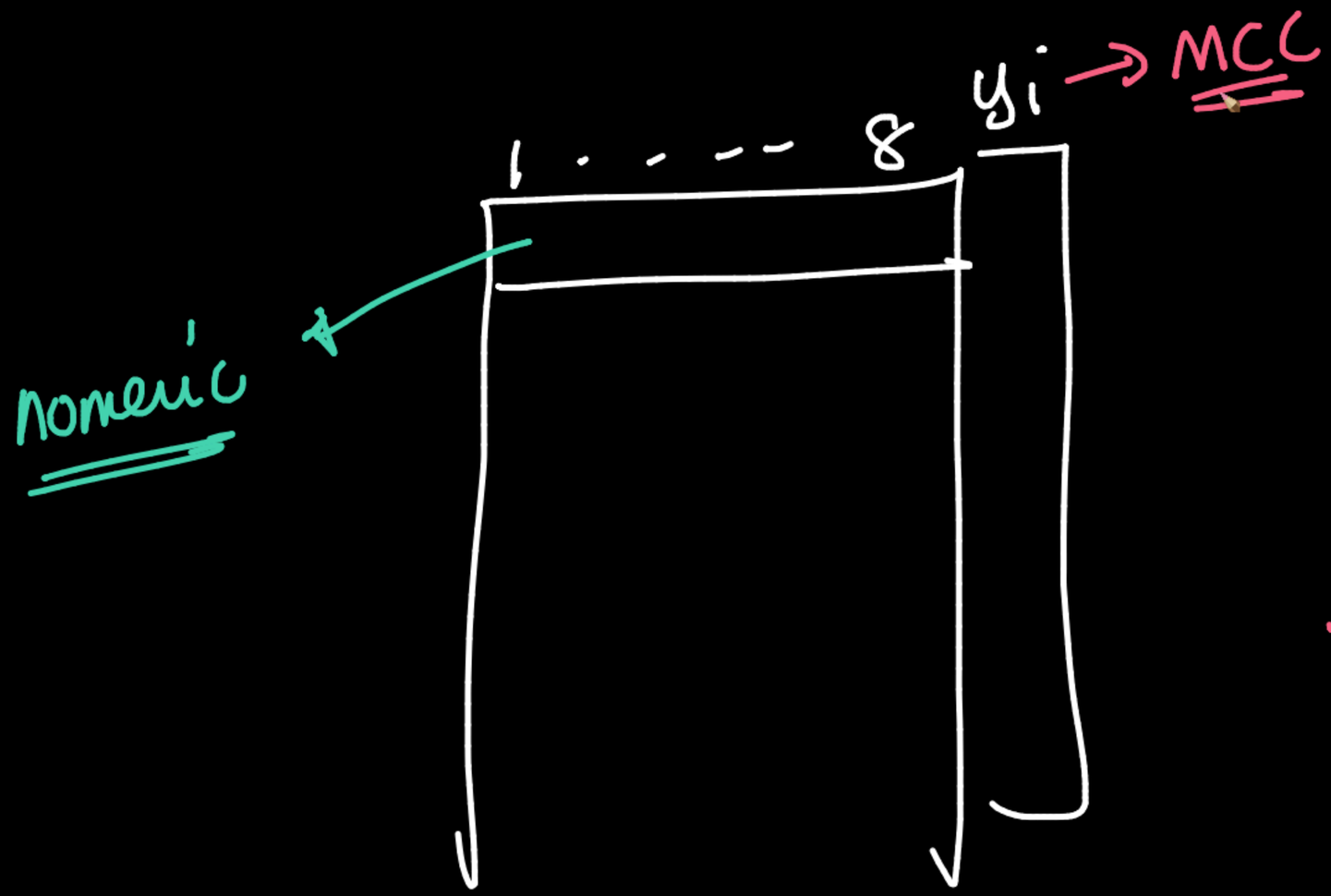
Source publication



Influence of Different Strategies of Treatment Muscle Contraction and Relaxation Phases on EMG Signal Processing and Analysis During Cyclic Exercise

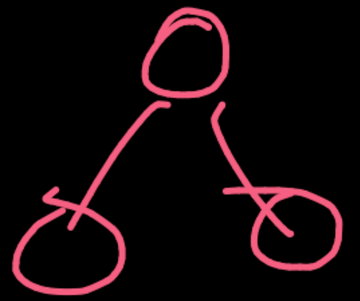
Chapter Full-text available Oct 2013

Leandro Altimari ·



DT / RF / GBDT

X scaling (?)



```
[ ] # Train, test split
    # no CV?
    from sklearn.model_selection import train_test_split

    X = np.array(X.values.tolist())
    Y = np.array(Y.values.tolist())

    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, shuffle = True)
    print(f"Sizes of the sets created are:\nTraining set:{X_train.shape[0]}\nTest set:{X_test.shape[0]}")
```

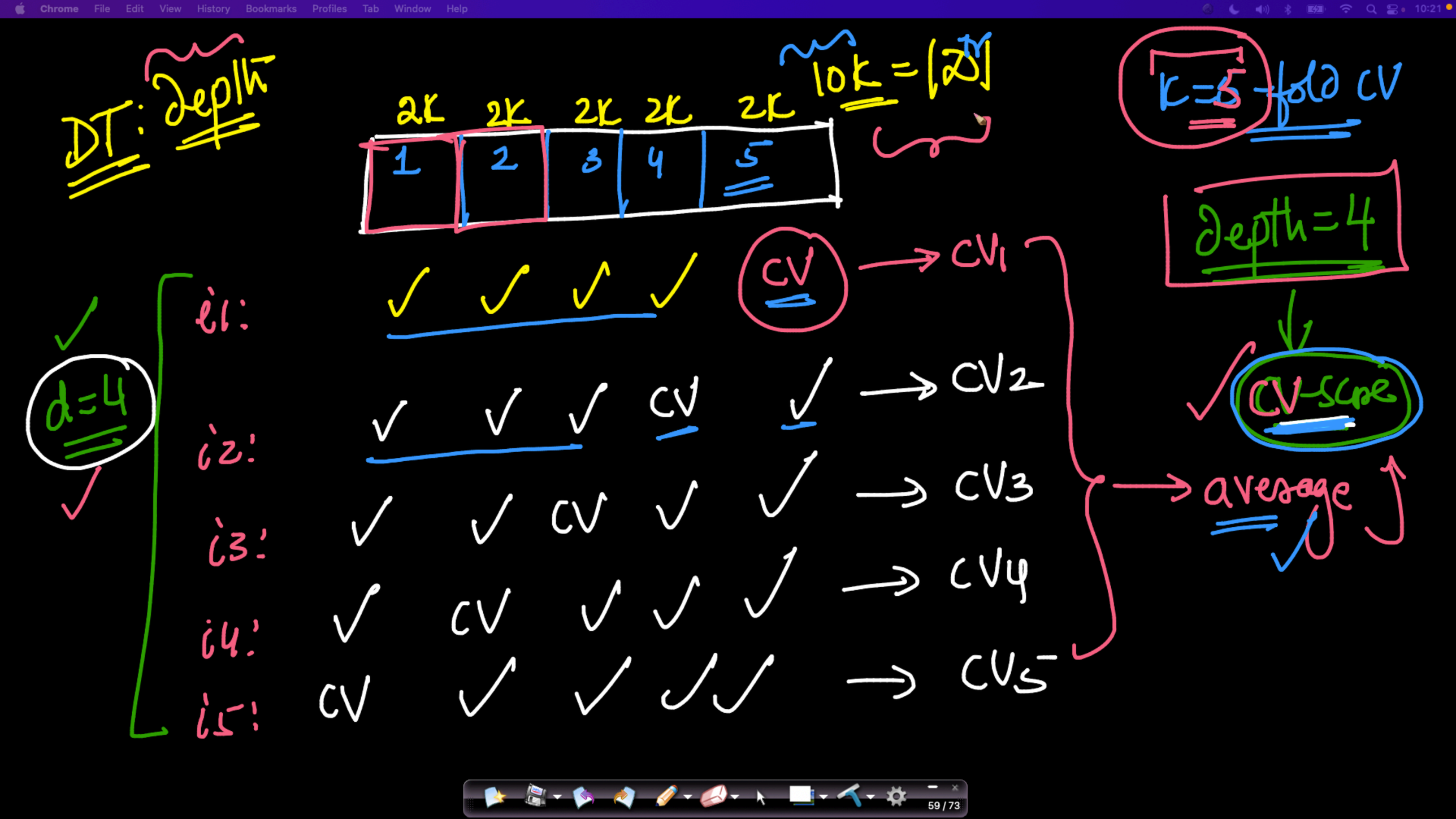
```
Sizes of the sets created are:
Training set:15768
Test set:3943
```

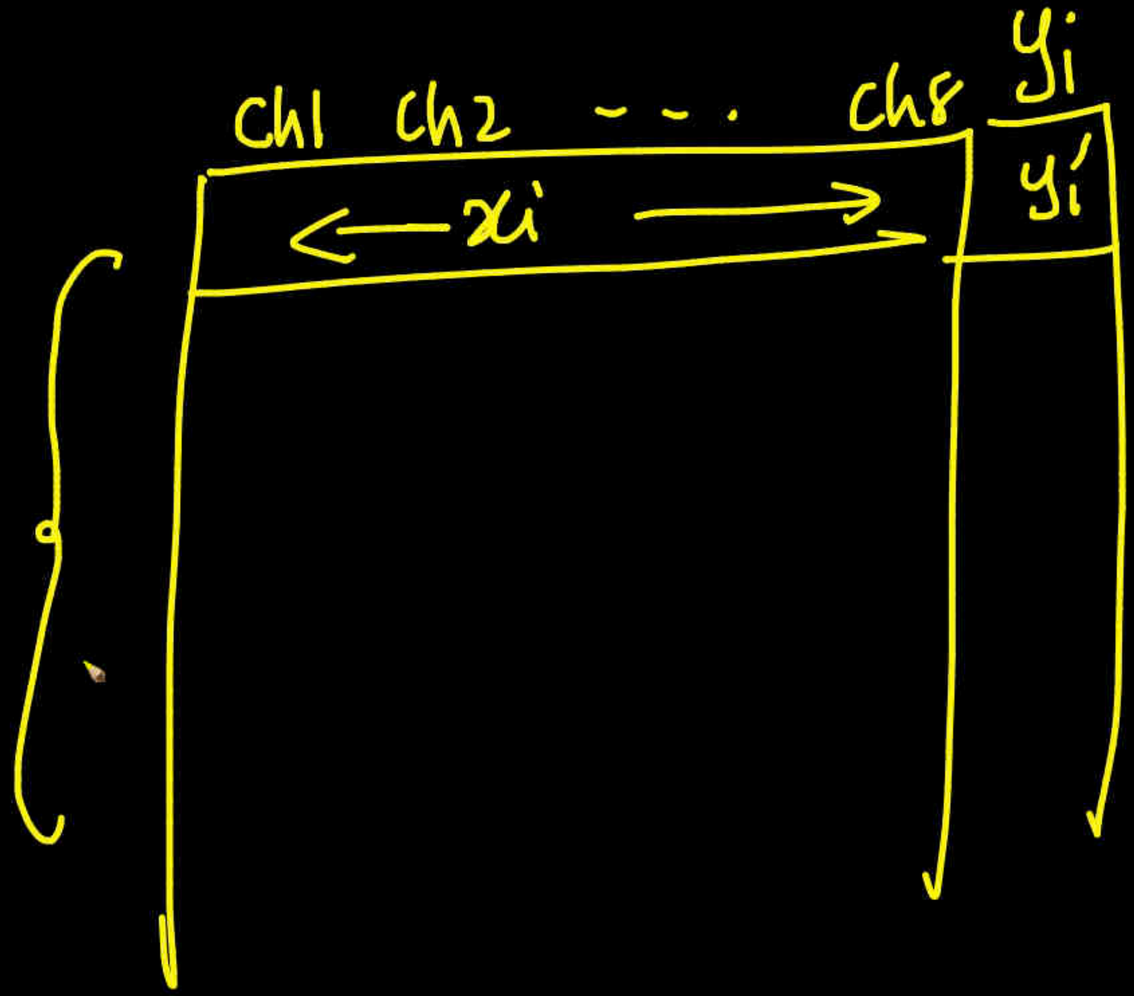
```
# Simple DT
# 5-fold CV
# Grid Search for best hyper-param
from sklearn.tree import DecisionTreeClassifier as DTC
from sklearn import tree
from sklearn.model_selection import GridSearchCV

params = {
```

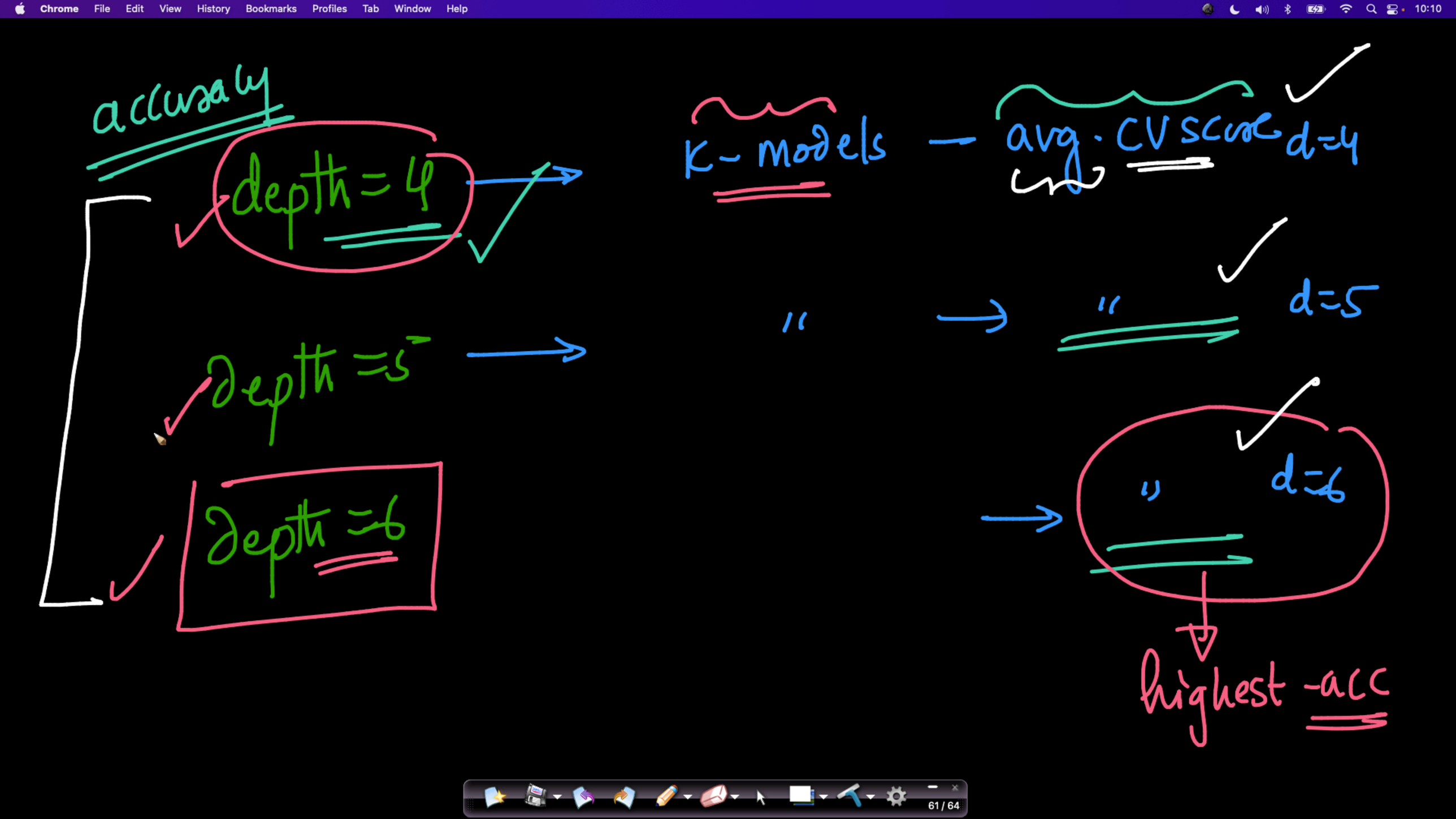
K-fold CV → small dataset sizes

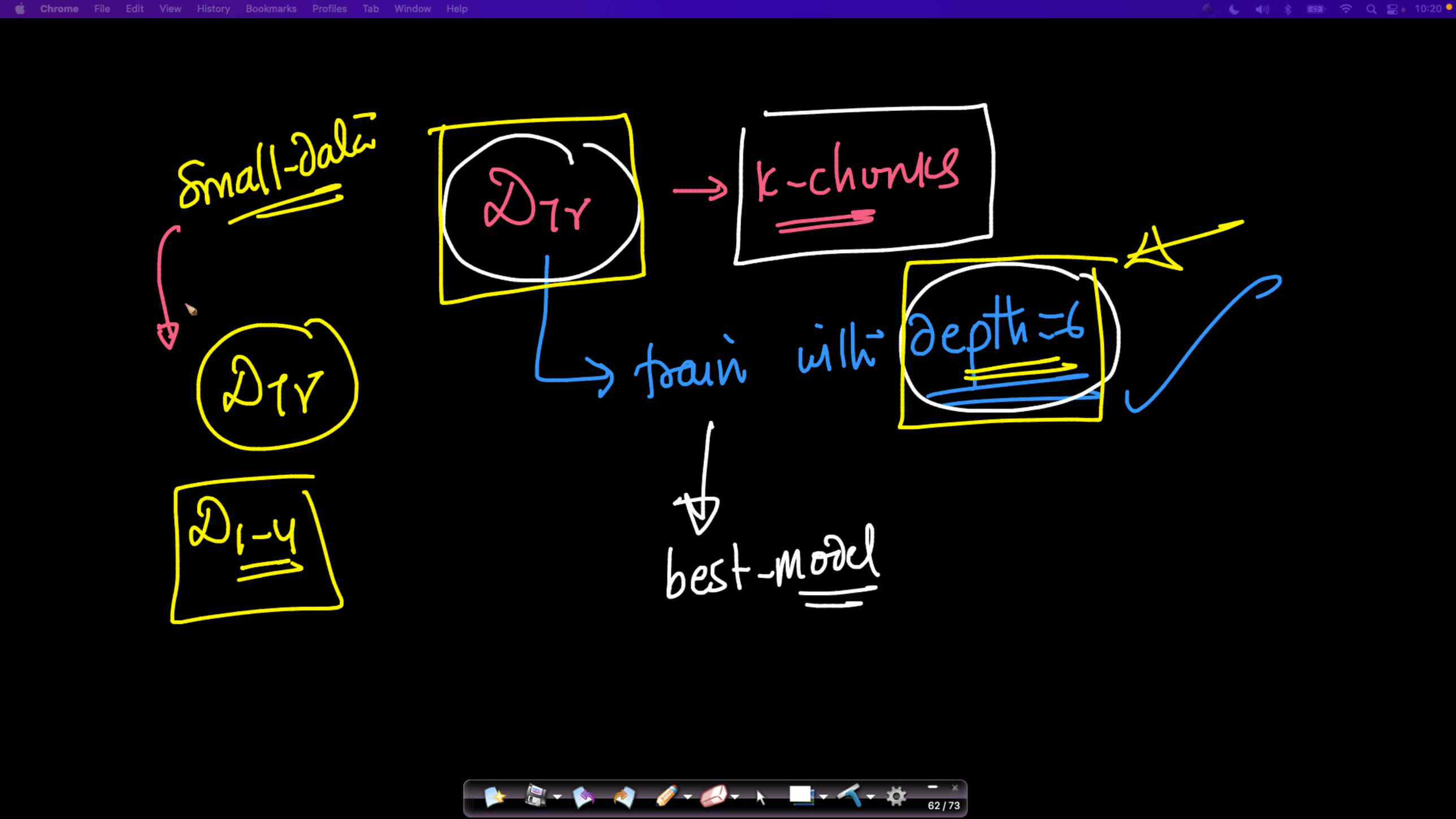
Speed disadv

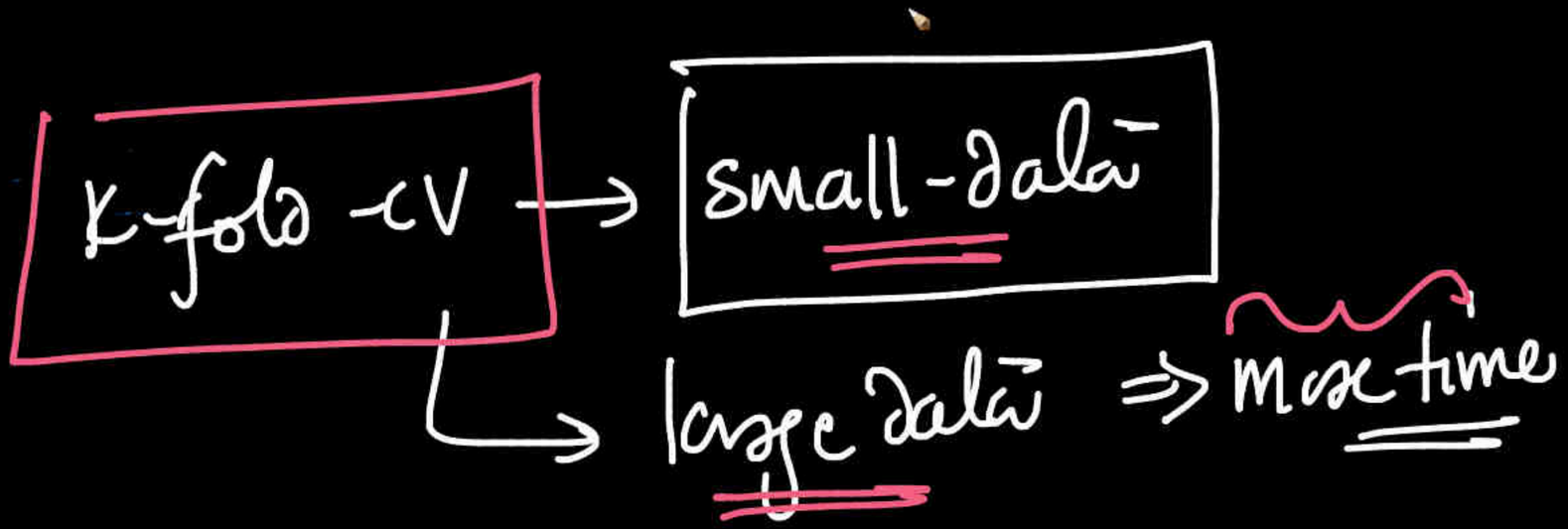
D → DTr → 60%.
D → DCV — 20%.
D → DTe — 20%.


DT: Depth
2K 2K 2K 2K 2K
1 2 3 4 5
10K = |2|
K=5 - fold CV
Depth=4
d=4
i1:
i2:
i3:
i4:
i5:
CV
CV1
CV2
CV3
CV4
CV5
average
CV-score


Ch1 Ch2 ... Ch8
yi
xi
yi


accuracy
depth = 4
depth = 5
depth = 6
K- models
avg. CV score
d=4
"
"
d=5
"
d=6
highest -acc


Small-Data
D_{TR}
D_{TR}
D_{1-4}
k-chunks
train with
depth=6
best-model


K-fold -cV
small-data
large data ⇒ more time

```
[ ]  # Simple DT
     # 5-fold CV
     # Grid Search for best hyper-param
     from sklearn.tree import DecisionTreeClassifier as DTC
     from sklearn import tree
     from sklearn.model_selection import GridSearchCV

     params = {
         "max_depth" : [3, 5, 7],
         "max_leaf_nodes" : [15, 20, 25]
     }

     model1 = DTC()
     clf = GridSearchCV(model1, params, scoring = "accuracy", cv=5)

     clf.fit(X_train, Y_train)
```

```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
             param_grid={'max_depth': [3, 5, 7],
                         'max_leaf_nodes': [15, 20, 25]},
             scoring='accuracy')
```

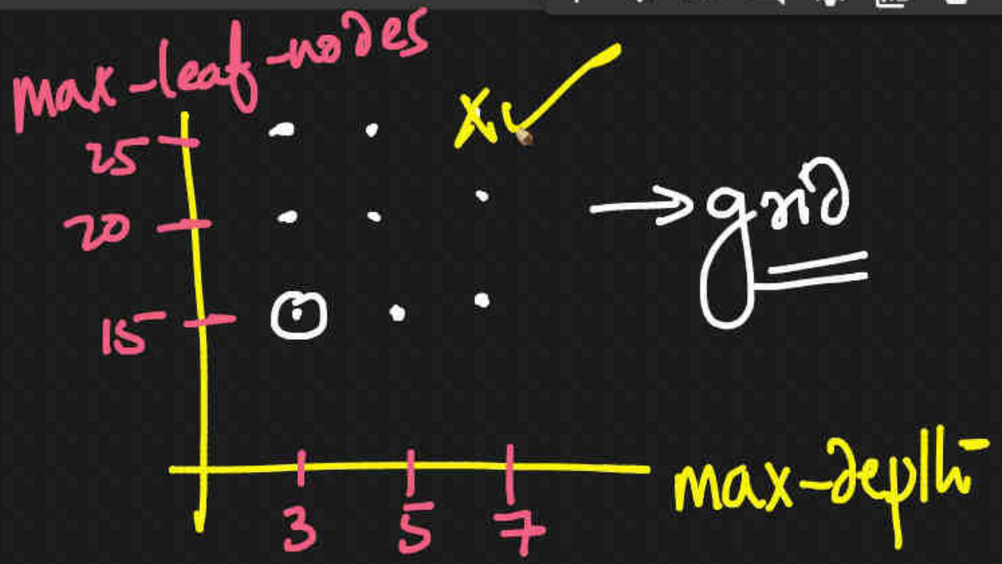
```
[ ]  # Simple DT
     # 5-fold CV
     # Grid Search for best hyper-param
     from sklearn.tree import DecisionTreeClassifier as DTC
     from sklearn import tree
     from sklearn.model_selection import GridSearchCV

     params = {
         "max_depth" : [3, 5, 7],
         "max_leaf_nodes" : [15, 20, 25]
     }

     model1 = DTC()
     clf = GridSearchCV(model1, params, scoring = "accuracy", cv=5)

     clf.fit(X_train, Y_train)
```


```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
             param_grid={'max_depth': [3, 5, 7],
                         'max_leaf_nodes': [15, 20, 25]},
             scoring='accuracy')
```

```
[ ]  # Simple DT
     # 5-fold CV
     # Grid Search for best hyper-param
     from sklearn.tree import DecisionTreeClassifier as DTC
     from sklearn import tree
     from sklearn.model_selection import GridSearchCV

     params = {
         "max_depth" : [3, 5, 7],
         "max_leaf_nodes" : [15, 20, 25]
     }

     model1 = DTC()
     clf = GridSearchCV(model1, params, scoring = "accuracy", cv=5)

     clf.fit(X_train, Y_train)
```

```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
             param_grid={'max_depth': [3, 5, 7],
                         'max_leaf_nodes': [15, 20, 25]},
             scoring='accuracy')
```

```
# Simple DT
# 5-fold CV
# Grid Search for best hyper-param
from sklearn.tree import DecisionTreeClassifier as DTC
from sklearn import tree
from sklearn.model_selection import GridSearchCV

params = {
    "max_depth" : [3, 5, 7],
    "max_leaf_nodes" : [15, 20, 25]
}

model1 = DTC()
clf = GridSearchCV(model1, params, scoring = "accuracy", cv=5)

clf.fit(X_train, Y_train)
```




```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
             param_grid={'max_depth': [3, 5, 7],
                         'max_leaf_nodes': [15, 20, 25]},
             scoring='accuracy')
```

```
[ ] # Simple DT
    # 5-fold CV
    # Grid Search for best hyper-param
    from sklearn.tree import DecisionTreeClassifier as DTC
    from sklearn import tree
    from sklearn.model_selection import GridSearchCV

    params = {
        "max_depth" : [3, 5, 7],
        "max_leaf_nodes" : [15, 20, 25]
    }

    model1 = DTC()
    clf = GridSearchCV(model1, params, scoring = "accuracy", cv=5)

    clf.fit(X_train, Y_train)
```

balanced

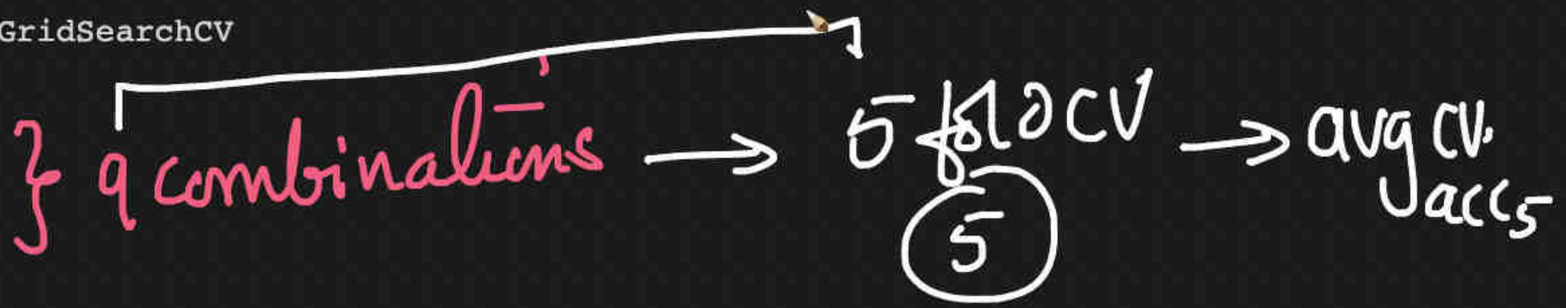
5-fold CV

```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
             param_grid={'max_depth': [3, 5, 7],
                         'max_leaf_nodes': [15, 20, 25]},
             scoring='accuracy')
```

```
# Simple DT
# 5-fold CV
# Grid Search for best hyper-param
from sklearn.tree import DecisionTreeClassifier as DTC
from sklearn import tree
from sklearn.model_selection import GridSearchCV

params = {
    "max_depth" : [3, 5, 7],
    "max_leaf_nodes" : [15, 20, 25]
}

model1 = DTC()
clf = GridSearchCV(model1, params, scoring = "accuracy", cv=5)

clf.fit(X_train, Y_train)
```

```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
             param_grid={'max_depth': [3, 5, 7],
                         'max_leaf_nodes': [15, 20, 25]},
             scoring='accuracy')
```

```
[ ] # Simple DT
    # 5-fold CV
    # Grid Search for best hyper-param
    from sklearn.tree import DecisionTreeClassifier as DTC
    from sklearn import tree
    from sklearn.model_selection import GridSearchCV

    params = {
        "max_depth" : [3, 5, 7],
        "max_leaf_nodes" : [15, 20, 25]
    }

    model1 = DTC()
    clf = GridSearchCV(model1, params, scoring = "accuracy", cv=5)

    clf.fit(X_train, Y_train)
```




```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
             param_grid={'max_depth': [3, 5, 7],
                         'max_leaf_nodes': [15, 20, 25]},
             scoring='accuracy')
```

```
[ ] GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
                 param_grid={'max_depth': [3, 5, 7],
                             'max_leaf_nodes': [15, 20, 25]},
                 scoring='accuracy')
```

mean-CV-score

```
res = clf.cv_results_

for i in range(len(res["params"])):
  print(f"Parameters:{res['params'][i]} Mean_score: {res['mean_test_score'][i]} Rank: {res['rank_test_score'][i]}")
```

```
Parameters:{'max_depth': 3, 'max_leaf_nodes': 15} Mean_score: 0.3192542617764362 Rank: 7
Parameters:{'max_depth': 3, 'max_leaf_nodes': 20} Mean_score: 0.3192542617764362 Rank: 7
Parameters:{'max_depth': 3, 'max_leaf_nodes': 25} Mean_score: 0.3192542617764362 Rank: 7
Parameters:{'max_depth': 5, 'max_leaf_nodes': 15} Mean_score: 0.4054414462899422 Rank: 5
Parameters:{'max_depth': 5, 'max_leaf_nodes': 20} Mean_score: 0.4360091072889887 Rank: 3
Parameters:{'max_depth': 5, 'max_leaf_nodes': 25} Mean_score: 0.44482467905574924 Rank: 2
Parameters:{'max_depth': 7, 'max_leaf_nodes': 15} Mean_score: 0.4054414462899422 Rank: 5
Parameters:{'max_depth': 7, 'max_leaf_nodes': 20} Mean_score: 0.42288167140996247 Rank: 4
Parameters:{'max_depth': 7, 'max_leaf_nodes': 25} Mean_score: 0.44562240712059515 Rank: 1
```

```
[ ] print(clf.best_estimator_)
```

```
DecisionTreeClassifier(max_depth=7, max_leaf_nodes=25)
```

```
for i in range(len(res["params"])):
  print(f"Parameters:{res['params'][i]} Mean_score: {res['mean_test_score'][i]} Rank: {res['rank_test_score'][i]}")
```

```
Parameters:{'max_depth': 3, 'max_leaf_nodes': 15} Mean_score: 0.3192542617764362 Rank: 7
Parameters:{'max_depth': 3, 'max_leaf_nodes': 20} Mean_score: 0.3192542617764362 Rank: 7
Parameters:{'max_depth': 3, 'max_leaf_nodes': 25} Mean_score: 0.3192542617764362 Rank: 7
Parameters:{'max_depth': 5, 'max_leaf_nodes': 15} Mean_score: 0.4054414462899422 Rank: 5
Parameters:{'max_depth': 5, 'max_leaf_nodes': 20} Mean_score: 0.4360091072889887 Rank: 3
Parameters:{'max_depth': 5, 'max_leaf_nodes': 25} Mean_score: 0.44482467905574924 Rank: 2
Parameters:{'max_depth': 7, 'max_leaf_nodes': 15} Mean_score: 0.4054414462899422 Rank: 5
Parameters:{'max_depth': 7, 'max_leaf_nodes': 20} Mean_score: 0.42288167140996247 Rank: 4
Parameters:{'max_depth': 7, 'max_leaf_nodes': 25} Mean_score: 0.44552240712059515 Rank: 1
```

```
[ ] print(clf.best_estimator_)
```

```
DecisionTreeClassifier(max_depth=7, max_leaf_nodes=25)
```

```
[ ] # Learning Curves
    from sklearn.model_selection import learning_curve

    def plot_learning_curve(estimator, X, Y, title):

      train_sizes, train_scores, test_scores, _, _ = learning_curve(estimator,
                                                                    X,
```

```
                param_grid={'max_depth': [3, 5, 7],
                            'max_leaf_nodes': [15, 20, 25]},
                scoring='accuracy')
```

```
[ ] res = clf.cv_results_

    for i in range(len(res["params"])):
      print(f"Parameters:{res['params'][i]} Mean_score: {res['mean_test_score'][i]} Rank: {res['rank_test_score'][i]}")
```

```
Parameters:{'max_depth': 3, 'max_leaf_nodes': 15} Mean_score: 0.3192542617764362 Rank: 7
Parameters:{'max_depth': 3, 'max_leaf_nodes': 20} Mean_score: 0.3192542617764362 Rank: 7
Parameters:{'max_depth': 3, 'max_leaf_nodes': 25} Mean_score: 0.3192542617764362 Rank: 7
Parameters:{'max_depth': 5, 'max_leaf_nodes': 15} Mean_score: 0.4054414462899422 Rank: 5
Parameters:{'max_depth': 5, 'max_leaf_nodes': 20} Mean_score: 0.4360091072889887 Rank: 3
Parameters:{'max_depth': 5, 'max_leaf_nodes': 25} Mean_score: 0.44482467905574924 Rank: 2
Parameters:{'max_depth': 7, 'max_leaf_nodes': 15} Mean_score: 0.4054414462899422 Rank: 5
Parameters:{'max_depth': 7, 'max_leaf_nodes': 20} Mean_score: 0.42288167140996247 Rank: 4
Parameters:{'max_depth': 7, 'max_leaf_nodes': 25} Mean_score: 0.4455240712059515 Rank: 1
```

```
[ ] print(clf.best_estimator_)
```

```
DecisionTreeClassifier(max_depth=7, max_leaf_nodes=25)
```

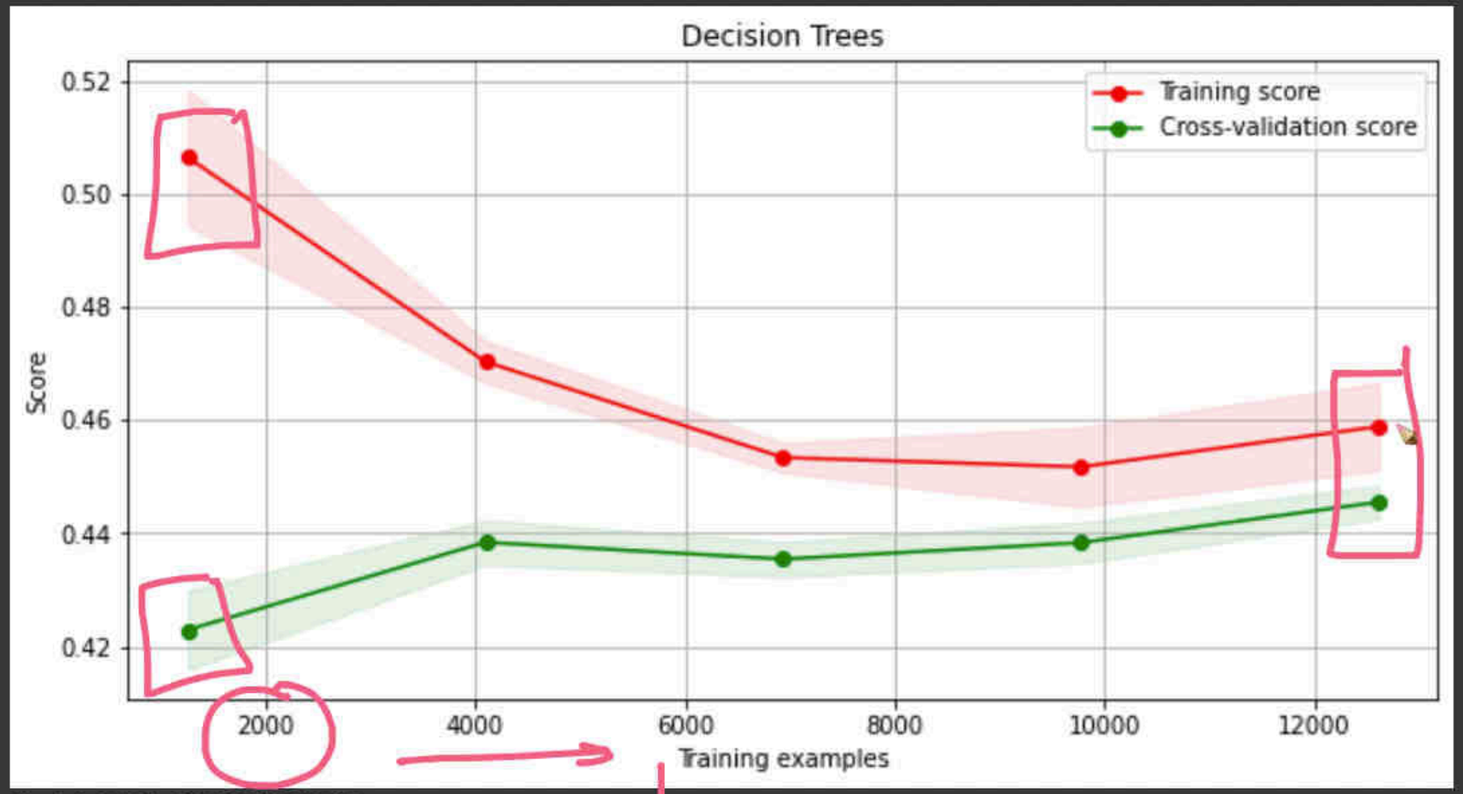
```
[ ] # Learning Curves
    from sklearn.model_selection import learning_curve
```

```
# Simple DT
# 5-fold CV
# Grid Search for best hyper-param
from sklearn.tree import DecisionTreeClassifier as DTC
from sklearn import tree
from sklearn.model_selection import GridSearchCV

params = {
    "max_depth" : [3, 5, 7],
    "max_leaf_nodes" : [15, 20, 25]
}

model1 = DTC()
clf = GridSearchCV(model1, params, scoring = "accuracy", cv=5)

clf.fit(X_train, Y_train)
```

} → 9 combinations → 5-fold CV → mean-error

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-9851e8f8f4ca> in <module>()
     14 clf = GridSearchCV(model1, params, scoring = "accuracy", cv=5)
     15
---> 16 clf.fit(X_train, Y_train)
```

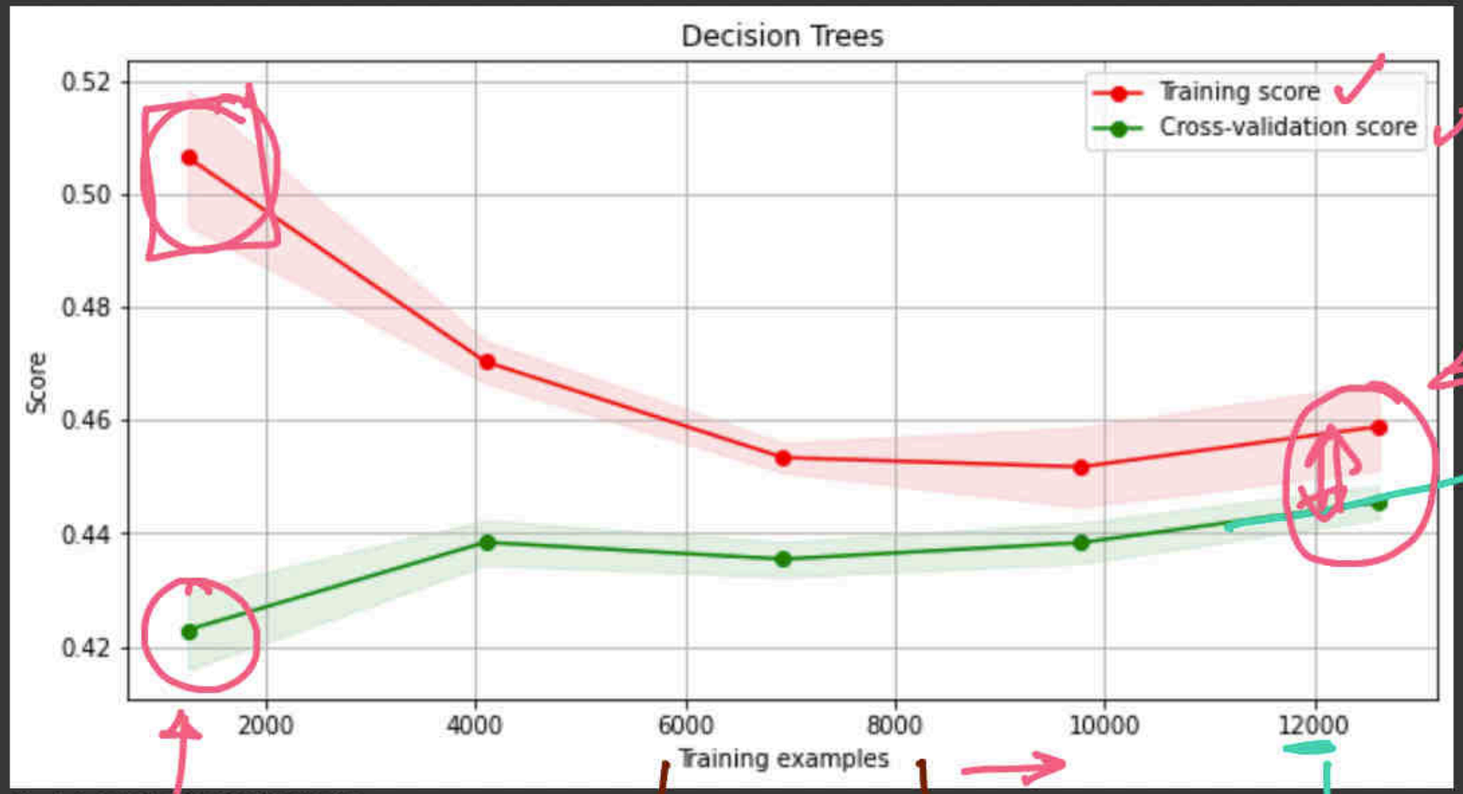
```
[ ] # more data could help as CV-score is improving as datset size increases.
```




```
0.4644216133942161
```

```
[ ] # plot the decision tree
    from sklearn import tree
```

```
[ ]  # more data could help as CV-score is improving as datset size increases.
```



not overfitting

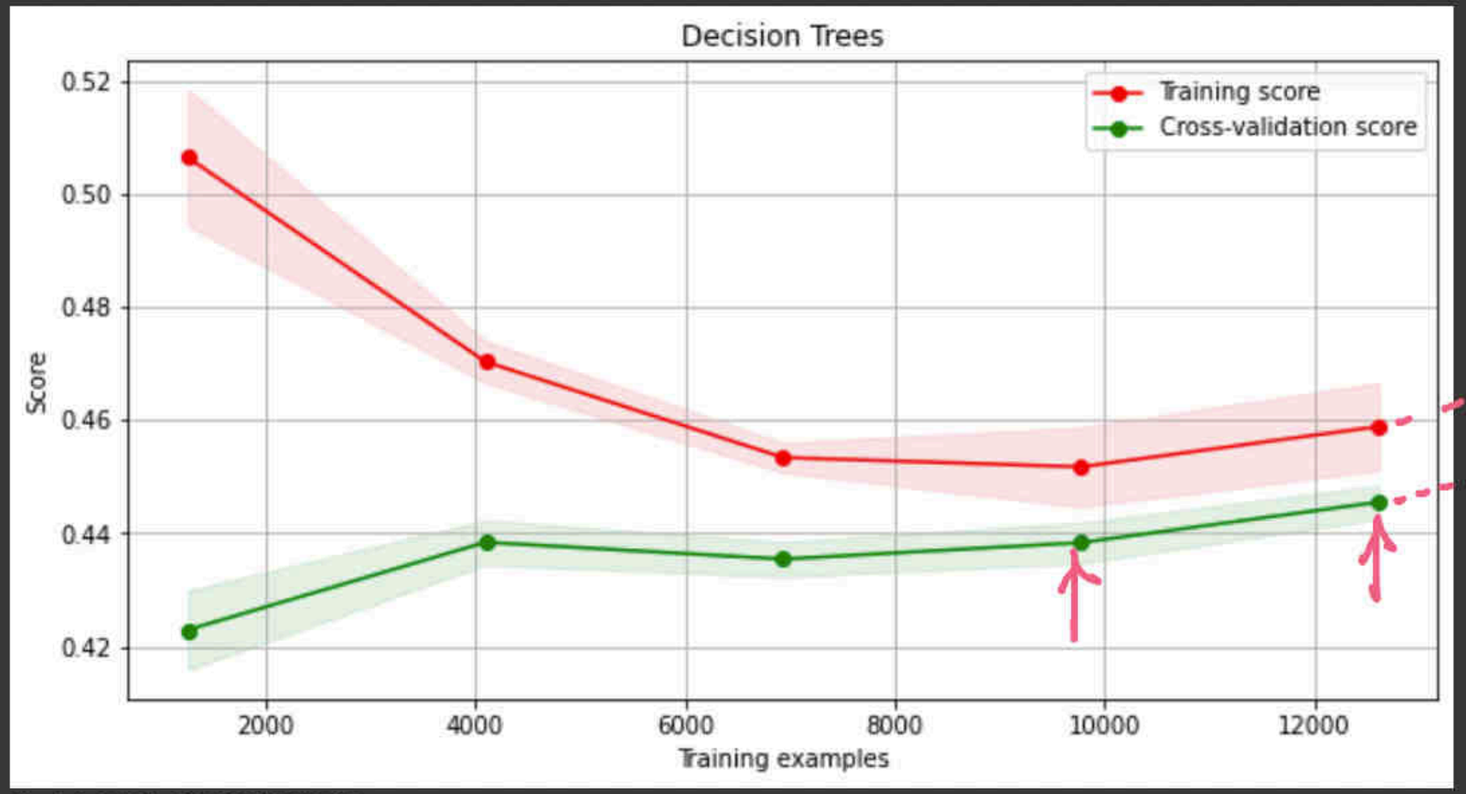
```
0.4644216133942161
```

```
[ ]  # plot the decision tree
     from sklearn import tree
```

+ Code + Text

RAM
Disk

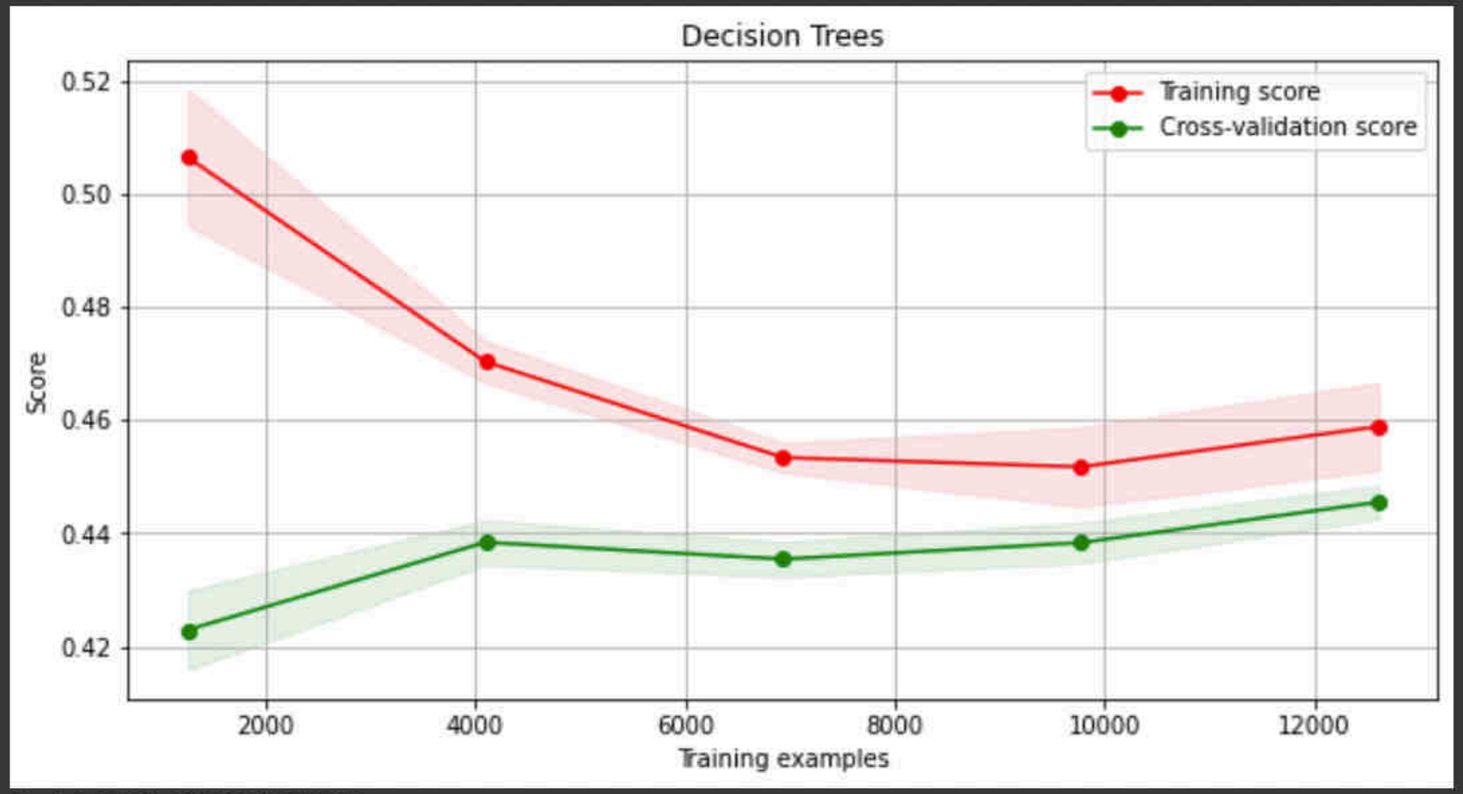
```
[ ]  # more data could help as CV-score is improving as datset size increases.
```




```
0.4644216133942161
```

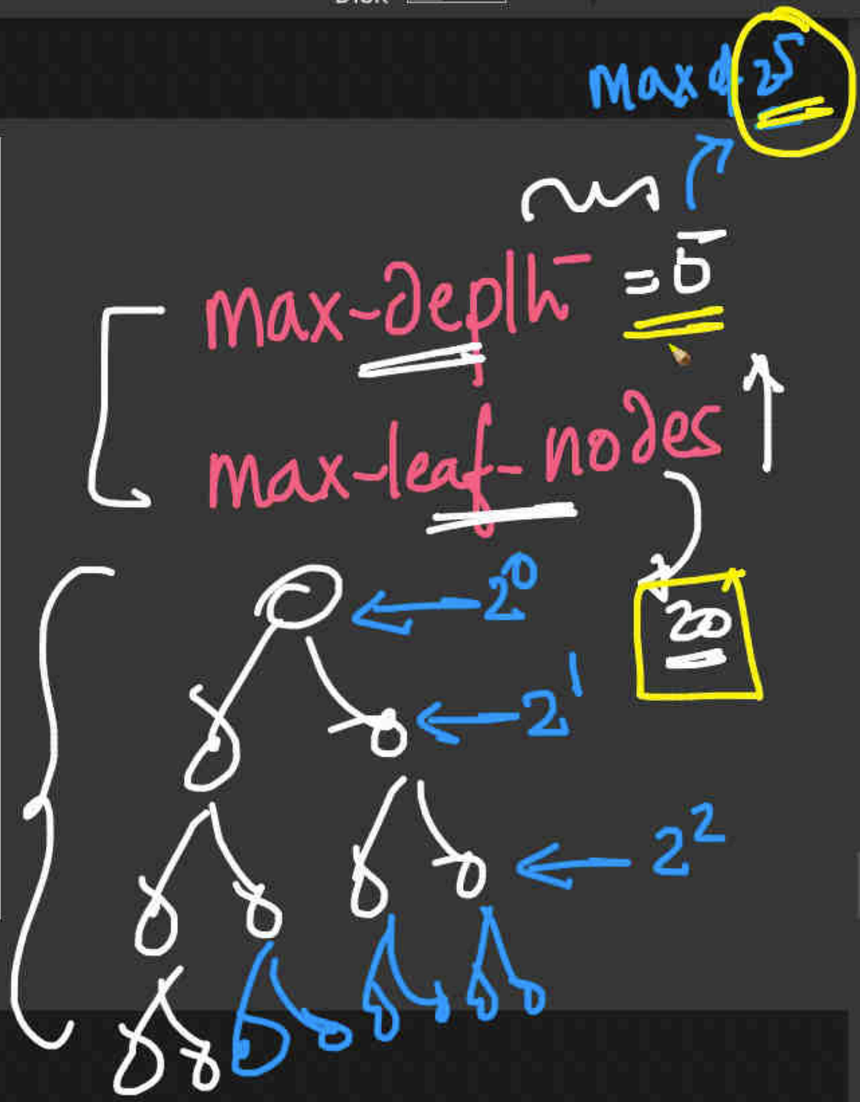
```
[ ]  # plot the decision tree
     from sklearn import tree
```

```
[ ]  # more data could help as CV-score is improving as datset size increases.
```

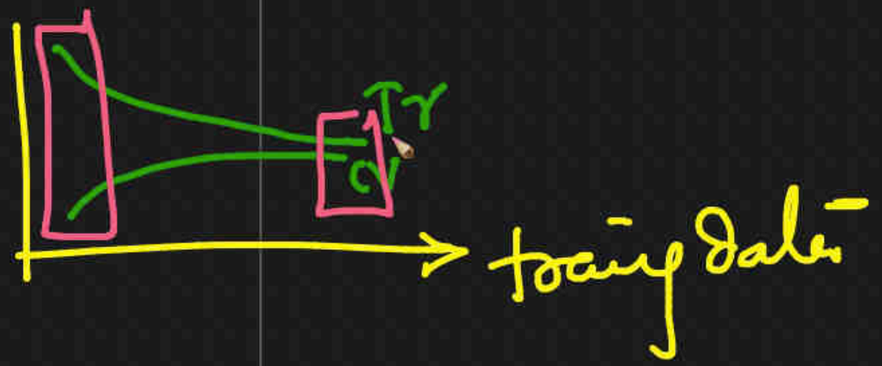



```
0.4644216133942161
```

```
[ ]  # plot the decision tree
     from sklearn import tree
```



```python
# Learning Curves
from sklearn.model_selection import learning_curve

def plot_learning_curve(estimator, X, Y, title):

  train_sizes, train_scores, test_scores, _, _ = learning_curve(estimator,
                                                                X,
                                                                Y,
                                                                return_times=True
                                                                )


  fig, axes = plt.subplots(1, 1, figsize = (10, 5))

  axes.set_title(title)
  axes.plot
  axes.set_xlabel("Training examples")
  axes.set_ylabel("Score")


  train_scores_mean = np.mean(train_scores, axis=1)
  train_scores_std = np.std(train_scores, axis=1)
  test_scores_mean = np.mean(test_scores, axis=1)
  test_scores_std = np.std(test_scores, axis=1)
```



```
# Learning Curves
from sklearn.model_selection import learning_curve

def plot_learning_curve(estimator, X, Y, title):

  train_sizes, train_scores, test_scores, _, _ = learning_curve(estimator,
                                                                X,
                                                                Y,
                                                                return_times=True
                                                                )


  fig, axes = plt.subplots(1, 1, figsize = (10, 5))

  axes.set_title(title)
  axes.plot
  axes.set_xlabel("Training examples")
  axes.set_ylabel("Score")


  train_scores_mean = np.mean(train_scores, axis=1)
  train_scores_std = np.std(train_scores, axis=1)
  test_scores_mean = np.mean(test_scores, axis=1)
  test_scores_std = np.std(test_scores, axis=1)
```

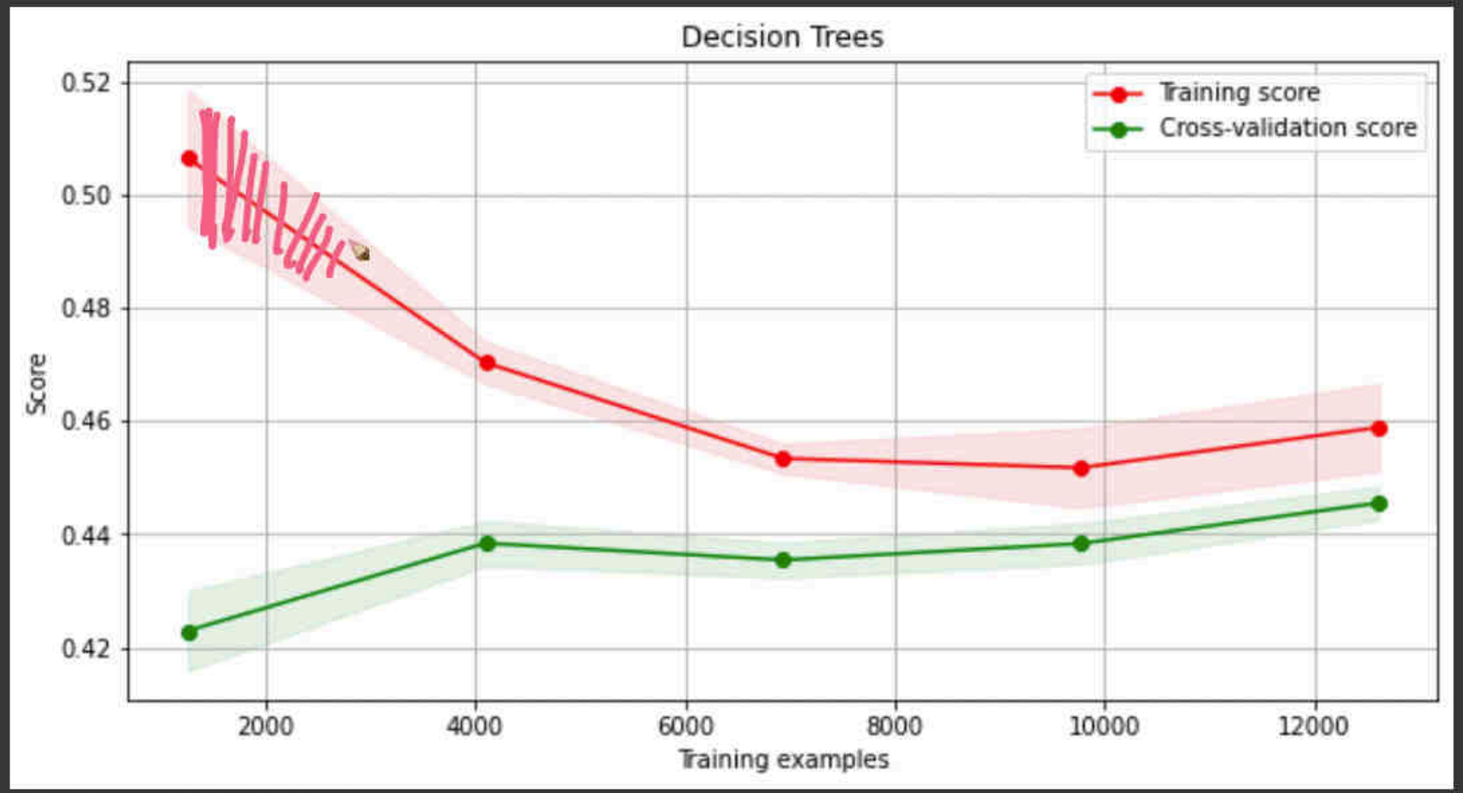
```
Parameters:{'max_depth': 7, 'max_leaf_nodes': 25} Mean_score: 0.44552240712059515 Rank: 1
```

```
[ ] print(clf.best_estimator_)
```

```
DecisionTreeClassifier(max_depth=7, max_leaf_nodes=25)
```

```
[ ] # Learning Curves
    from sklearn.model_selection import learning_curve

    def plot_learning_curve(estimator, X, Y, title):

      train_sizes, train_scores, test_scores, _, _ = learning_curve(estimator,
                                                                    X,
                                                                    Y,
                                                                    return_times=True
                                                                    )


      fig, axes = plt.subplots(1, 1, figsize = (10, 5))

      axes.set_title(title)
      axes.plot
      axes.set_xlabel("Training examples")
      axes.set_ylabel("Score")
```

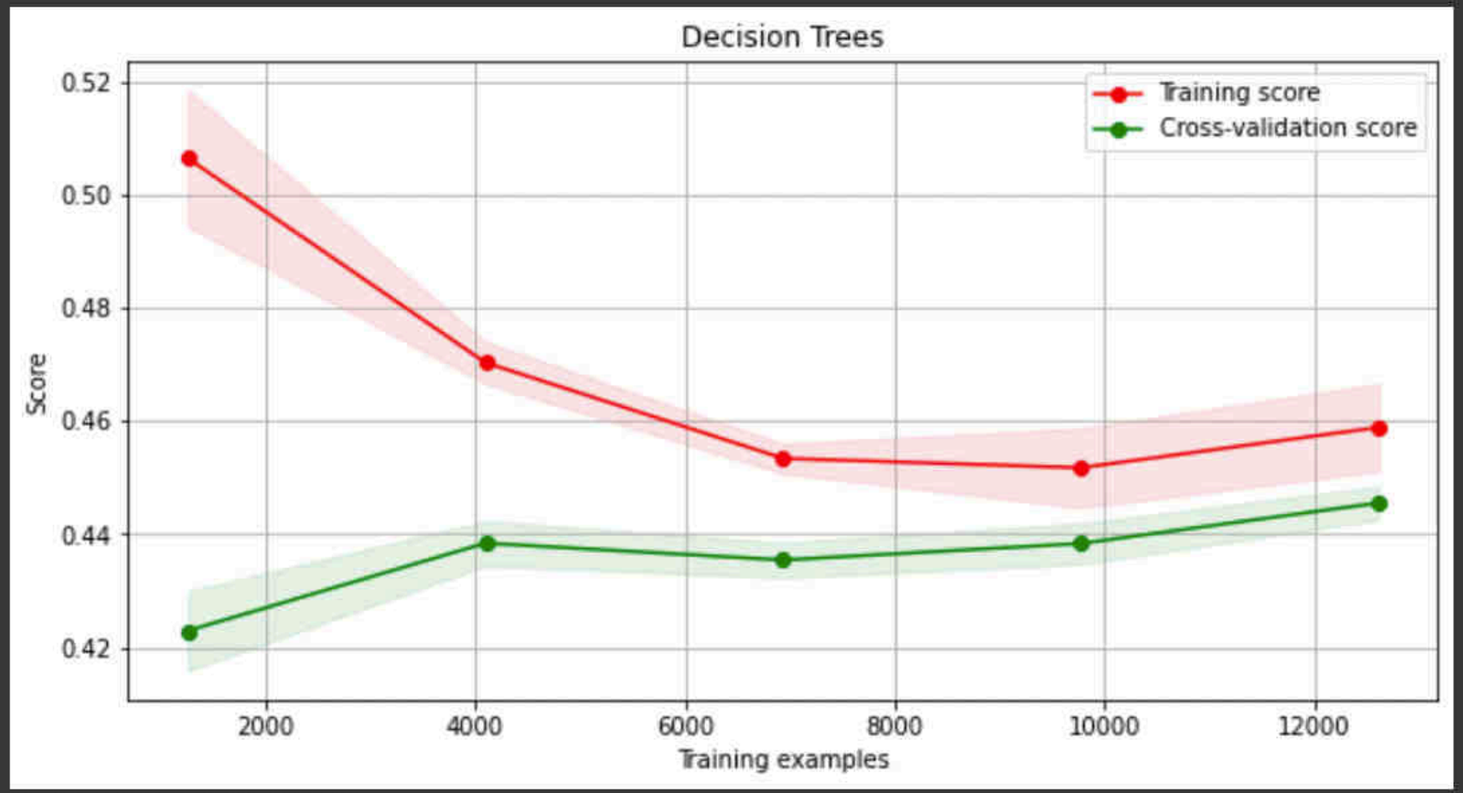
```
print(model1.score(X_train, y_train));

# more data could help as CV-score is improving as datset size increases.
```




```
0.4644216133942161
```

```
[ ]  # plot the decision tree
     from sklearn import tree
```

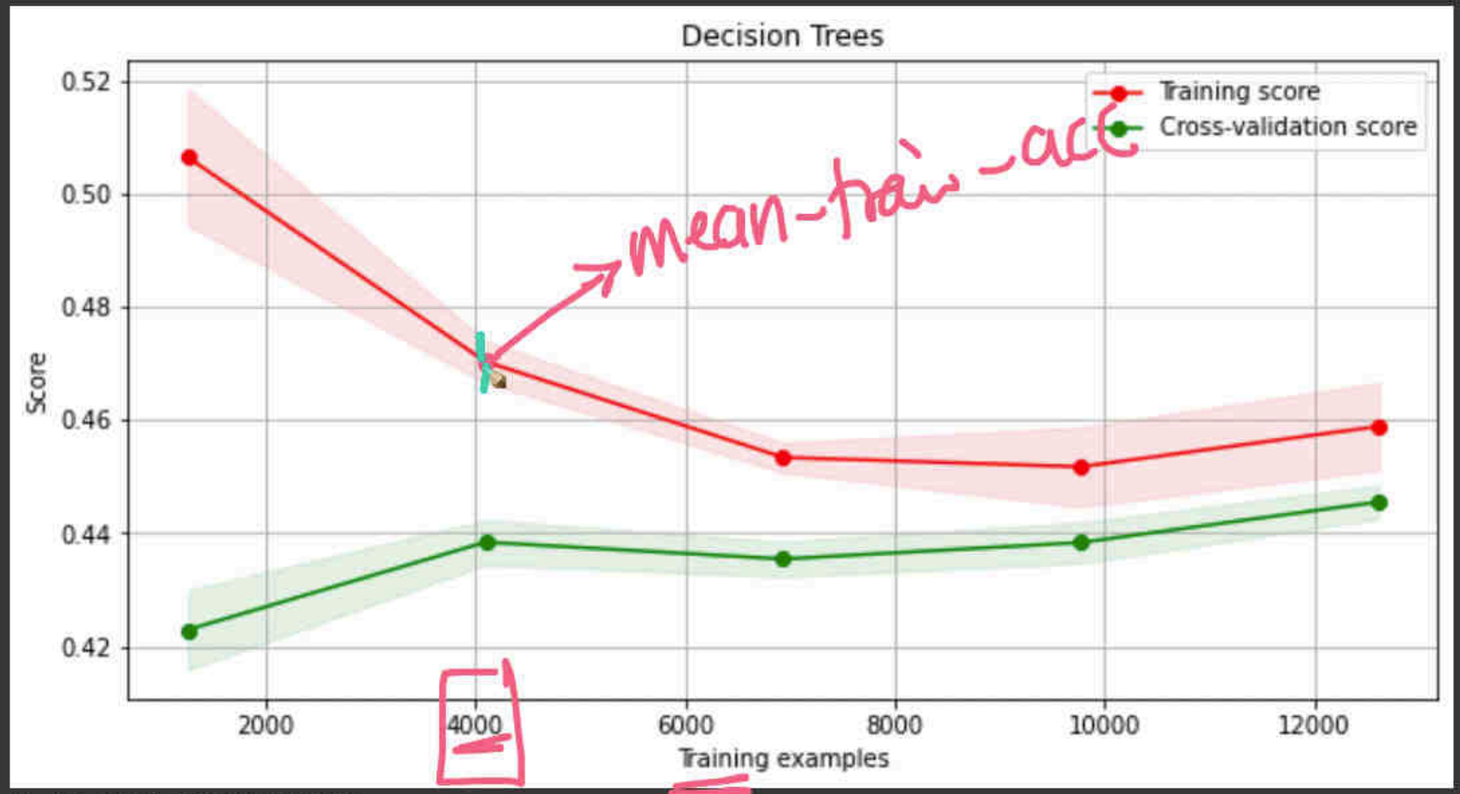
```
print(model1.score(X_train, y_train));

# more data could help as CV-score is improving as datset size increases.
```




```
0.4644216133942161
```

```
[ ]  # plot the decision tree
     from sklearn import tree
```

```
print(model1.score(X_train, y_train))

# more data could help as CV-score is improving as datset size increases.
```



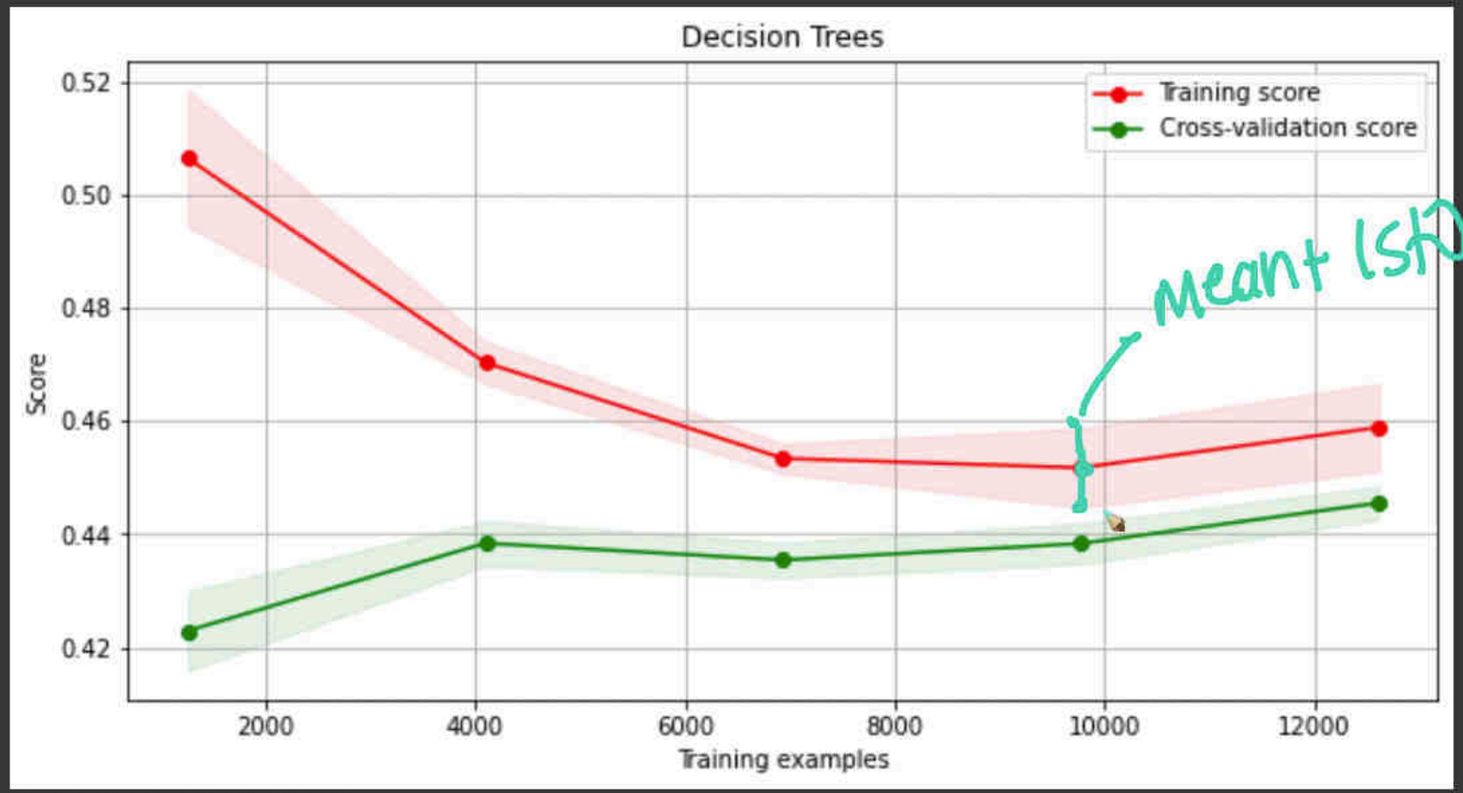
0.4644216133942161

k-fold cv = 5

{ 5 cv-acc
5 train acc

```
[ ] # plot the decision tree
    from sklearn import tree
```

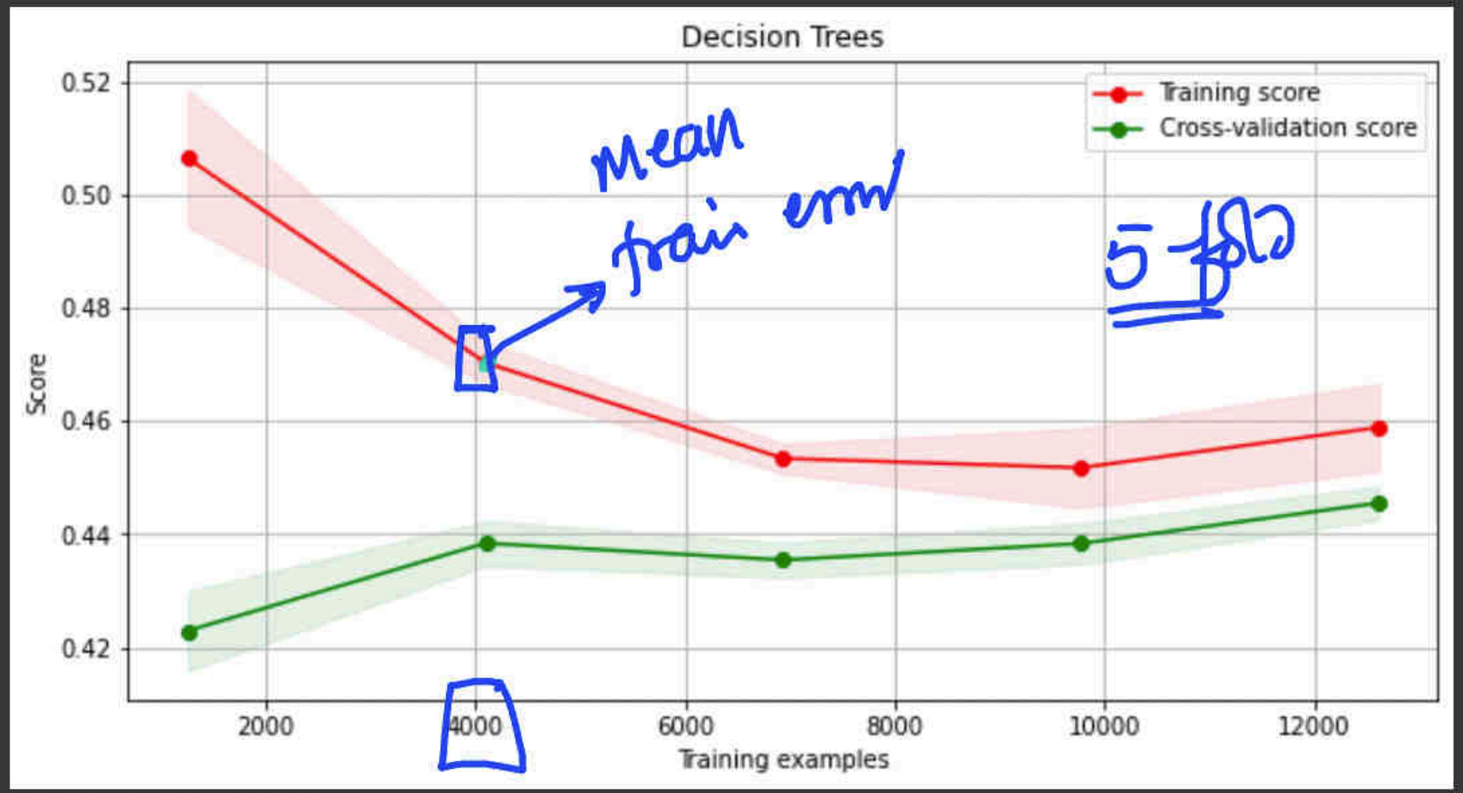
```
print(model1.score(X_train, y_train))

# more data could help as CV-score is improving as datset size increases.
```




```
0.4644216133942161
```

```
[ ]  # plot the decision tree
     from sklearn import tree
```

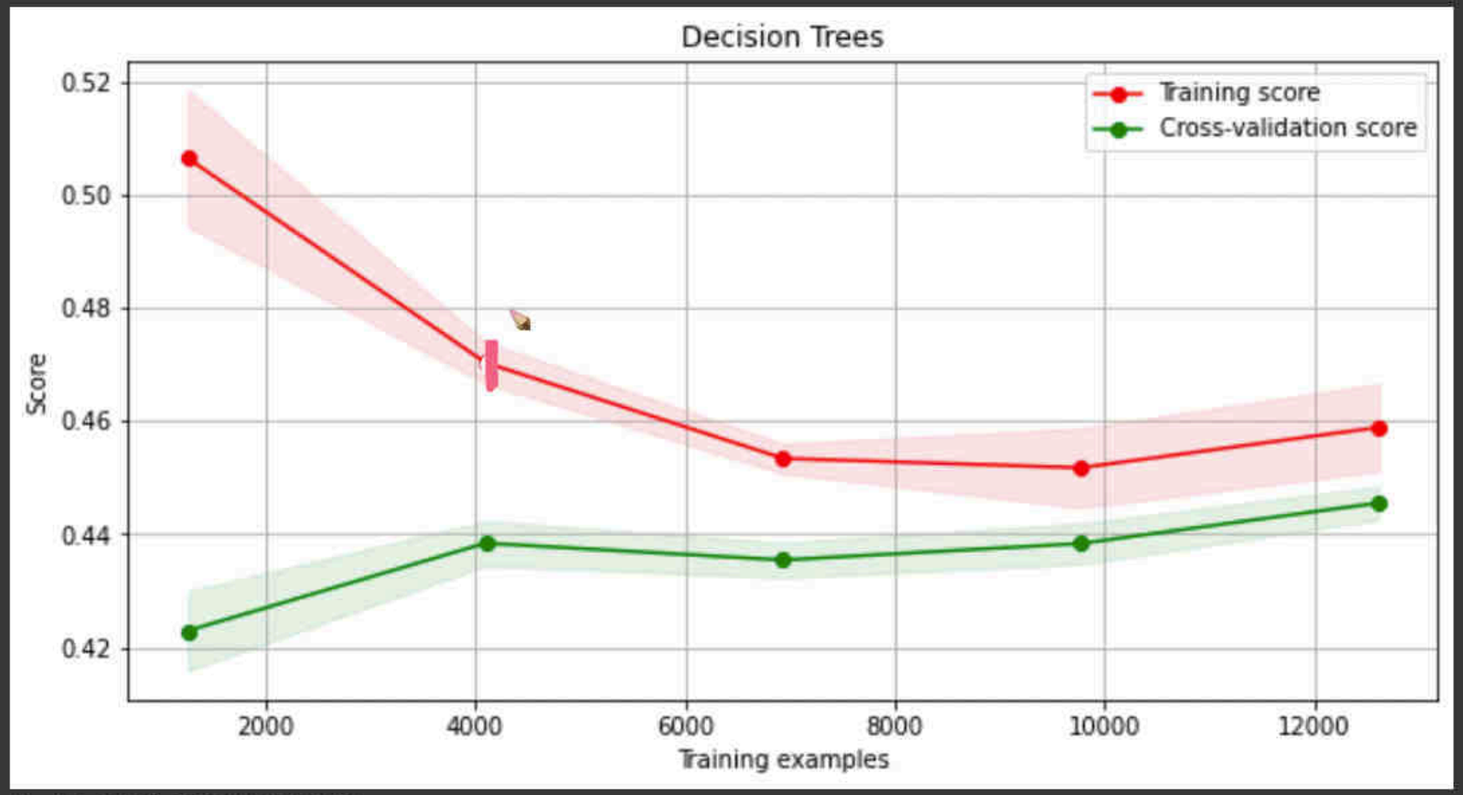
```
# more data could help as CV-score is improving as datset size increases.
```



```
0.4644216133942161
```

```
[ ]  # plot the decision tree
     from sklearn import tree
```

```
print(model1.score(X_train, y_train));

# more data could help as CV-score is improving as datset size increases.
```




```
0.4644216133942161
```

```
[ ]  # plot the decision tree
     from sklearn import tree
```

```
[ ]  from sklearn.model_selection import learning_curve

     def plot_learning_curve(estimator, X, Y, title):

       train_sizes, train_scores, test_scores, _, _ = learning_curve(estimator,
                                                                     X,
                                                                     Y,
                                                                     return_times=True
                                                                     )


       fig, axes = plt.subplots(1, 1, figsize = (10, 5))

       axes.set_title(title)
       axes.plot
       axes.set_xlabel("Training examples")
       axes.set_ylabel("Score")


       train_scores_mean = np.mean(train_scores, axis=1)
       train_scores_std = np.std(train_scores, axis=1)
       test_scores_mean = np.mean(test_scores, axis=1)
       test_scores_std = np.std(test_scores, axis=1)

       # Plot learning curve
       axes.grid()
```

```
axes.set_ylabel( "Score" )
```

[ ]

```
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

# Plot learning curve
axes.grid()
axes.fill_between(
    train_sizes,
    train_scores_mean - train_scores_std,
    train_scores_mean + train_scores_std,
    alpha=0.1,
    color="r",
)
axes.fill_between(
    train_sizes,
    test_scores_mean - test_scores_std,
    test_scores_mean + test_scores_std,
    alpha=0.1,
    color="g",
)
axes.plot(
    train_sizes, train_scores_mean, "o-", color="r", label="Training score"
```

```
    axes.set_ylabel( "Score" )
```

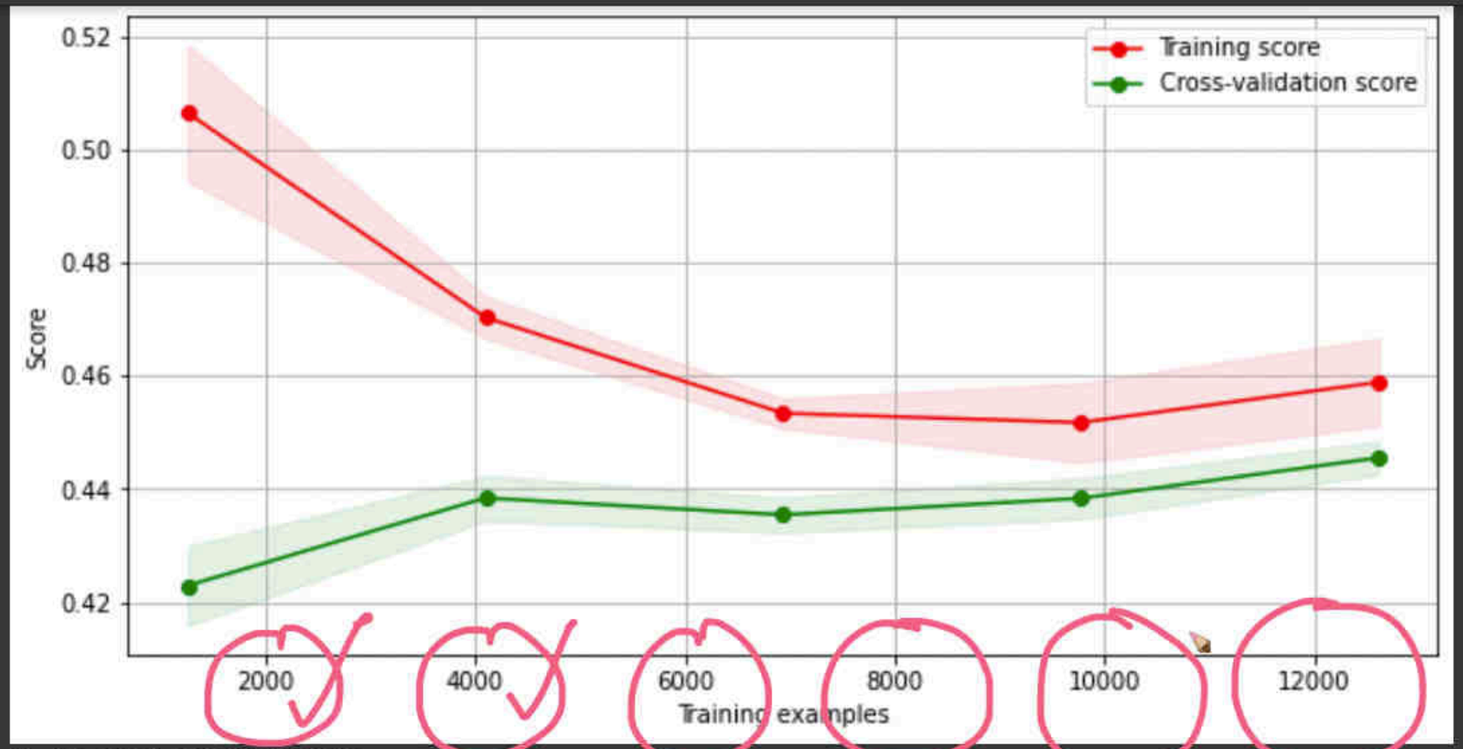
[ ]

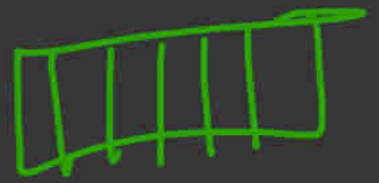
```
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    # Plot learning curve
    axes.grid()
    axes.fill_between(
        train_sizes,
        train_scores_mean - train_scores_std,
        train_scores_mean + train_scores_std,
        alpha=0.1,
        color="r",
    )
    axes.fill_between(
        train_sizes,
        test_scores_mean - test_scores_std,
        test_scores_mean + test_scores_std,
        alpha=0.1,
        color="g",
    )
    axes.plot(
        train_sizes, train_scores_mean, "o-", color="r", label="Training score"
```

+ Code + Text

RAM
Disk

[ ]



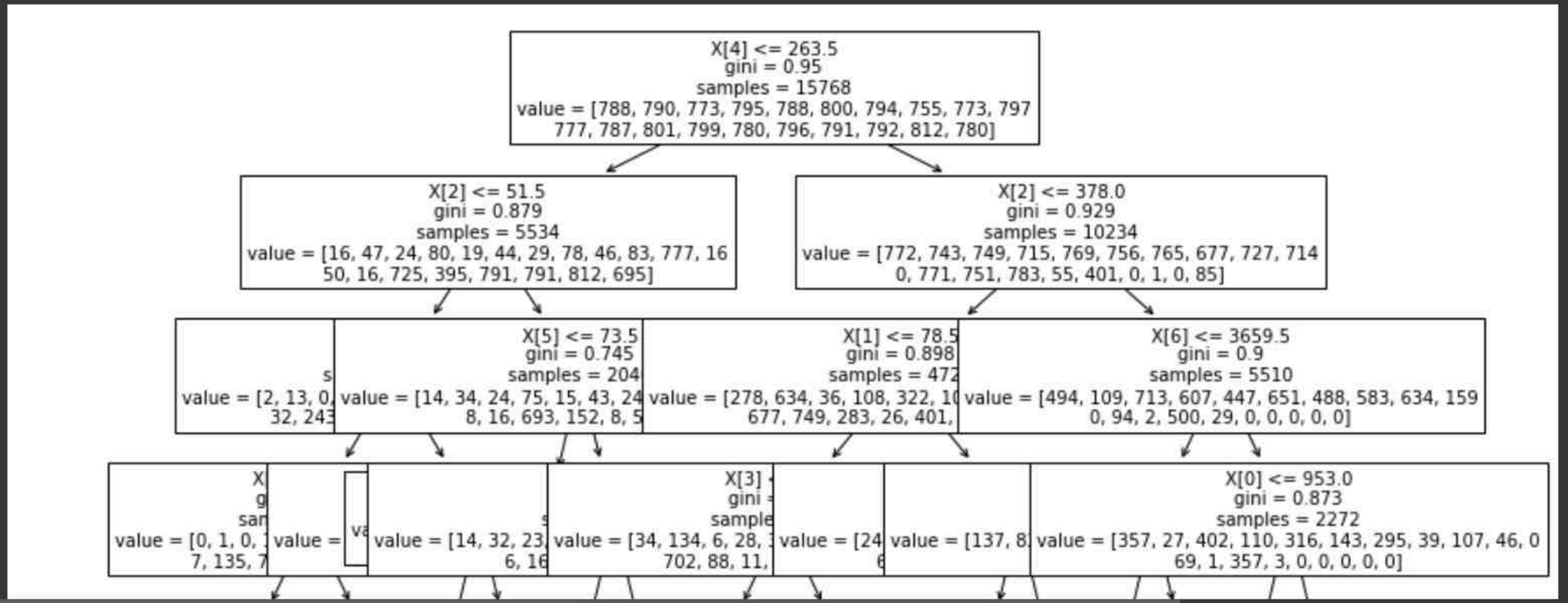


0.4644216133942161

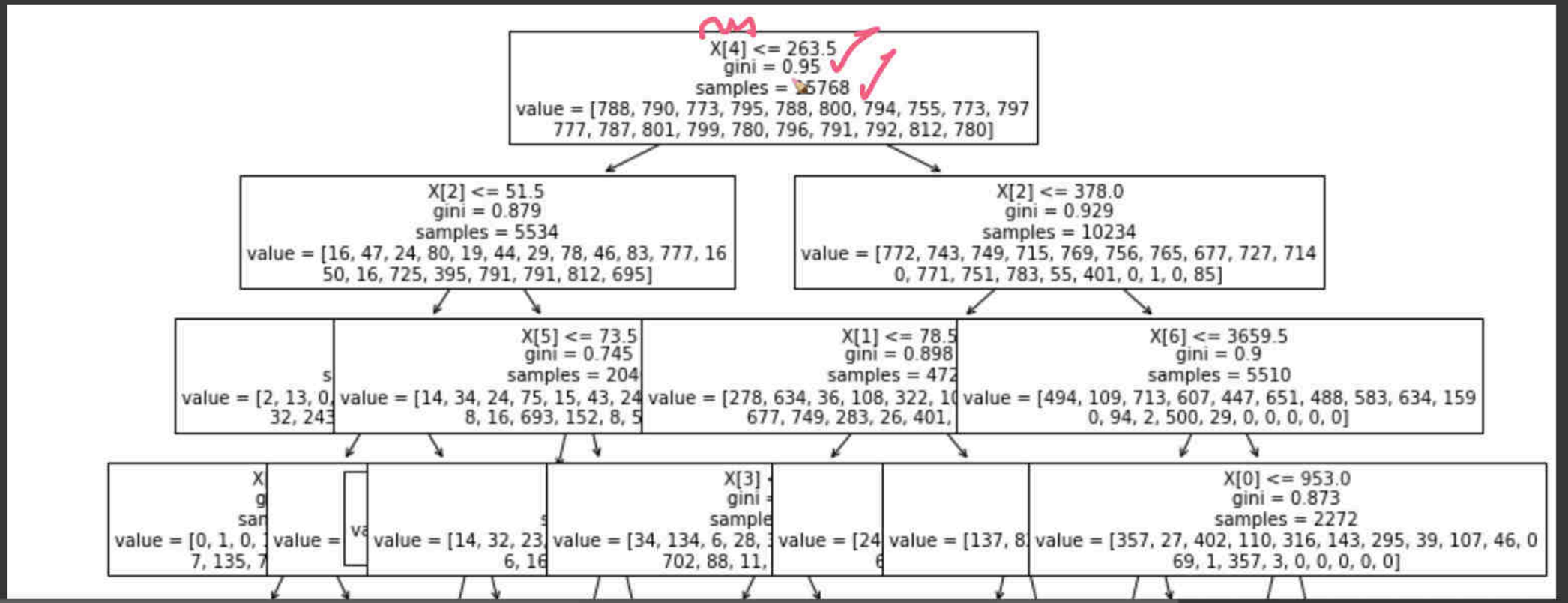
[ ]

```
# plot the decision tree
from sklearn import tree

plt.figure(figsize=(12,12))  # set plot size (denoted in inches)
tree.plot_tree(model1, fontsize=10)
plt.show()
```

```
# plot the decision tree
from sklearn import tree

plt.figure(figsize=(12,12))  # set plot size (denoted in inches)
tree.plot_tree(model1, fontsize=10)
plt.show()
```



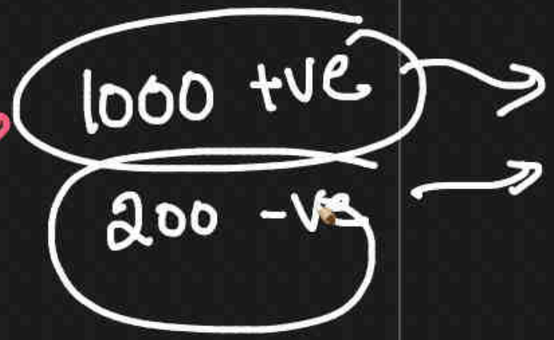
```
# plot the decision tree
from sklearn import tree

plt.figure(figsize=(12,12))  # set plot size (denoted in inches)
tree.plot_tree(model1, fontsize=10)
plt.show()
```



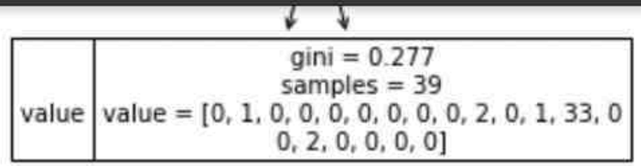
[ ]

value	gini = 0.277 samples = 39 value = [0, 1, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 33, 0 0, 2, 0, 0, 0, 0]

```
# 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(Yn_estimators=100, objective='multi:softmax', num_class=20, silent=True)
```



folds = 3




```
# 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(Yn_estimators=100, objective='multi:softmax', num_class=20, silent=True)
```

folds = 3

```python
# 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(Yn_estimators=100, objective='multi:softmax', num_class=20, silent=True)
```

```python
[ ] folds = 3

    skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)

    random_search = RandomizedSearchCV(xgb, param_distributions=params, n_iter=10, scoring='accuracy', n_jobs=4, cv=skf
```

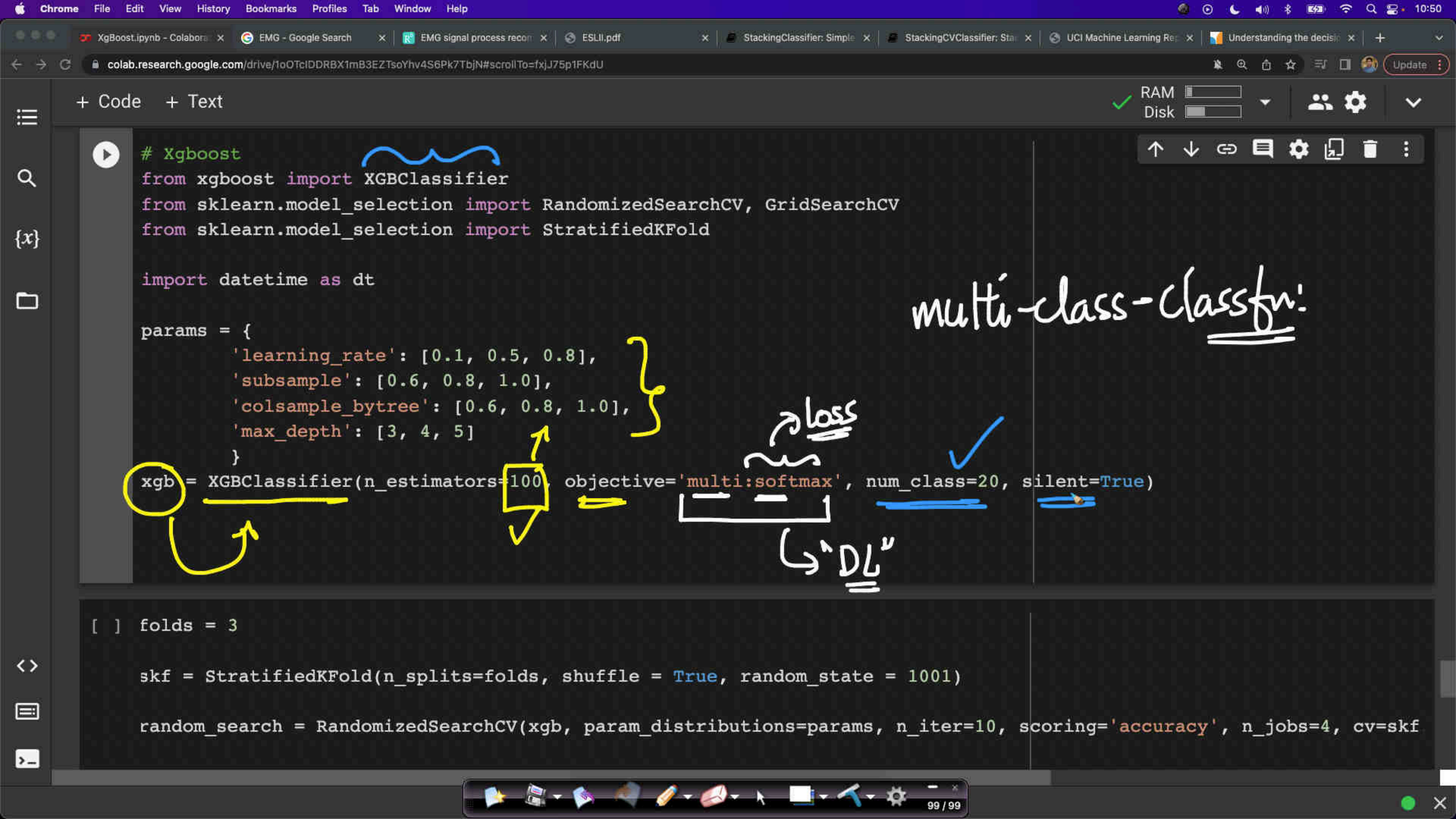
```
# Xgboost
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.model_selection import StratifiedKFold

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(Yn_estimators=100, objective='multi:softmax', num_class=20, silent=True)
```

bagging hr-s

```
[ ] folds = 3

    skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)

    random_search = RandomizedSearchCV(xgb, param_distributions=params, n_iter=10, scoring='accuracy', n_jobs=4, cv=skf
```

```
# 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(Yn_estimators=100, objective='multi:softmax', num_class=20, silent=True)
```

```
[ ] folds = 3

    skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)

    random_search = RandomizedSearchCV(xgb, param_distributions=params, n_iter=10, scoring='accuracy', n_jobs=4, cv=skf
```

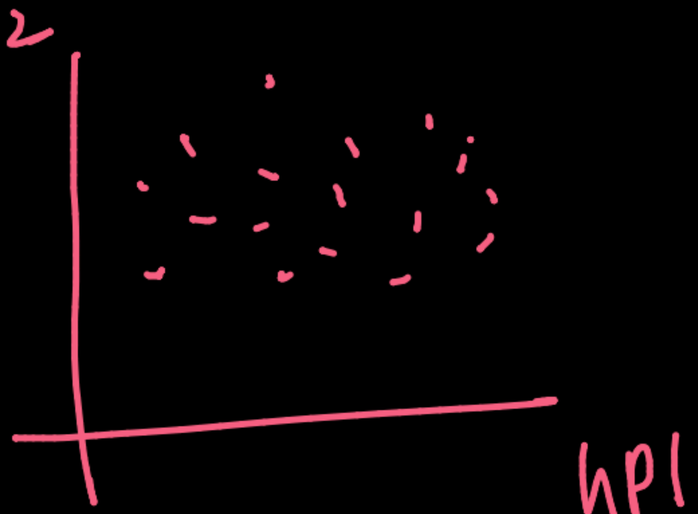

Chrome File Edit View History Bookmarks Profiles Tab Window Help
10:50
XgBoost.ipynb - Colabora
EMG - Google Search
EMG signal process recon
ESLII.pdf
StackingClassifier: Simple
StackingCVClassifier: Sta
UCI Machine Learning Re
Understanding the decisio
colab.research.google.com/drive/1oOTcIDDRBX1mB3EZTsoYhv4S6Pk7TbjN#scrollTo=fxjJ75p1FKdU
Update
+ Code
+ Text
RAM
Disk
# 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)
multi-class-classfn:
"DL"
[ ] folds = 3
skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)
random_search = RandomizedSearchCV(xgb, param_distributions=params, n_iter=10, scoring='accuracy', n_jobs=4, cv=skf
99 / 99

```
# 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)
```

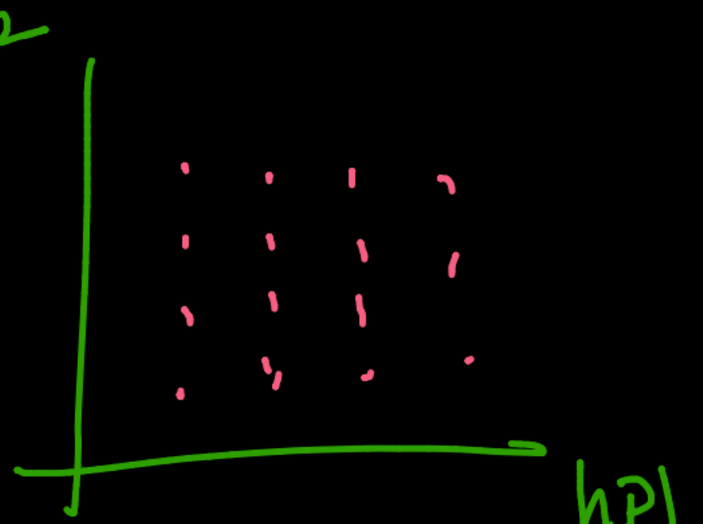
$h_i(x)$

```
folds = 3

skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)

random_search = RandomizedSearchCV(xgb, param_distributions=params, n_iter=10, scoring='accuracy', n_jobs=4, cv=skf
```

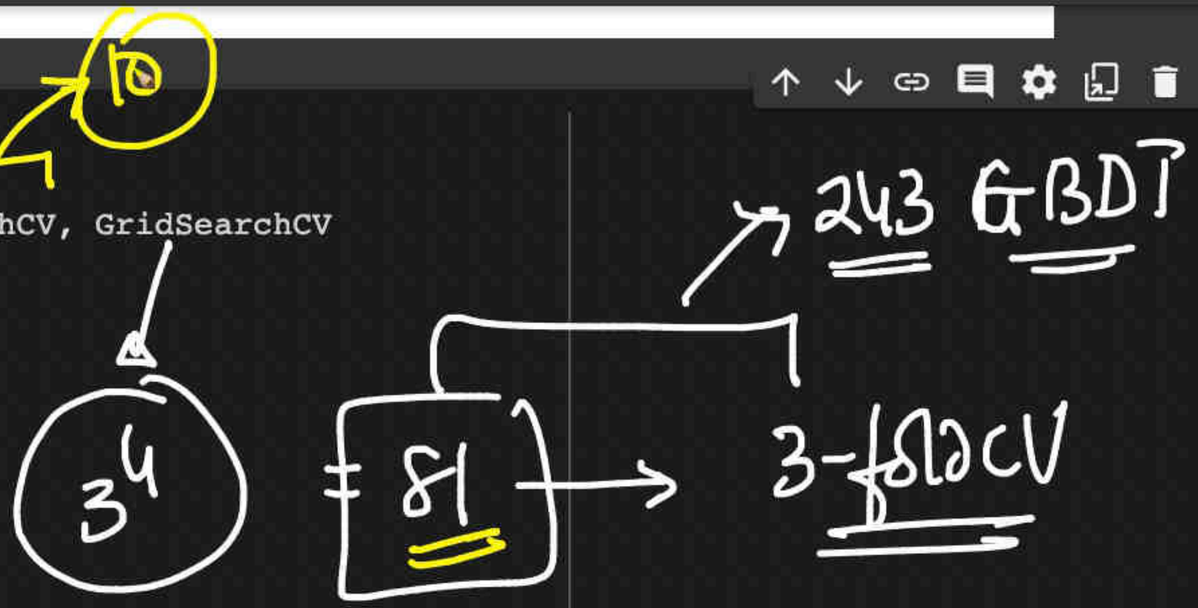
Random Search (DL)



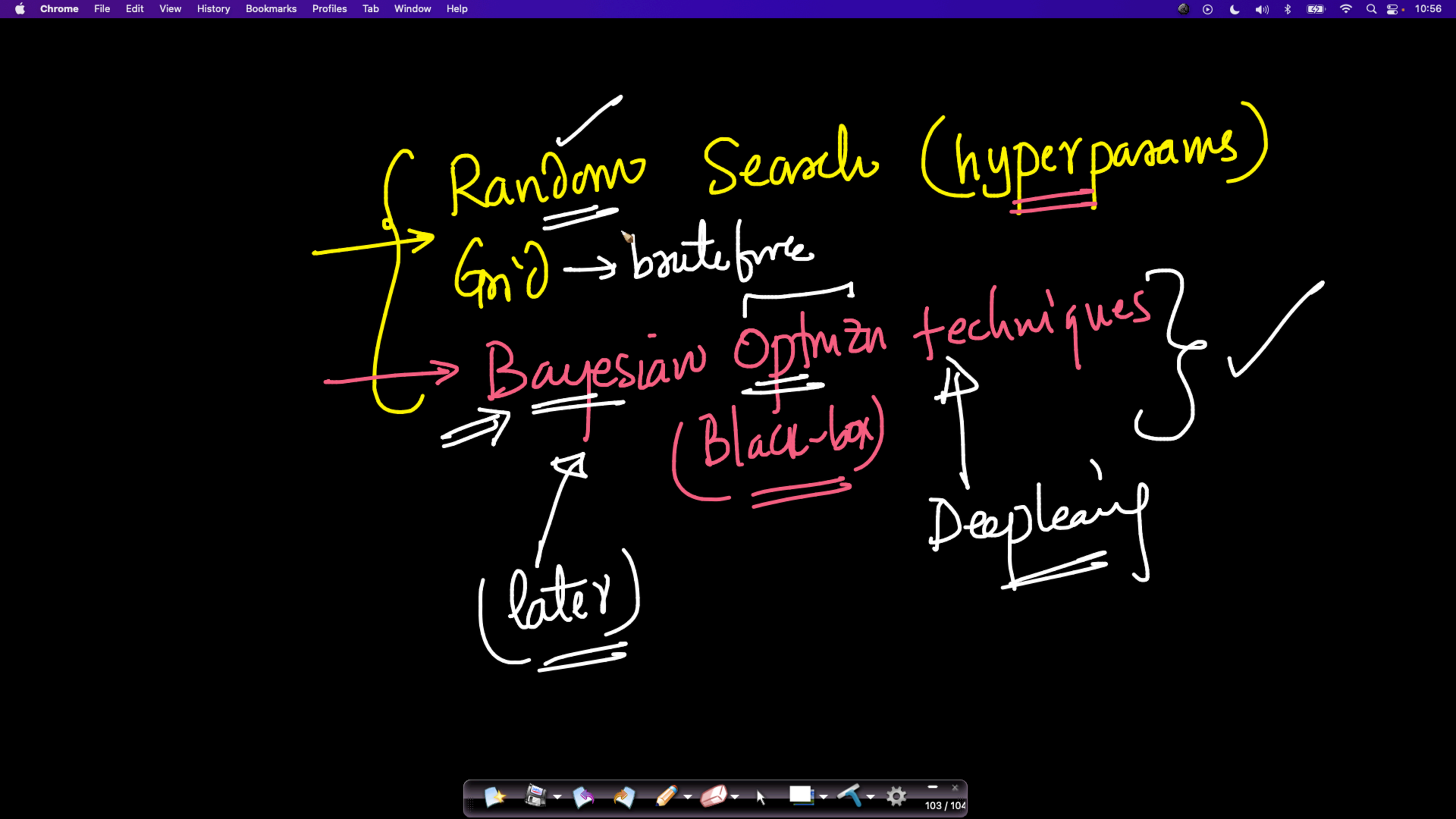
Grid Search!



```
# 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)
```




```
[ ]  folds = 3

     skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)
```


Random Search (hyperparams)
Grid → brute force
Bayesian Optimzn techniques
(Black-box)
Deeplearng
later

```
        '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)
```

```
[ ] folds = 3

skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)

random_search = RandomizedSearchCV(xgb, param_distributions=params, n_iter=10, scoring='accuracy', n_jobs=4, cv=skf


start = dt.datetime.now()
random_search.fit(X_train, Y_train)
end = dt.datetime.now()
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

```
print('\n Best hyperparameters:')
```

```
        '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)
```

```
[ ] folds = 3

    skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)

    random_search = RandomizedSearchCV(xgb, param_distributions=params, n_iter=10, scoring='accuracy', n_jobs=4, cv=skf


    start = dt.datetime.now()
    random_search.fit(X_train, Y_train)
    end = dt.datetime.now()
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

```
print('\n Best hyperparameters:')
```

```
        '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)
```

81 : brute free

```
folds = 3

skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)

random_search = RandomizedSearchCV(xgb, param_distributions=params, n_iter=10, scoring='accuracy', n_jobs=4, cv=skf

start = dt.datetime.now()
random_search.fit(X_train, Y_train)
end = dt.datetime.now()
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

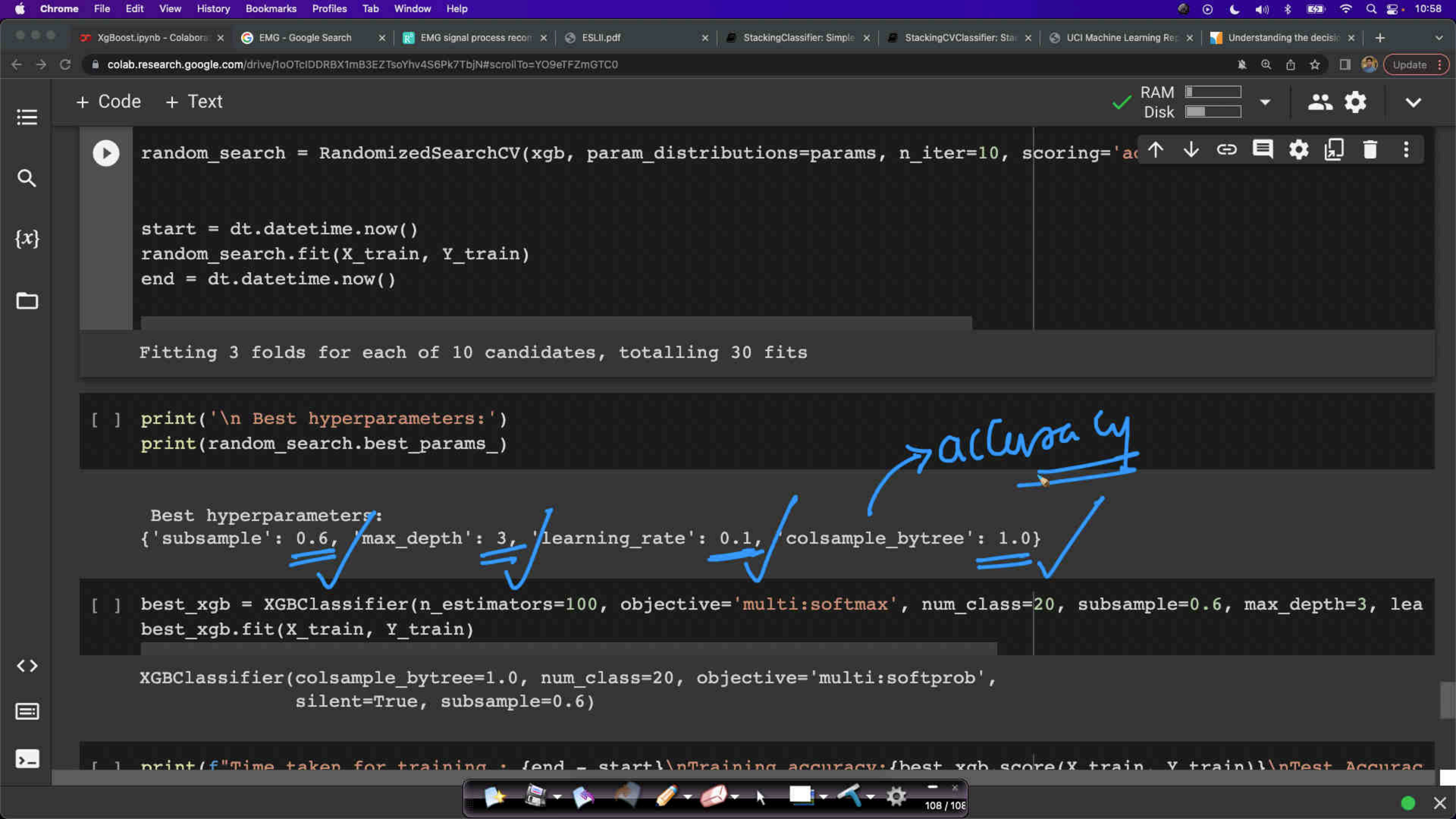
```
print('\n Best hyperparameters:')
print(random_search.best_params_)
```

```
        '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)
```

```
folds = 3

skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)

random_search = RandomizedSearchCV(xgb, param_distributions=params, n_iter=10, scoring='accuracy', n_jobs=4, cv=skf

start = dt.datetime.now()
random_search.fit(X_train, Y_train)
end = dt.datetime.now()
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

```
print('\n Best hyperparameters:')
print(random_search.best_params_)
```


Chrome File Edit View History Bookmarks Profiles Tab Window Help
10:58
XgBoost.ipynb - Colabora
EMG - Google Search
EMG signal process recon
ESLII.pdf
StackingClassifier: Simple
StackingCVClassifier: Sta
UCI Machine Learning Re
Understanding the decisio
colab.research.google.com/drive/1oOTcIDDRBX1mB3EZTsoYhv4S6Pk7TbjN#scrollTo=YO9eTFZmGTC0
Update
+ Code
+ Text
RAM
Disk
random_search = RandomizedSearchCV(xgb, param_distributions=params, n_iter=10, scoring='ac
start = dt.datetime.now()
random_search.fit(X_train, Y_train)
end = dt.datetime.now()
Fitting 3 folds for each of 10 candidates, totalling 30 fits
print('\n Best hyperparameters:')
print(random_search.best_params_)
accuracy
Best hyperparameters:
{'subsample': 0.6, 'max_depth': 3, 'learning_rate': 0.1, 'colsample_bytree': 1.0}
best_xgb = XGBClassifier(n_estimators=100, objective='multi:softmax', num_class=20, subsample=0.6, max_depth=3, lea
best_xgb.fit(X_train, Y_train)
XGBClassifier(colsample_bytree=1.0, num_class=20, objective='multi:softprob',
silent=True, subsample=0.6)
print(f"Time taken for training : {end - start}\nTraining accuracy:{best_xgb.score(X_train, Y_train)}\nTest Accurac
108 / 108

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

```
print('\n Best hyperparameters:')
print(random_search.best_params_)
```

```
 Best hyperparameters:
{'subsample': 0.6, 'max_depth': 3, 'learning_rate': 0.1, 'colsample_bytree': 1.0}
```

```
best_xgb = XGBClassifier(n_estimators=100, objective='multi:softmax', num_class=20, subsample=0.6, max_depth=3, lea
best_xgb.fit(X_train, Y_train)
```

```
XGBClassifier(colsample_bytree=1.0, num_class=20, objective='multi:softprob',
              silent=True, subsample=0.6)
```

```
print(f"Time taken for training : {end - start}\nTraining accuracy:{best_xgb.score(X_train, Y_train)}\nTest Accurac
```

```
Time taken for training : 0:05:07.294350
Training accuracy:0.6747209538305429
Test Accuracy: 0.6114633527770733
```

```
print(best_xgb.feature_importances_)

plt.bar(range(len(best_xgb.feature_importances_)), best_xgb.feature_importances_)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

```
print('\n Best hyperparameters:')
print(random_search.best_params_)
```

```
 Best hyperparameters:
{'subsample': 0.6, 'max_depth': 3, 'learning_rate': 0.1, 'colsample_bytree': 1.0}
```

```
best_xgb = XGBClassifier(n_estimators=100, objective='multi:softmax', num_class=20, subsample=0.6, max_depth=3, lea
best_xgb.fit(X_train, Y_train)
```

```
XGBClassifier(colsample_bytree=1.0, num_class=20, objective='multi:softprob',
              silent=True, subsample=0.6)
```

```
print(f"Time taken for training : {end - start}\nTraining accuracy:{best_xgb.score(X_train, Y_train)}\nTest Accurac
```

```
Time taken for training : 0:05:07.294350
Training accuracy:0.6747209538305429
Test Accuracy: 0.6114633527770733
```

```
print(best_xgb.feature_importances_)
```

```
print(random_search.best_params_)
```

```
 Best hyperparameters:
{'subsample': 0.6, 'max_depth': 3, 'learning_rate': 0.1, 'colsample_bytree': 1.0}
```

```
best_xgb = XGBClassifier(n_estimators=100, objective='multi:softmax', num_class=20, subsample=0.6, max_depth=3, lea
best_xgb.fit(X_train, Y_train)
```

```
XGBClassifier(colsample_bytree=1.0, num_class=20, objective='multi:softprob',
              silent=True, subsample=0.6)
```

```
print(f"Time taken for training : {end - start}\nTraining accuracy:{best_xgb.score(X_train, Y_train)}\nTest Accurac
```

```
Time taken for training : 0:05:07.294350
Training accuracy:0.6741209538305429
Test Accuracy: 0.6114633527770733
```

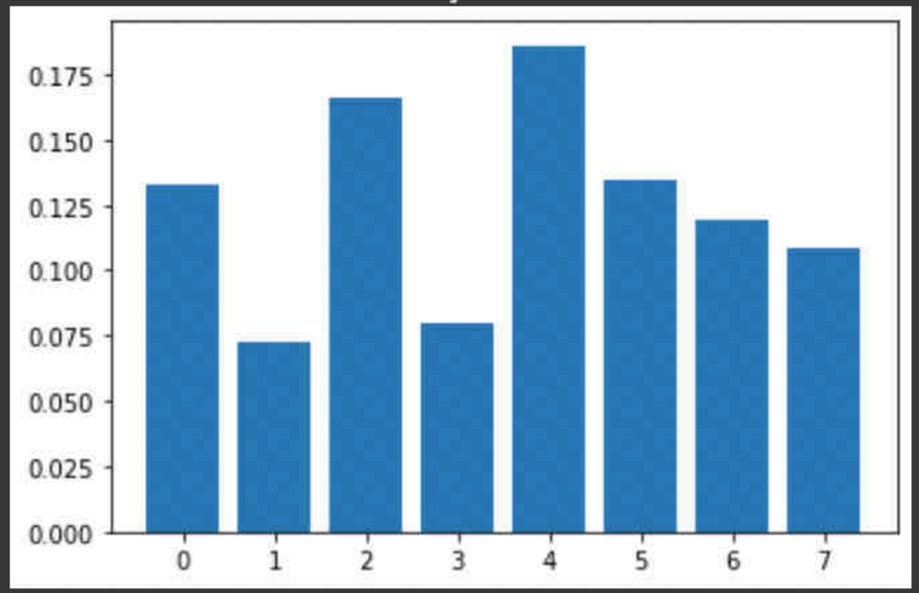
```
print(best_xgb.feature_importances_)

plt.bar(range(len(best_xgb.feature_importances_)), best_xgb.feature_importances_)
plt.show()
```

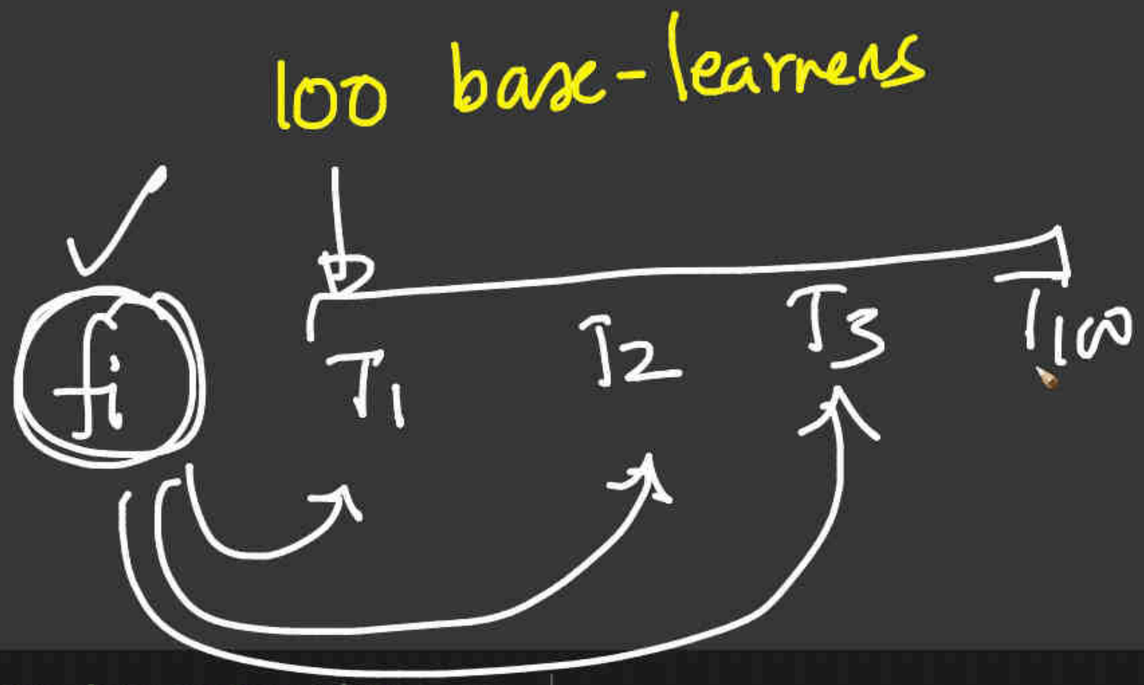
```
[0.13315983 0.07298602 0.16566639 0.07955533 0.18605755 0.13500534
 0.11930533 0.10826417]
```

```
Training accuracy:0.6747209538305429
Test Accuracy: 0.6114633527770733
```

```
[ ] print(best_xgb.feature_importances_)

    plt.bar(range(len(best_xgb.feature_importances_)), best_xgb.feature_importances_)
    plt.show()
```

```
[0.13315983 0.07298602 0.16566639 0.07955533 0.18605755 0.13500534
 0.11930533 0.10826417]
```



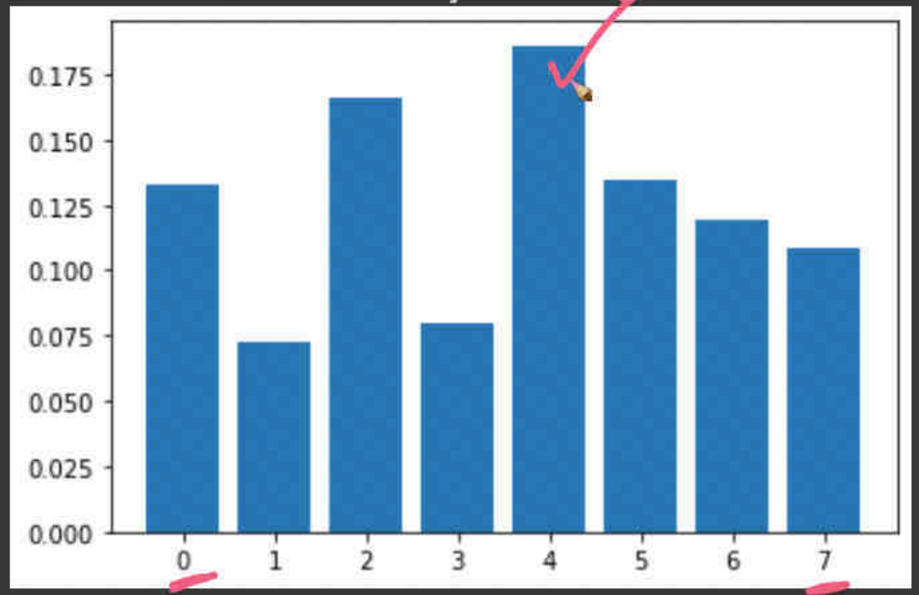



```
[ ] # Domain specific ideas to improve the results: Average data across time.
```

```
Training accuracy:0.6747209538305429
Test Accuracy: 0.6114633527770733
```

```
[ ]  print(best_xgb.feature_importances_)

     plt.bar(range(len(best_xgb.feature_importances_)), best_xgb.feature_importances_)
     plt.show()
```

```
[0.13315983 0.07298602 0.16566639 0.07955533 0.18605755 0.13500534
 0.11930533 0.10826417]
```

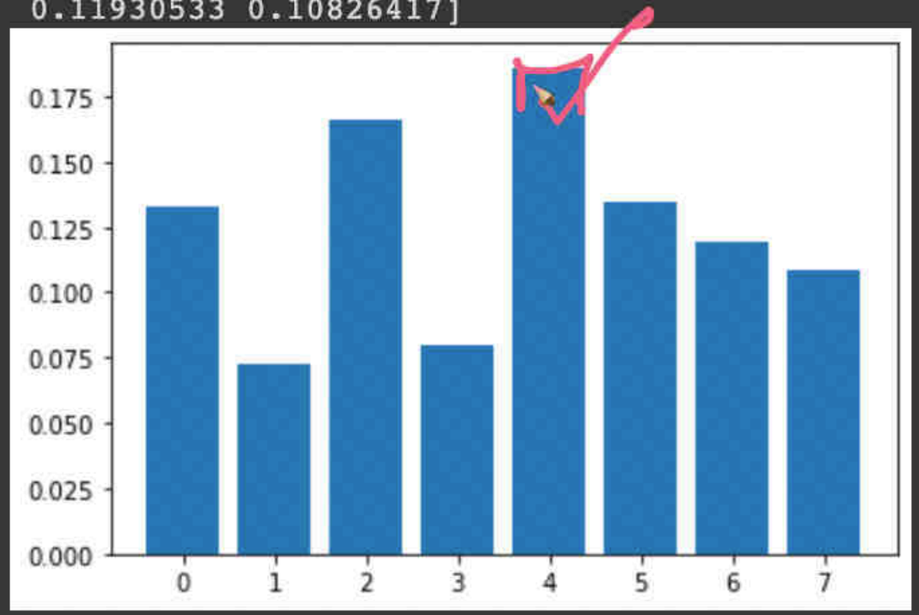



```
[ ]  # Domain specific ideas to improve the results: Average data across time.
```

Training accuracy:0.6747209538305429
Test Accuracy: 0.6114633527770733

```
print(best_xgb.feature_importances_)

plt.bar(range(len(best_xgb.feature_importances_)), best_xgb.feature_importances_)
plt.show()
```

```
[0.13315983 0.07298602 0.16566639 0.07955533 0.18605755 0.13500534
 0.11930533 0.10826417]
```




```
# Domain specific ideas to improve the results: Average data across time.
```

evals_result will contain the **eval_metrics** passed to the `fit()` function.

The returned evaluation result is a dictionary:

```
{'validation_0': {'logloss': ['0.604835', '0.531479']},
 'validation_1': {'logloss': ['0.41965', '0.17686']}}
```

Return type: evals_result

property **feature_importances_**: *numpy.ndarray*

Feature importances property, return depends on *importance_type* parameter.

Returns:

- **feature_importances_** (array of shape `[n_features]` except for multi-class)
- linear model, which returns an array with shape *(n_features, n_classes)*

property **feature_names_in_**: *numpy.ndarray*

Names of features seen during `fit()`. Defined only when *X* has feature names that are all strings.

fit(*X, y, *, sample_weight=None, base_margin=None, eval_set=None, eval_metric=None, early_stopping_rounds=None, verbose=True, xgb_model=None, sample_weight_eval_set=None, bas*

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).

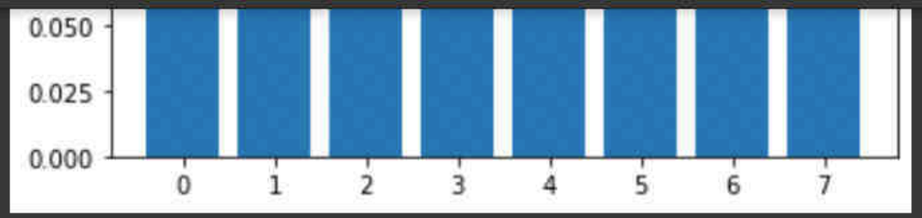
Table 10.2 summarizes the gradients for commonly used loss functions. For squared error loss, the negative gradient is just the ordinary residual $-g_{im} = y_i - f_{m-1}(x_i)$, so that (10.37) on its own is equivalent standard least squares boosting. With absolute error loss, the negative gradient is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

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

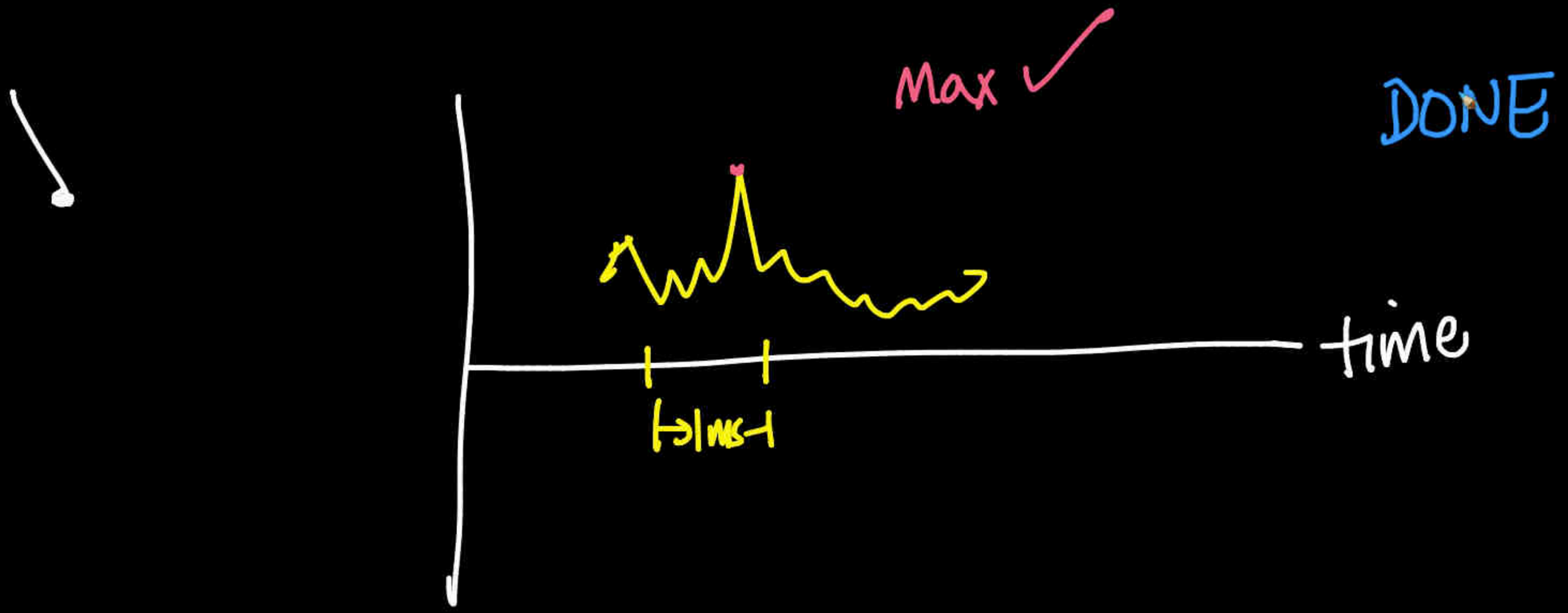
Setting	Loss Function	$-\partial L(y_i, f(x_i))/\partial f(x_i)$
Regression	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$
Regression	$\lvert y_i - f(x_i)\rvert$	$\text{sign}[y_i - f(x_i)]$
Regression	Huber	$y_i - f(x_i)$ for $\lvert y_i - f(x_i)\rvert \le \delta_m$ $\delta_m \text{sign}[y_i - f(x_i)]$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

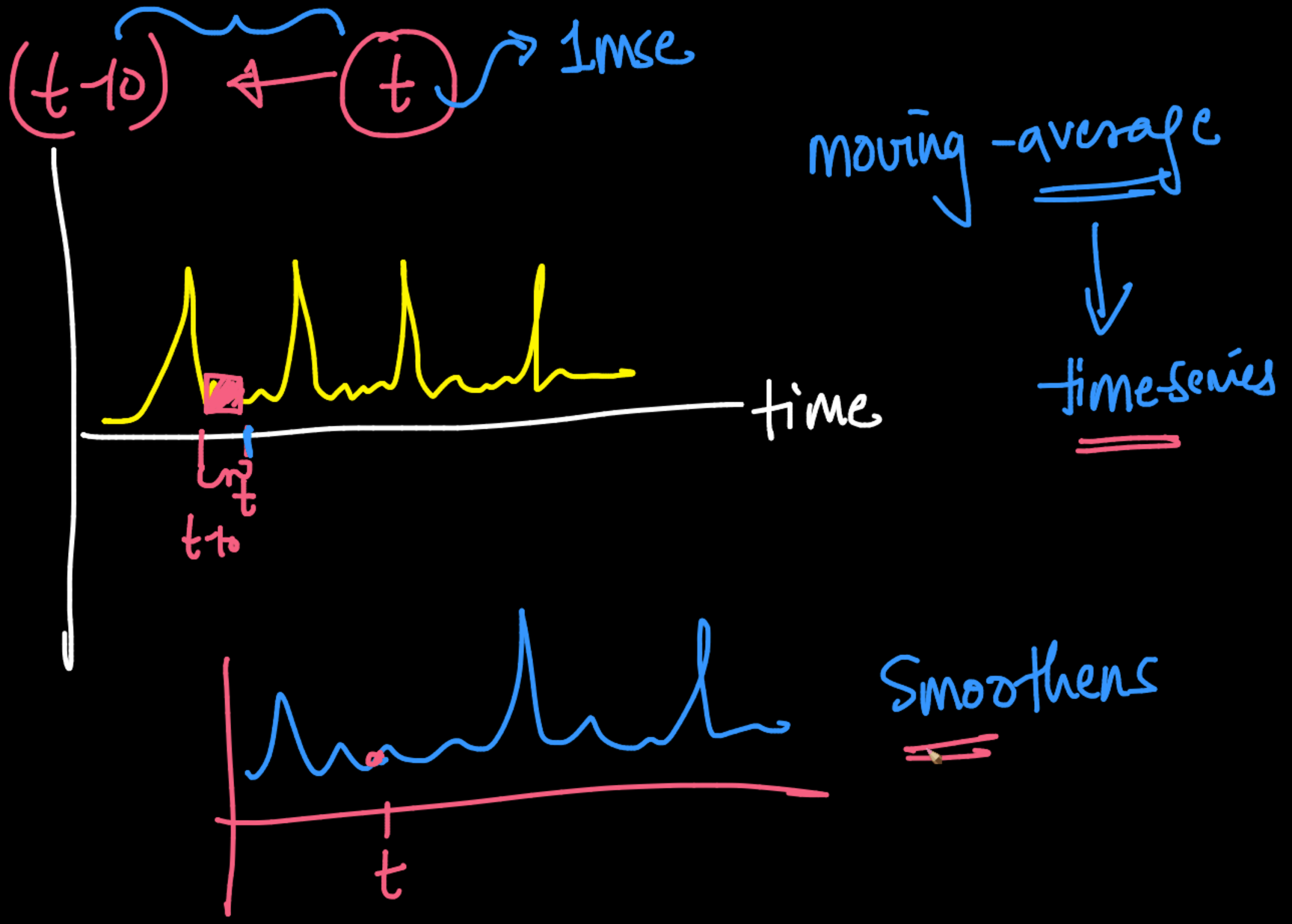
boosting procedure, and top-down decision tree induction, are themselves approximation procedures. After constructing the tree (10.37), the corresponding constants in each region are given by (10.30).

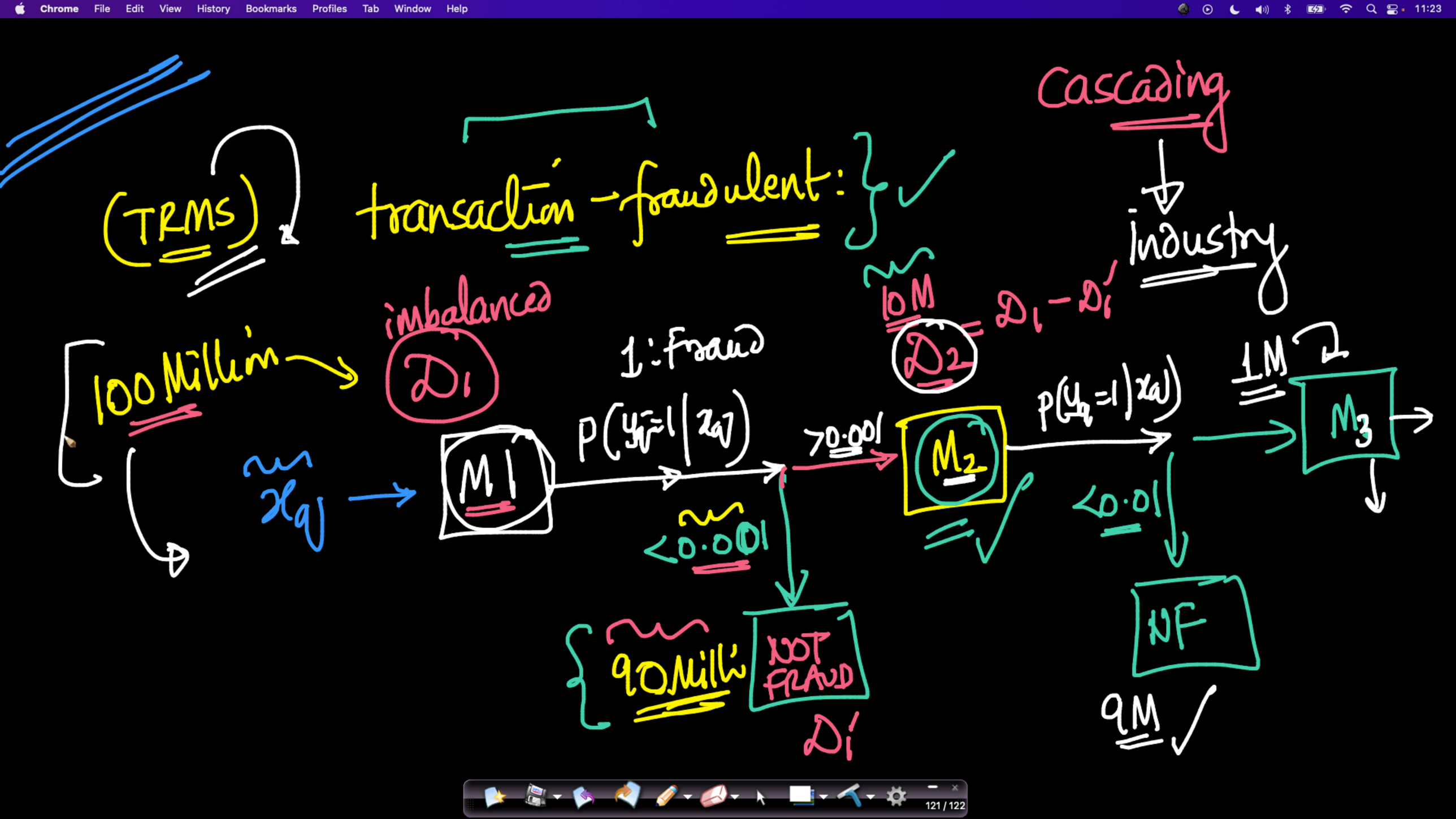
Table 10.2 summarizes the gradients for commonly used loss functions. For squared error loss, the negative gradient is just the ordinary residual $-g_{im} = y_i - f_{m-1}(x_i)$, so that (10.37) on its own is equivalent standard least squares boosting. With absolute error loss, the negative gradient is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

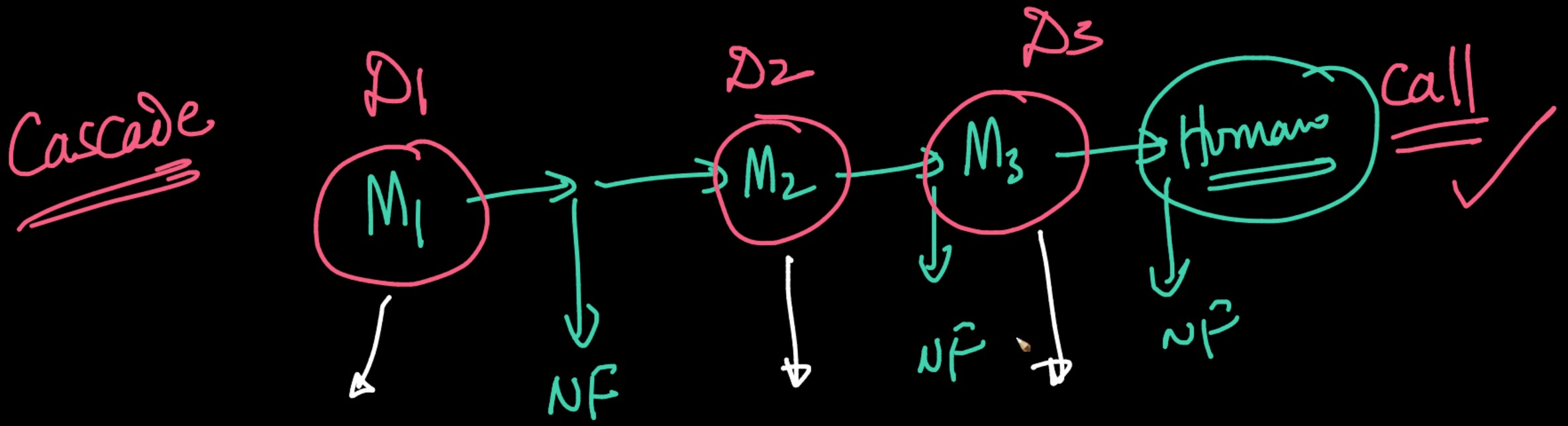



```
# Domain specific ideas to improve the results: Average data across time.
## Source: https://www.researchgate.net/figure/EMG-signal-process-recommended-Green-The-raw-signal-no-treatment-was
```

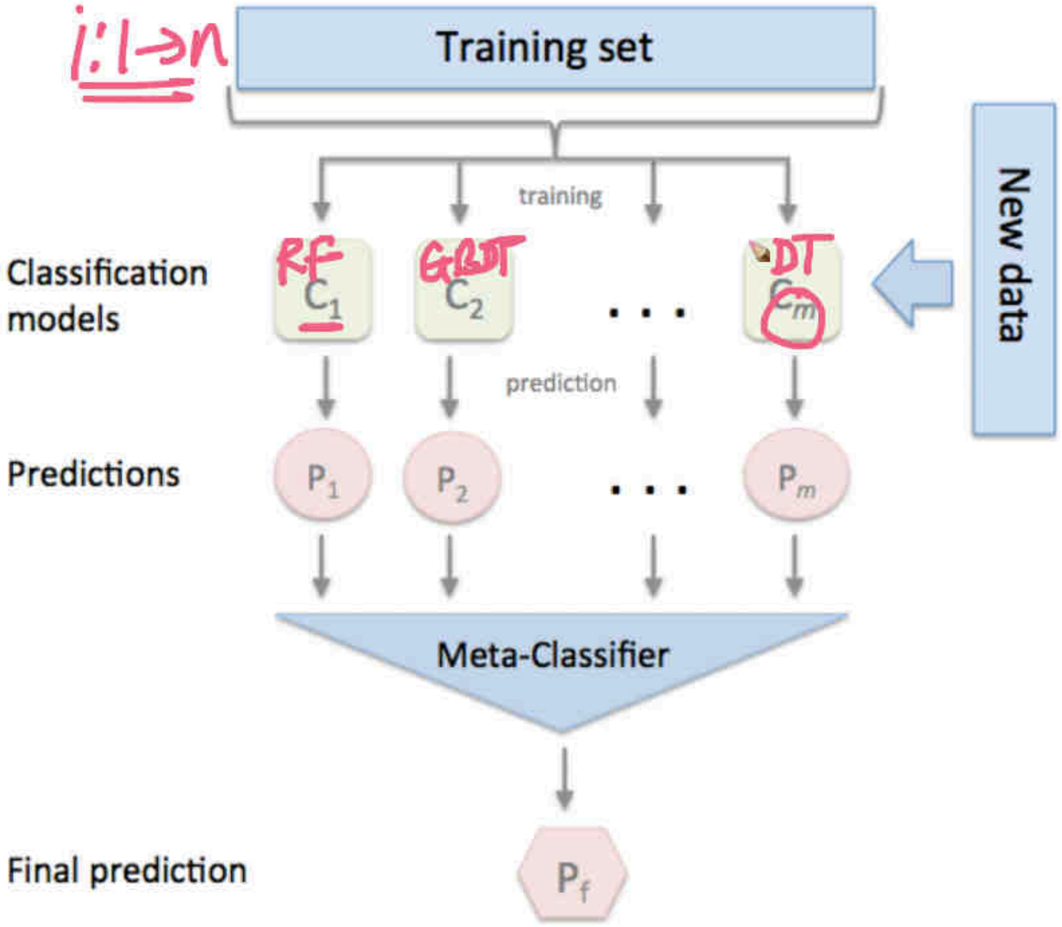





Cascading
industry
(TRMS)
transaction - fraudulent: { ✓
100 Millin
imbalanced
D_1
1: Fraud
x_q
M_1
$P(y_i=1|x_q)$
>0.001
<0.0001
90Milli
NOT FRAUD
D_1'
10M
$D_2 = D_1 - D_1'$
M_2
$P(y_i=1|x_q)$
<0.01
1M
M_3
NF
9M ✓

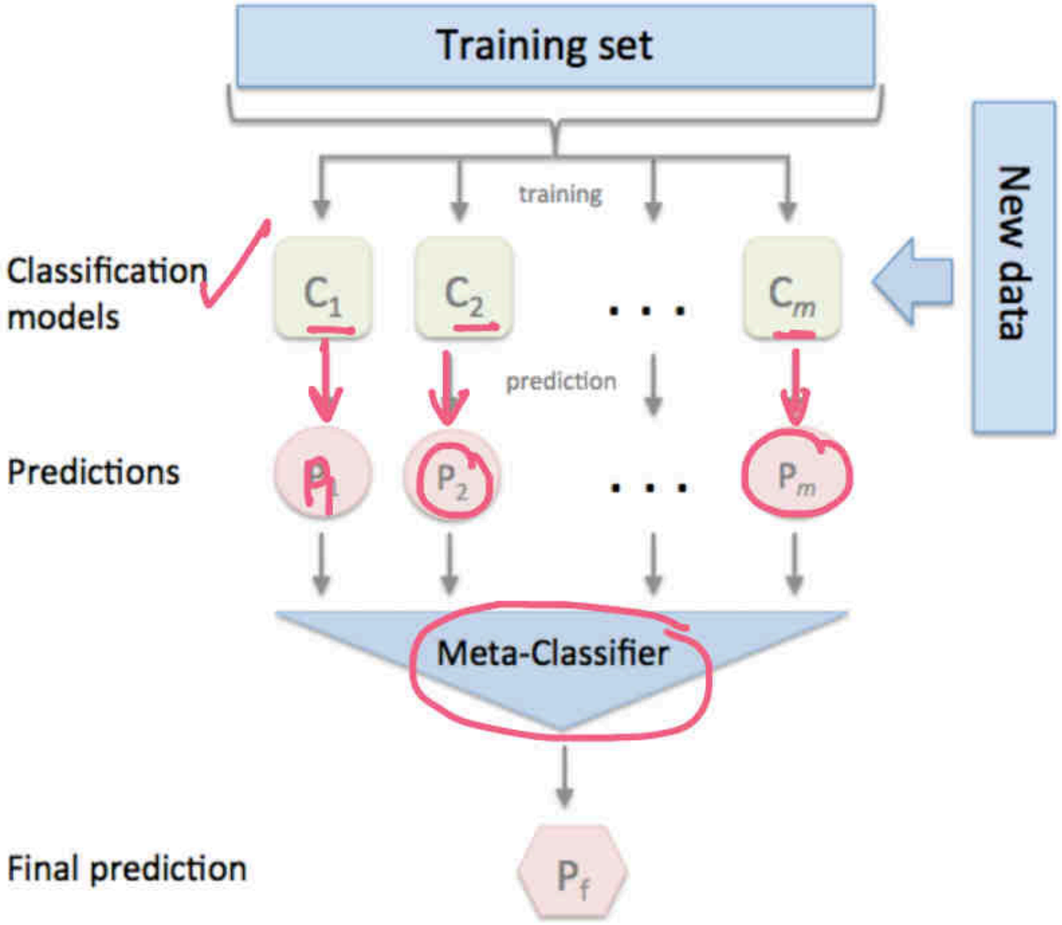


meta-classifier can either be trained on the predicted class labels or probabilities from the ensemble.



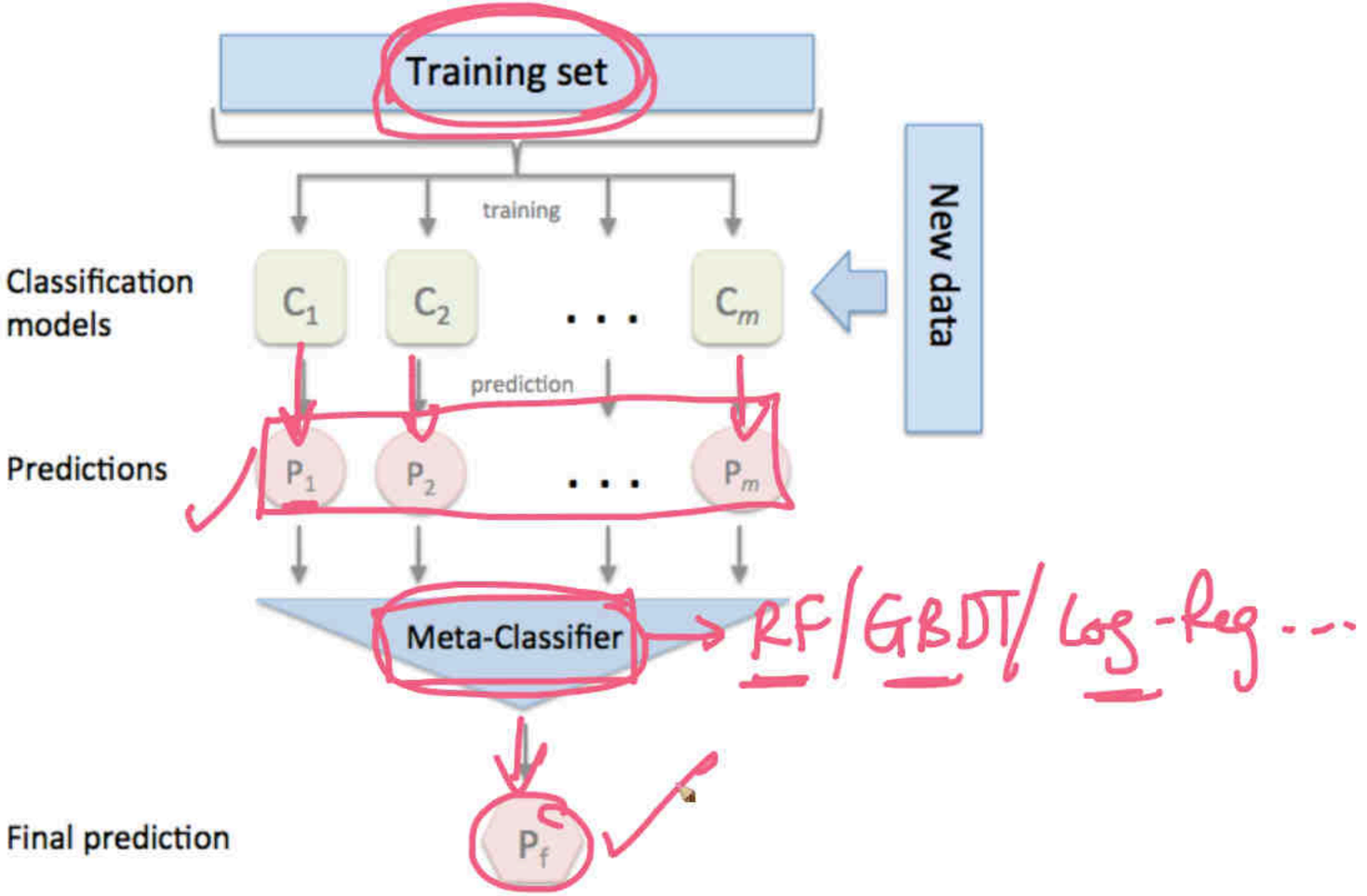
The algorithm can be summarized as follows (source: [1]):

meta-classifier can either be trained on the predicted class labels or probabilities from the ensemble.



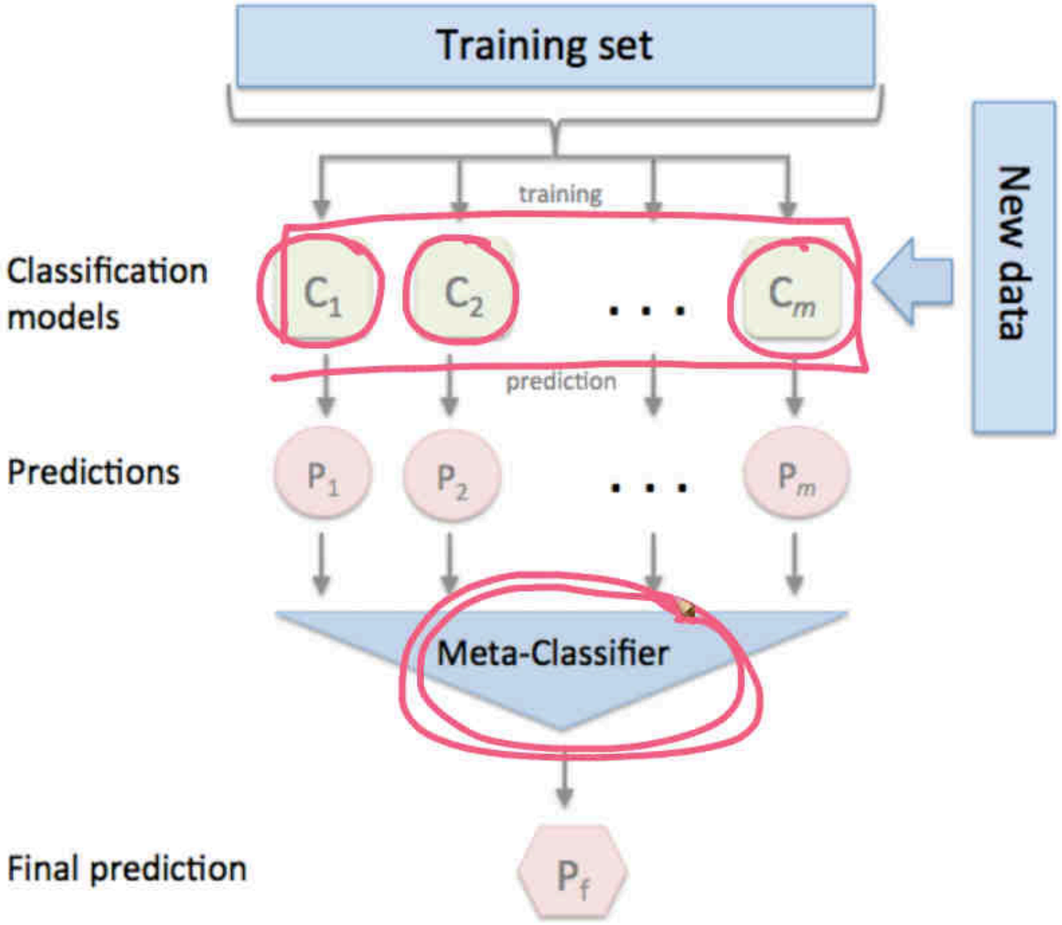
The algorithm can be summarized as follows (source: [1]):

meta-classifier can either be trained on the predicted class labels or probabilities from the ensemble.



The algorithm can be summarized as follows (source: [1]):

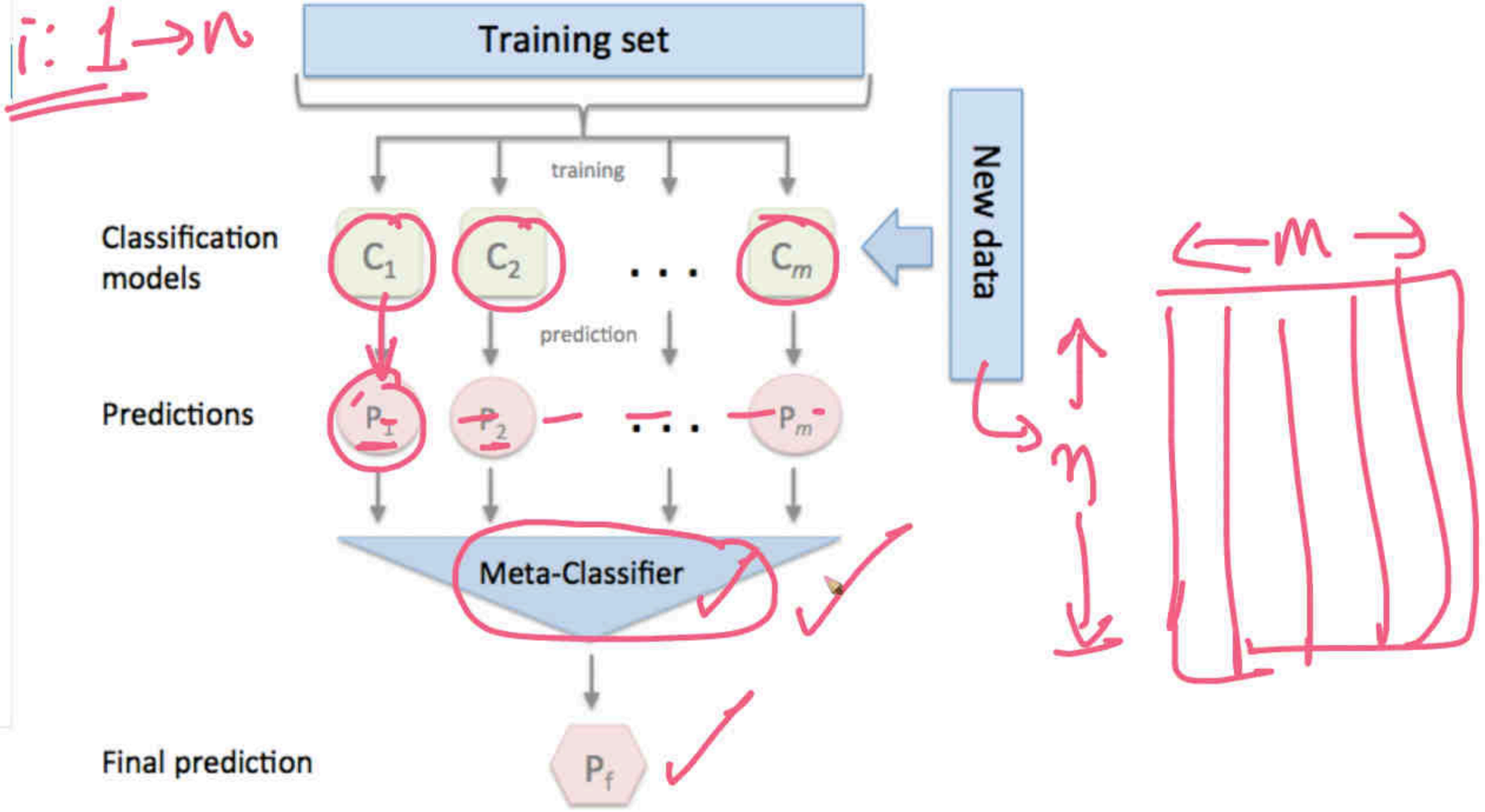
meta-classifier can either be trained on the predicted class labels or probabilities from the ensemble.



Baggup:
Var ↓ bias →

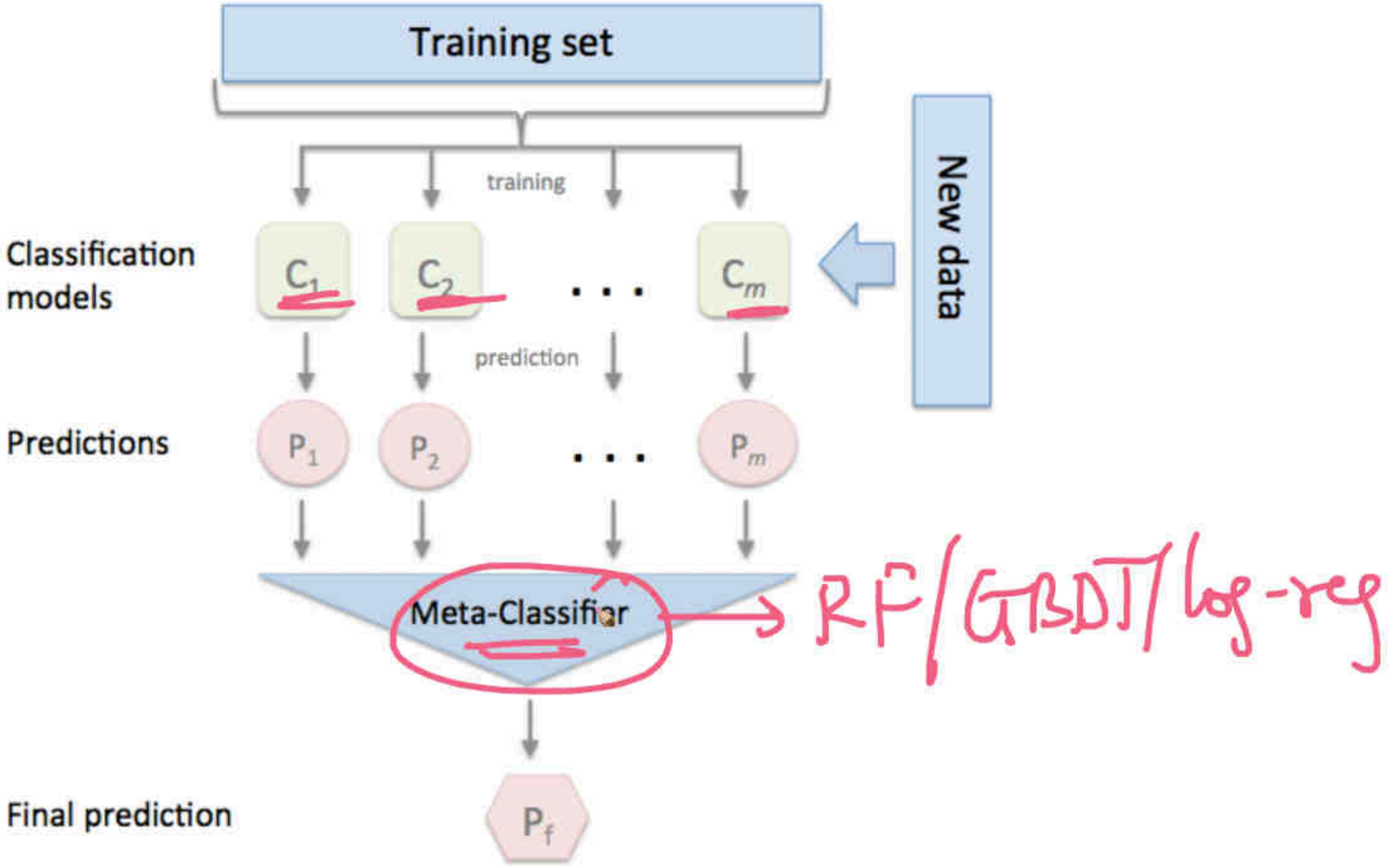
The algorithm can be summarized as follows (source: [1]):

meta-classifier can either be trained on the predicted class labels or probabilities from the ensemble.



The algorithm can be summarized as follows (source: [1]):

meta-classifier can either be trained on the predicted class labels or probabilities from the ensemble.



The algorithm can be summarized as follows (source: [1]):

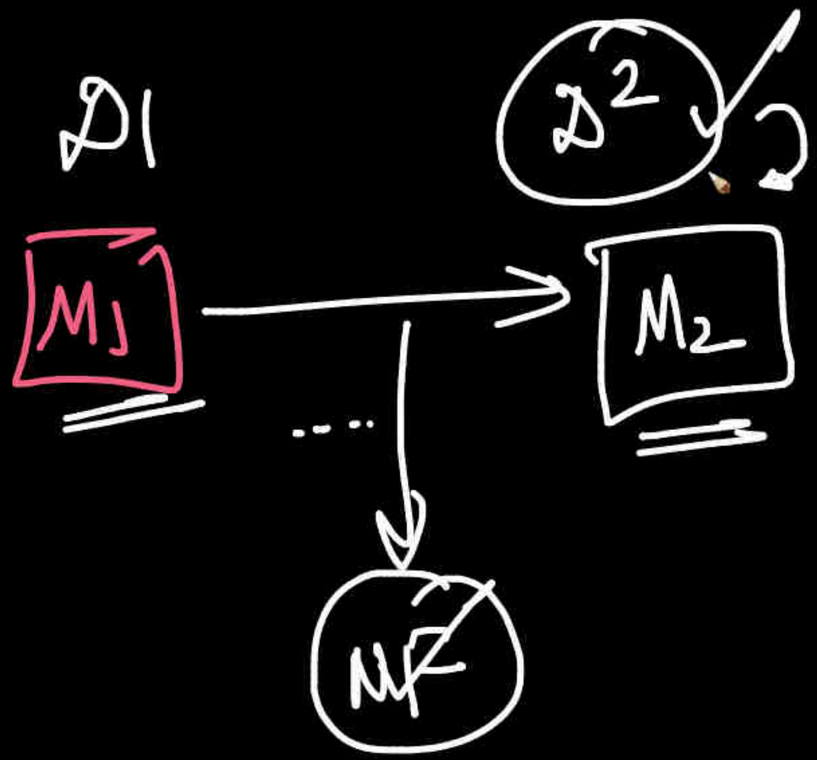
```
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from mlxtend.classifier import StackingClassifier
import numpy as np
import warnings

warnings.simplefilter('ignore')

clf1 = KNeighborsClassifier(n_neighbors=1)
clf2 = RandomForestClassifier(random_state=1)
clf3 = GaussianNB()
lr = LogisticRegression()
sclf = StackingClassifier(classifiers=[clf1, clf2, clf3],
                          meta_classifier=lr)

print('3-fold cross validation:\n')

for clf, label in zip([clf1, clf2, clf3, sclf],
                      ['KNN',
                       'Random Forest',
                       'Naive Bayes',
                       'StackingClassifier']):

    scores = model_selection.cross_val_score(clf, X, y,
                                             cv=3, scoring='accuracy')
    print("Accuracy: %0.2f (+/- %0.2f) [%s]"
```

```
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from mlxtend.classifier import StackingClassifier
import numpy as np
import warnings

warnings.simplefilter('ignore')

clf1 = KNeighborsClassifier(n_neighbors=1)
clf2 = RandomForestClassifier(random_state=1)
clf3 = GaussianNB()
lr = LogisticRegression()
sclf = StackingClassifier(classifiers=[clf1, clf2, clf3],
                          meta_classifier=lr)

print('3-fold cross validation:\n')

for clf, label in zip([clf1, clf2, clf3, sclf],
                      ['KNN',
                       'Random Forest',
                       'Naive Bayes',
                       'StackingClassifier']):

    scores = model_selection.cross_val_score(clf, X, y,
                                             cv=3, scoring='accuracy')
    print("Accuracy: %0.2f (+/- %0.2f) [%s]"
```

```
warnings.simplefilter('ignore')

clf1 = KNeighborsClassifier(n_neighbors=1)
clf2 = RandomForestClassifier(random_state=1)
clf3 = GaussianNB()
lr = LogisticRegression()
sclf = StackingClassifier(classifiers=[clf1, clf2, clf3],
                          meta_classifier=lr)

print('3-fold cross validation:\n')

for clf, label in zip([clf1, clf2, clf3, sclf],
                      ['KNN',
                       'Random Forest',
                       'Naive Bayes',
                       'StackingClassifier']):

    scores = model_selection.cross_val_score(clf, X, y,
                                             cv=3, scoring='accuracy')
    print("Accuracy: %0.2f (+/- %0.2f) [%s]"
          % (scores.mean(), scores.std(), label))
```

```
3-fold cross validation:

Accuracy: 0.91 (+/- 0.01) [KNN]
Accuracy: 0.95 (+/- 0.01) [Random Forest]
Accuracy: 0.91 (+/- 0.02) [Naive Bayes]
Accuracy: 0.95 (+/- 0.02) [StackingClassifier]
```

Cascading



Example 1 - Simple Stacked Classification

```
from sklearn import datasets

iris = datasets.load_iris()
X, y = iris.data[:, 1:3], iris.target
```

```
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from mlxtend.classifier import StackingClassifier
import numpy as np
import warnings

warnings.simplefilter('ignore')

clf1 = KNeighborsClassifier(n_neighbors=1)
clf2 = RandomForestClassifier(random_state=1)
clf3 = GaussianNB()
lr = LogisticRegression()
sclf = StackingClassifier(classifiers=[clf1, clf2, clf3],
```

Next-class: → minor details (recap)

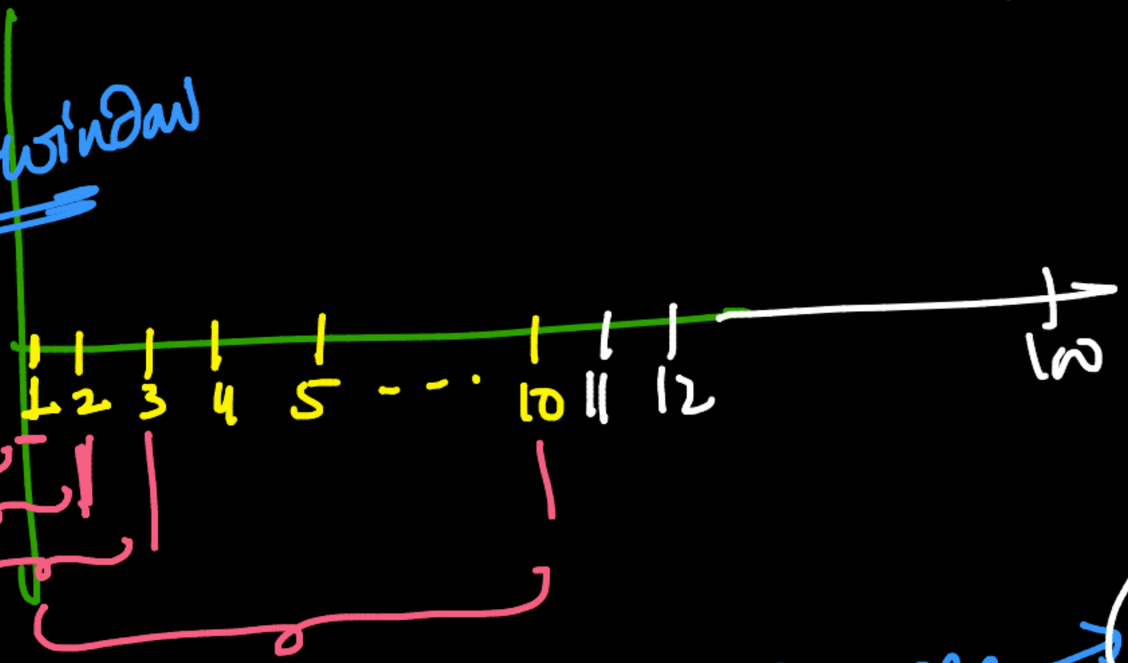
→ stacking, cascading

→ next-Model

Q&A

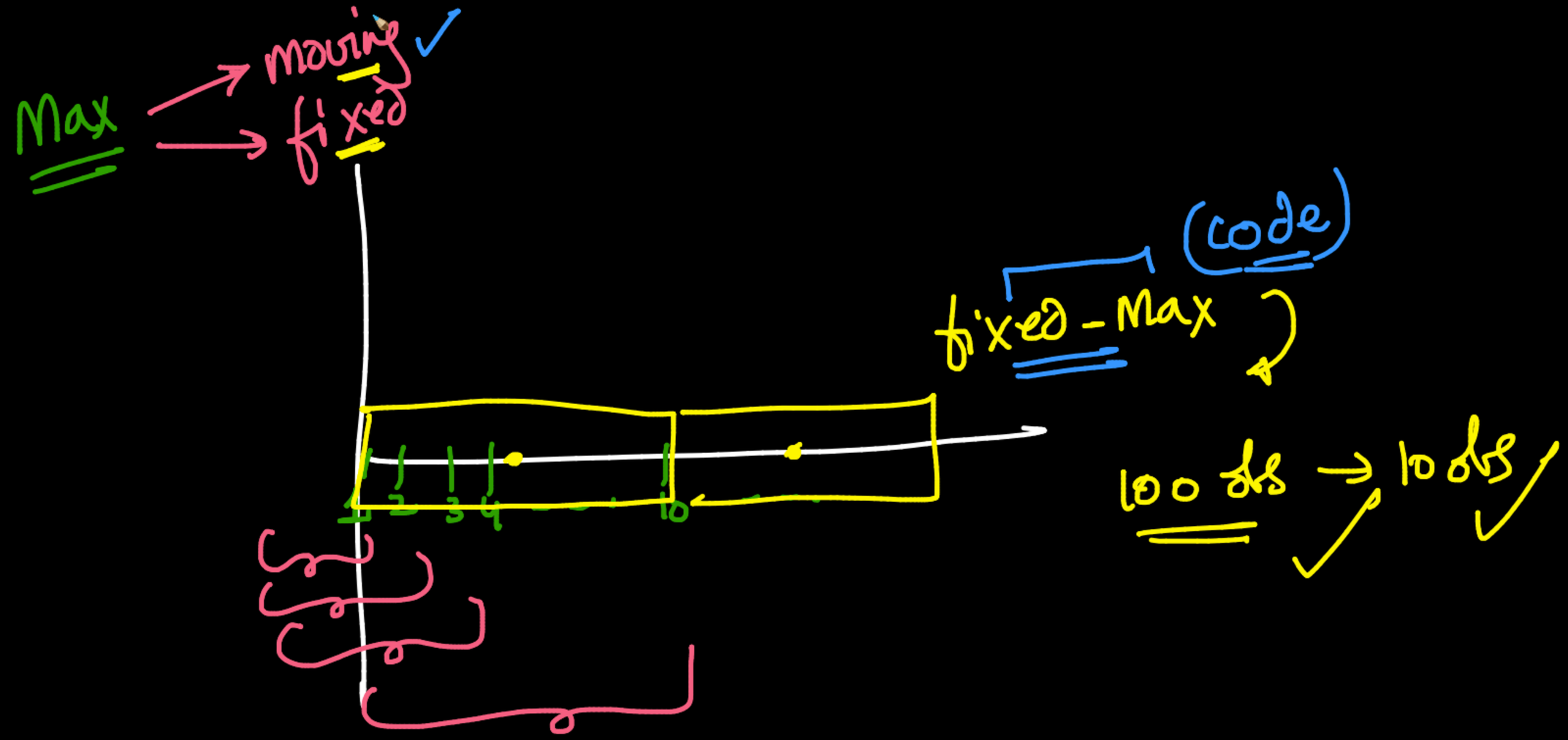
Moving average

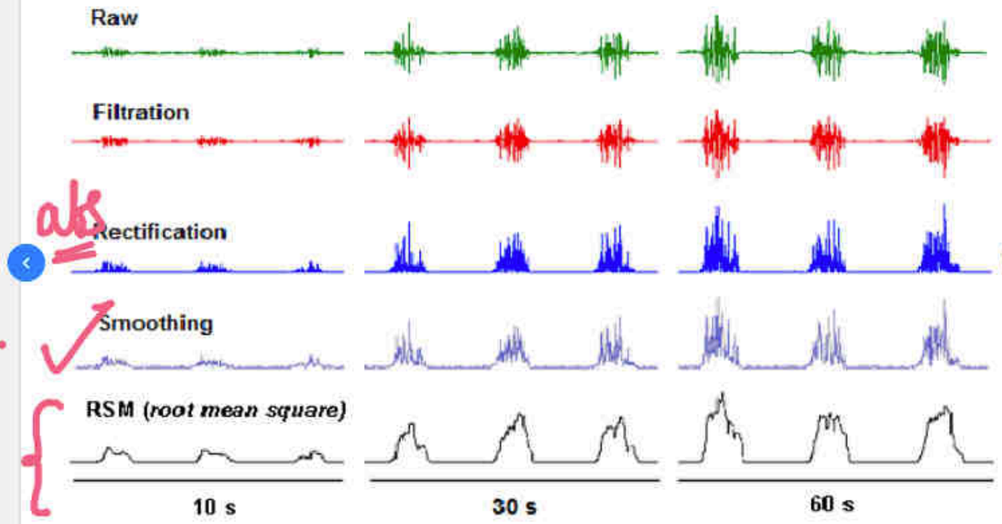
moving window



home-work

average → MW → FW


Max
moving
fixed
fixed - Max
(code)
1
2
3
4
10
100 ds → 10 ds



EMG signal process recommended. Green: The raw signal, no treatment was applied until this moment; Red: Filtrated signal, a limit was created for the signal, excluding everything out of it; Blue: Rectified signal, all negative values were transformed in positive ones and added; Purple: the smoothed signal, a linear enveloped was created and the extreme parts of the signal was excluded; Black: The RMS values after all the treatments.

$$L = y_i \cdot log\left(\frac{1}{1+e^{-\hat{y}_i}}\right) + (1-y_i) \cdot log\left(\frac{e^{-\hat{y}_i}}{1+e^{-\hat{y}_i}}\right)$$

First derivative obtained using Wolfram Alpha:

$$L' = \frac{y_i - (1-y_i) \cdot e^{\hat{y}_i}}{1+e^{\hat{y}_i}}$$

After multiplying by $\frac{e^{-\hat{y}_i}}{e^{-\hat{y}_i}}$:

$$L' = \frac{y_i \cdot e^{-\hat{y}_i} + y_i - 1}{1+e^{-\hat{y}_i}} = \frac{y_i \cdot (1+e^{-\hat{y}_i})}{1+e^{-\hat{y}_i}} - \frac{1}{1+e^{-\hat{y}_i}} = y_i - p_i$$

After changing sign we have expression for gradient of logistic loss function:

$$p_i - y_i$$

Share Cite Improve this answer Follow

answered Jun 17, 2016 at 16:38

Ogurtsov

534 1 3 13

4 What you're calling $\hat{y}$ here is *not* a prediction of y, but a linear combination of predictors. In generalized linear modeling we use the notation ν and call this term the "linear predictor". Your derivative of the loglikelihood (score) is wrong, there should be a squared term in the denominator, since bernoullis form an exponential likelihood. The score should be of the form $\frac{1}{p_i(1-p_i)}(y_i - p_i)$ – AdamO Jul 5, 2016 at 18:04

Add a comment

bicycles?

What should I call my mathematics thesis's 1st chapter that covers background concepts?

A weapon to kill angels

Circuit diagrams use unit prefix symbol as a decimal point

After rolling pizza dough, how do I move it to a pan without ruining it?

Can I use my own Stack Exchange answer in a book?

How do I check if my outlet's ground and neutral are connected correctly?

Is Ubuntu 22.04 stable?

Who's all the characters / games represented in Steam's Humble Games Publisher Birthday Sale 2022 banner / picture?

Tropical Operation Conversions: Multiple Operations

How to tell my manager's manager that I don't want to work under my manager anymore

Lots of dull colors!

Is there a way to do an "slice boolean"-type effect with just geometry nodes and/or material nodes?

Count the Liberties - Advanced

Why doesn't a nucleus-like body made up of just neutrons exist?

Are there any scriptures available (or mention) that are related to GyanVapi mandir?

Home

PUBLIC

Questions

Tags

Users

Unanswered

TEAMS

First, logistic loss is just negative log-likelihood, so we can start with expression for log-likelihood (p. 74 - this expression is log-likelihood itself, not negative log-likelihood):

$$L = y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i)$$

p_i is logistic function: $p_i = \frac{1}{1+e^{-\hat{y}_i}}$, where $\hat{y}_i$ is predicted values before logistic transformation (i.e., log-odds):

$$L = y_i \cdot log\left(\frac{1}{1+e^{-\hat{y}_i}}\right) + (1 - y_i) \cdot log\left(\frac{e^{-\hat{y}_i}}{1+e^{-\hat{y}_i}}\right)$$

First derivative obtained using Wolfram Alpha:

$$L' = \frac{y_i - (1 - y_i) \cdot e^{\hat{y}_i}}{1 + e^{\hat{y}_i}}$$

After multiplying by $\frac{e^{-\hat{y}_i}}{e^{-\hat{y}_i}}$:

$$L' = \frac{y_i \cdot e^{-\hat{y}_i} + y_i - 1}{1 + e^{-\hat{y}_i}} = \frac{y_i \cdot (1 + e^{-\hat{y}_i})}{1 + e^{-\hat{y}_i}} - \frac{1}{1 + e^{-\hat{y}_i}} = y_i - p_i$$

After changing sign we have expression for gradient of logistic loss function:

$$p_i - y_i$$

Share Cite Improve this answer Follow

answered Jun 17, 2016 at 16:38

Ogurtsov

still stable?

Can a balloon start from Earth and fly to the Moon, using Helium for lift to the top of the atmosphere and then as propellant?

Entering Spain with Soon-expiring US passport and expired Irish passport

Why do we say we "ride" and not "drive" bicycles?

What should I call my mathematics thesis's 1st chapter that covers background concepts?

A weapon to kill angels

Circuit diagrams use unit prefix symbol as a decimal point

After rolling pizza dough, how do I move it to a pan without ruining it?

Can I use my own Stack Exchange answer in a book?

How do I check if my outlet's ground and neutral are connected correctly?

Is Ubuntu 22.04 stable?

Who's all the characters / games represented in Steam's Humble Games Publisher Birthday Sale 2022 banner / picture?

Tropical Operation Conversions: Multiple Operations

How to tell my manager's manager that I don't want to work under my manager anymore

Lots of dull colors!

Is there a way to do an "slice boolean"-type

Home
PUBLIC
Questions
Tags
Users
Unanswered
TEAMS

First, logistic loss is just negative log-likelihood, so we can start with expression for log-likelihood (p. 74 - this expression is log-likelihood itself, not negative log-likelihood):

$$L = y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i)$$

p_i is logistic function: $p_i = \frac{1}{1+e^{-\hat{y}_i}}$, where $\hat{y}_i$ is predicted values before logistic transformation (i.e., log-odds):

$$L = y_i \cdot log\left(\frac{1}{1+e^{-\hat{y}_i}}\right) + (1 - y_i) \cdot log\left(\frac{e^{-\hat{y}_i}}{1+e^{-\hat{y}_i}}\right)$$

First derivative obtained using Wolfram Alpha:

$$L' = \frac{y_i - (1 - y_i) \cdot e^{\hat{y}_i}}{1 + e^{\hat{y}_i}}$$

After multiplying by $\frac{e^{-\hat{y}_i}}{e^{-\hat{y}_i}}$:

$$L' = \frac{y_i \cdot e^{-\hat{y}_i} + y_i - 1}{1 + e^{-\hat{y}_i}} = \frac{y_i \cdot (1 + e^{-\hat{y}_i})}{1 + e^{-\hat{y}_i}} - \frac{1}{1 + e^{-\hat{y}_i}} = y_i - p_i$$

After changing sign we have expression for gradient of logistic loss function:

$$p_i - y_i$$

Share Cite Improve this answer Follow

answered Jun 17, 2016 at 16:38
Ogurtsov

still stable?
Can a balloon start from Earth and fly to the Moon, using Helium for lift to the top of the atmosphere and then as propellant?
Entering Spain with Soon-expiring US passport and expired Irish passport
Why do we say we "ride" and not "drive" bicycles?
What should I call my mathematics thesis's 1st chapter that covers background concepts?
A weapon to kill angels
Circuit diagrams use unit prefix symbol as a decimal point
After rolling pizza dough, how do I move it to a pan without ruining it?
Can I use my own Stack Exchange answer in a book?
How do I check if my outlet's ground and neutral are connected correctly?
Is Ubuntu 22.04 stable?
Who's all the characters / games represented in Steam's Humble Games Publisher Birthday Sale 2022 banner / picture?
Tropical Operation Conversions: Multiple Operations
How to tell my manager's manager that I don't want to work under my manager anymore
Lots of dull colors!
Is there a way to do an "slice boolean"-type

$$-\frac{\partial L}{\partial F(x)} =$$

$$\log\text{-loss}(y_i, \hat{y_i})$$

$p_i = \hat{y_i}$

y_i

$$f_K(x) = p_i = \hat{y_i}$$

p_i is logistic function: $p_i = \frac{1}{1+e^{-\hat{y}_i}}$, where $\hat{y}_i$ is predicted values before logistic transformation (i.e., log-odds):

$$L = y_i \cdot log\left(\frac{1}{1+e^{-\hat{y}_i}}\right) + (1-y_i) \cdot log\left(\frac{e^{-\hat{y}_i}}{1+e^{-\hat{y}_i}}\right)$$

First derivative obtained using Wolfram Alpha:

$$L' = \frac{y_i - (1-y_i) \cdot e^{\hat{y}_i}}{1+e^{\hat{y}_i}}$$

After multiplying by $\frac{e^{-\hat{y}_i}}{e^{-\hat{y}_i}}$:

$$L' = \frac{y_i \cdot e^{-\hat{y}_i} + y_i - 1}{1+e^{-\hat{y}_i}} = \frac{y_i \cdot (1+e^{-\hat{y}_i})}{1+e^{-\hat{y}_i}} - \frac{1}{1+e^{-\hat{y}_i}} = y_i - p_i$$

After changing sign we have expression for gradient of logistic loss function:

$$p_i - y_i$$

Share Cite Improve this answer Follow

answered Jun 17, 2016 at 16:38
Ogurtsov
534 1 3 13

4 What you're calling $\hat{y}$ here is *not* a prediction of y, but a linear combination of predictors. In generalized linear modeling we use the notation v and call this term the "linear predictor". Your derivative of the loglikelihood (score) is ... is form an exponential likelihood ... at

Entering Spain with Soon-expiring US passport and expired Irish passport
Why do we say we "ride" and not "drive" bicycles?
What should I call my mathematics thesis's 1st chapter that covers background concepts?
A weapon to kill angels
Circuit diagrams use unit prefix symbol as a decimal point
After rolling pizza dough, how do I move it to a pan without ruining it?
Can I use my own Stack Exchange answer in a book?
How do I check if my outlet's ground and neutral are connected correctly?
Is Ubuntu 22.04 stable?
Who's all the characters / games represented in Steam's Humble Games Publisher Birthday Sale 2022 banner / picture?
Tropical Operation Conversions: Multiple Operations
How to tell my manager's manager that I don't want to work under my manager anymore
Lots of dull colors!
Is there a way to do an "slice boolean"-type effect with just geometry nodes and/or material nodes?
Count the Liberties - Advanced
Why doesn't a nucleus-like body made up of

Home
PUBLIC
Questions
Tags
Users
Unanswered
TEAMS

p_i is logistic function: $p_i = \frac{1}{1+e^{-\hat{y}_i}}$, where $\hat{y}_i$ is predicted values before logistic transformation (i.e., log-odds):

$$L = y_i \cdot log\left(\frac{1}{1+e^{-\hat{y}_i}}\right) + (1-y_i) \cdot log\left(\frac{e^{-\hat{y}_i}}{1+e^{-\hat{y}_i}}\right)$$

First derivative obtained using Wolfram Alpha:

$$L' = \frac{y_i - (1-y_i) \cdot e^{\hat{y}_i}}{1+e^{\hat{y}_i}}$$

After multiplying by $\frac{e^{-\hat{y}_i}}{e^{-\hat{y}_i}}$:

$$L' = \frac{y_i \cdot e^{-\hat{y}_i} + y_i - 1}{1+e^{-\hat{y}_i}} = \frac{y_i \cdot (1+e^{-\hat{y}_i})}{1+e^{-\hat{y}_i}} - \frac{1}{1+e^{-\hat{y}_i}} = y_i - p_i$$

After changing sign we have expression for gradient of logistic loss function:

$$p_i - y_i$$

Share Cite Improve this answer Follow

answered Jun 17, 2016 at 16:38
Ogurtsov
534 1 3 13

4 What you're calling $\hat{y}$ here is *not* a prediction of y, but a linear combination of predictors. In generalized linear modeling we use the notation ν and call this term the "linear predictor". Your derivative of the loglikelihood (score) is [illegible] is form an exponential likelihood [illegible] at

Entering Spain with Soon-expiring US passport and expired Irish passport
Why do we say we "ride" and not "drive" bicycles?
What should I call my mathematics thesis's 1st chapter that covers background concepts?
A weapon to kill angels
Circuit diagrams use unit prefix symbol as a decimal point
After rolling pizza dough, how do I move it to a pan without ruining it?
Can I use my own Stack Exchange answer in a book?
How do I check if my outlet's ground and neutral are connected correctly?
Is Ubuntu 22.04 stable?
Who's all the characters / games represented in Steam's Humble Games Publisher Birthday Sale 2022 banner / picture?
Tropical Operation Conversions: Multiple Operations
How to tell my manager's manager that I don't want to work under my manager anymore
Lots of dull colors!
Is there a way to do an "slice boolean"-type effect with just geometry nodes and/or material nodes?
Count the Liberties - Advanced
Why doesn't a nucleus-like body made up of

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 negative gradient is a compromise between these two (see the table).

For classification the loss function is the multinomial deviance (10.22), and K least squares trees are constructed at each iteration. Each tree T_{km} is fit to its respective negative gradient vector $\mathbf{g}_{km}$,

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 negative gradient is a compromise between these two (see the table).

For classification the loss function is the multinomial deviance (10.22), and K least squares trees are constructed at each iteration. Each tree T_{km} is fit to its respective negative gradient vector $\mathbf{g}_{km}$,

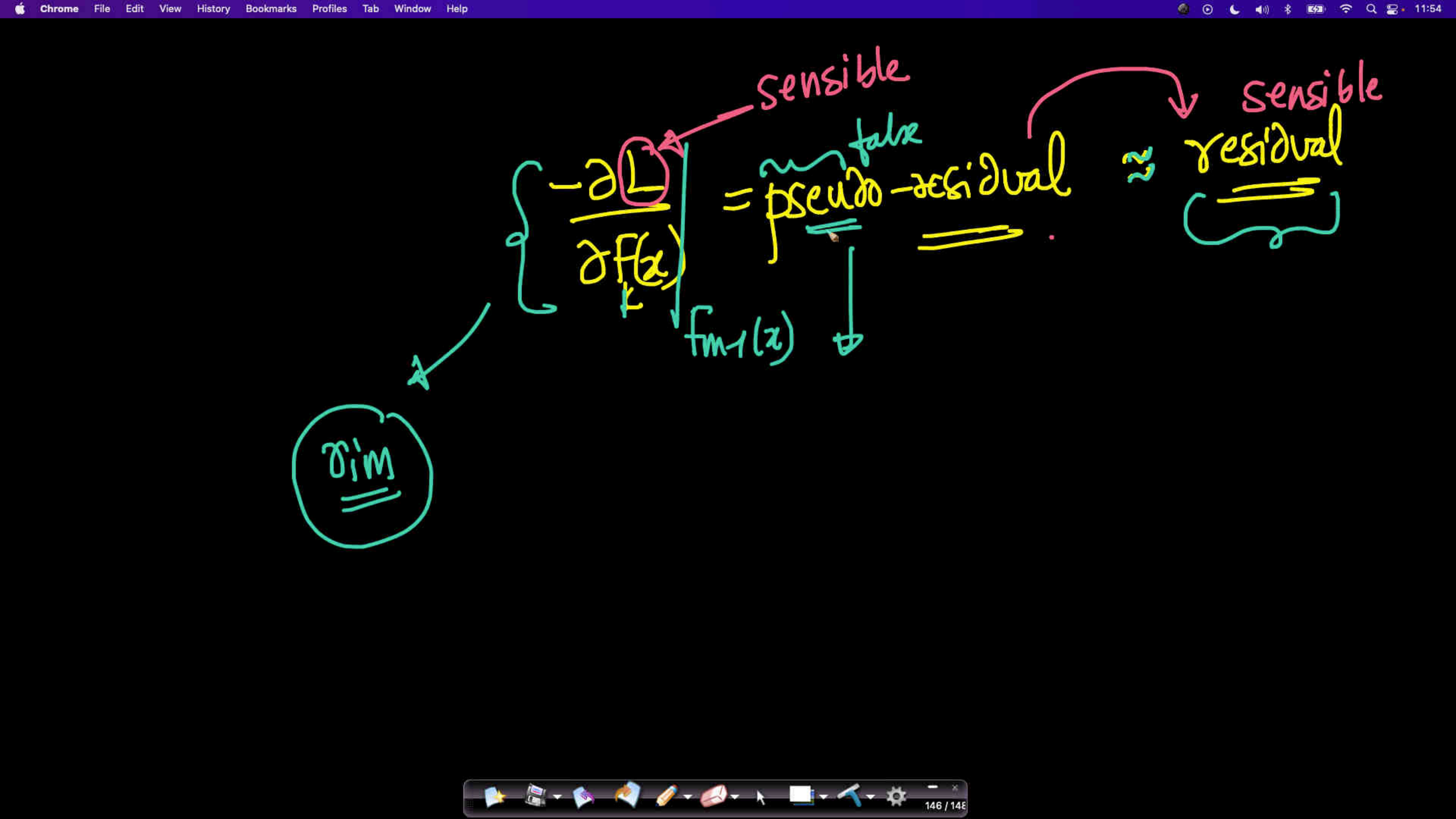

sensible
false
sensible
residual
pseudo-residual
fm-1(x)
xim

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 negative gradient is a compromise between these two (see the table).

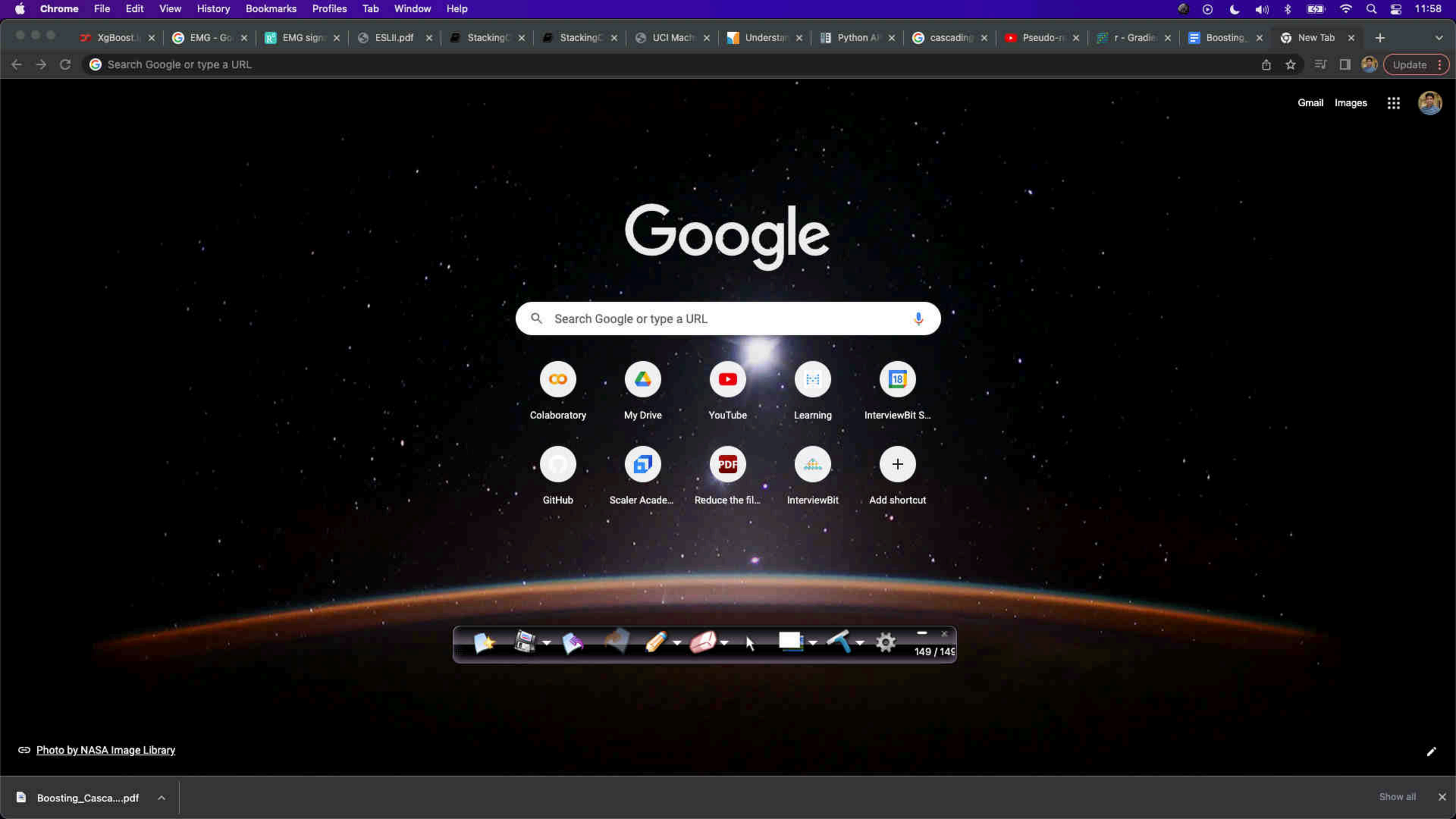
For classification the loss function is the multinomial deviance (10.22), and K least squares trees are constructed at each iteration. Each tree T_{km} is fit to its respective negative gradient vector $\mathbf{g}_{km}$,

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 square[illegible] ordinary residual


Chrome File Edit View History Bookmarks Profiles Tab Window Help
11:58
New Tab
Search Google or type a URL
Update
Gmail
Images
Google
Search Google or type a URL
Colaboratory
My Drive
YouTube
Learning
InterviewBit S...
GitHub
Scaler Acade...
Reduce the fil...
InterviewBit
Add shortcut
149 / 149
Photo by NASA Image Library
Boosting_Casca....pdf
Show all