Topics

GBDT

- ~~Code~~ for DecisionTrees ✓
- code for Random Forest ✓

HPT

- Boosting
  - Intuition ✓
  - Gradient Boosting
  - Bias-variance tradeoff.

[Adaboost]

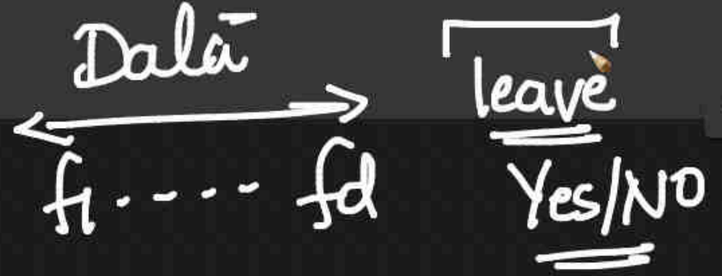
- Math dive-deep (Post-read videos)

Business Problem + Data Proprocessing


```
# HR/People Analytics
import numpy as np
import pandas as pd
import io
```

```
[ ] !wget "https://drive.google.com/uc?export=download&id=16KtxSt_QEGQvfluEaMls5cCHPwhRXgCk" -O HR-Employee-Attrition.c

--2022-05-13 13:24:44--  https://drive.google.com/uc?export=download&id=16KtxSt_QEGQvfluEaMls5cCHPwhRXgCk
Resolving drive.google.com (drive.google.com)... 142.250.141.113, 142.250.141.100, 142.250.141.139, ...
Connecting to drive.google.com (drive.google.com)|142.250.141.113|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://doc-08-64-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc717deffksulhg5h7mbp1/157t46bjk116kh
Warning: wildcards not supported in HTTP.
--2022-05-13 13:24:44--  https://doc-08-64-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc717deffksulhg5h7mbp1
Resolving doc-08-64-docs.googleusercontent.com (doc-08-64-docs.googleusercontent.com)... 142.250.141.132, 2607:f8b0:
Connecting to doc-08-64-docs.googleusercontent.com (doc-08-64-docs.googleusercontent.com)|142.250.141.132|:443... co
HTTP request sent, awaiting response... 200 OK
Length: 227977 (223K) [text/csv]
Saving to: 'HR-Employee-Attrition.csv'

HR-Employee-Attriti 100%[===================>] 222.63K  --.-KB/s    in 0.009s
```

Business Problem + Data Proprocessing




```
# HR/People Analytics
import numpy as np
import pandas as pd
import io
```

```
[ ] !wget "https://drive.google.com/uc?export=download&id=16KtxSt_QEGQvfluEaMls5cCHPwhRXgCk" -O HR-Employee-Attrition.c
```

```
--2022-05-13 13:24:44--  https://drive.google.com/uc?export=download&id=16KtxSt_QEGQvfluEaMls5cCHPwhRXgCk
Resolving drive.google.com (drive.google.com)... 142.250.141.113, 142.250.141.100, 142.250.141.139, ...
Connecting to drive.google.com (drive.google.com)|142.250.141.113|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://doc-08-64-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc717deffksulhg5h7mbp1/157t46bjk116kh
Warning: wildcards not supported in HTTP.
--2022-05-13 13:24:44--  https://doc-08-64-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc717deffksulhg5h7mbp1
Resolving doc-08-64-docs.googleusercontent.com (doc-08-64-docs.googleusercontent.com)... 142.250.141.132, 2607:f8b0:
Connecting to doc-08-64-docs.googleusercontent.com (doc-08-64-docs.googleusercontent.com)|142.250.141.132|:443... co
HTTP request sent, awaiting response... 200 OK
Length: 227977 (223K) [text/csv]
Saving to: 'HR-Employee-Attrition.csv'

HR-Employee-Attriti 100%[===================>] 222.63K  --.-KB/s    in 0.009s
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help 9:07

DT and RF.ipynb - Colaborator | sklearn.tree.DecisionTreeClass | sklearn.ensemble.RandomFore

colab.research.google.com/drive/1d2HJ9AKulf83i7pwmDrL-2FOCncFubRF#scrollTo=DymQNKTQODL8

Update

+ Code + Text Reconnect

Business Problem + Data Proprocessing

```
# HR/People Analytics
import numpy as np
import pandas as pd
import io
```

```
[ ] !wget "https://drive.google.com/uc?export=download&id=16KtxSt_QEGQvfluEaMls5cCHPwhRXgCk" -O HR-Employee-Attrition.c
```

```
--2022-05-13 13:24:44--  https://drive.google.com/uc?export=download&id=16KtxSt_QEGQvfluEaMls5cCHPwhRXgCk
Resolving drive.google.com (drive.google.com)... 142.250.141.113, 142.250.141.100, 142.250.141.139, ...
Connecting to drive.google.com (drive.google.com)|142.250.141.113|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://doc-08-64-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc717deffksulhg5h7mbp1/157t46bjk116kh
Warning: wildcards not supported in HTTP.
--2022-05-13 13:24:44--  https://doc-08-64-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc717deffksulhg5h7mbp1
Resolving doc-08-64-docs.googleusercontent.com (doc-08-64-docs.googleusercontent.com)... 142.250.141.132, 2607:f8b0:
Connecting to doc-08-64-docs.googleusercontent.com (doc-08-64-docs.googleusercontent.com)|142.250.141.132|:443... co
HTTP request sent, awaiting response... 200 OK
Length: 227977 (223K) [text/csv]
Saving to: 'HR-Employee-Attrition.csv'

HR-Employee-Attriti 100%[===================>] 222.63K  --.-KB/s    in 0.009s
```

4 / 4

```
HR-Employee-Attriti 100%[===================>] 222.63K  --.-KB/s    in 0.009s

[105]
2022-05-13 13:24:44 (24.7 MB/s) - 'HR-Employee-Attrition.csv' saved [227977/227977]
```

```
df = pd.read_csv('HR-Employee-Attrition.csv')
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	E
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	

5 rows × 35 columns

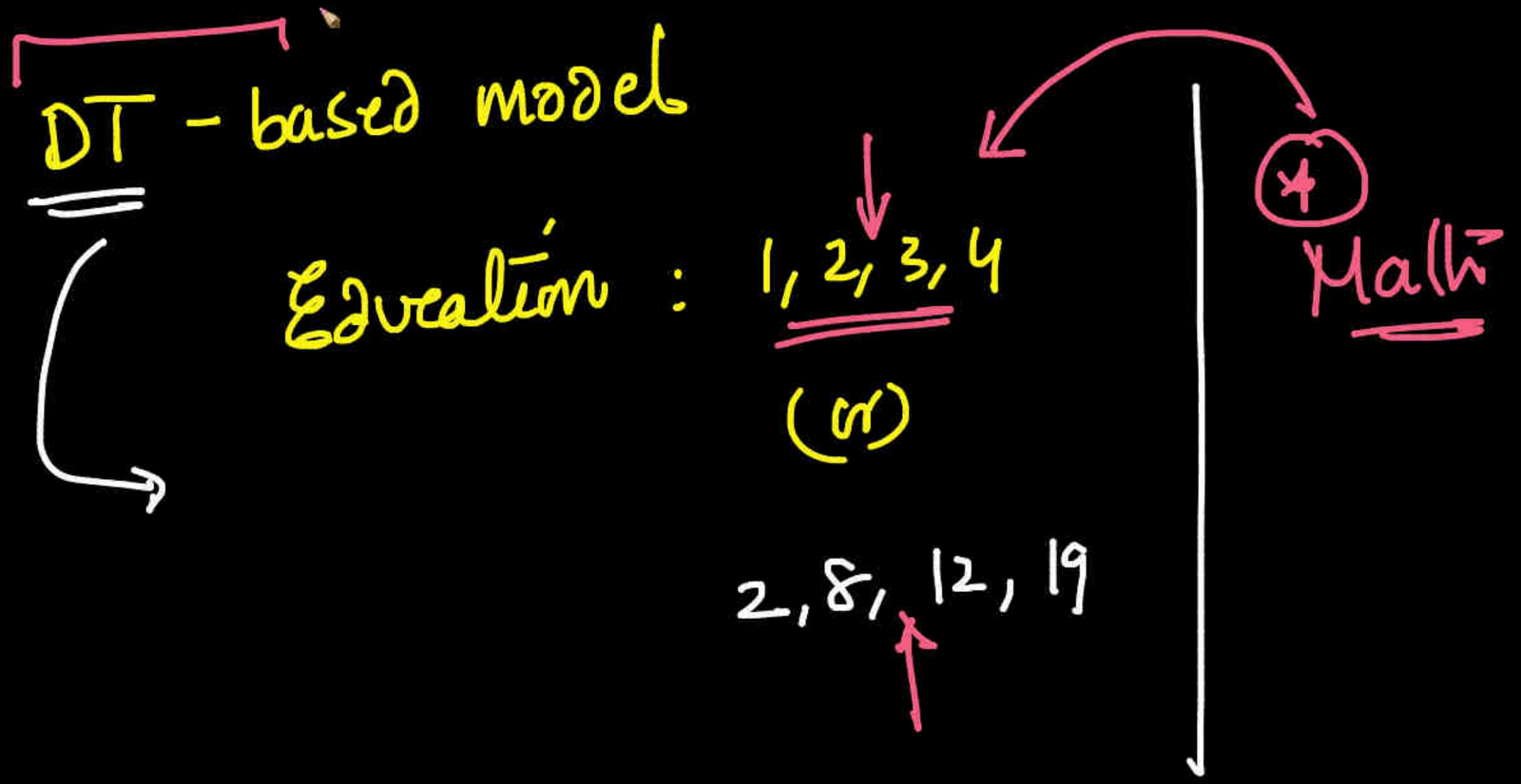
sch B M Ph2
1 2 3 4

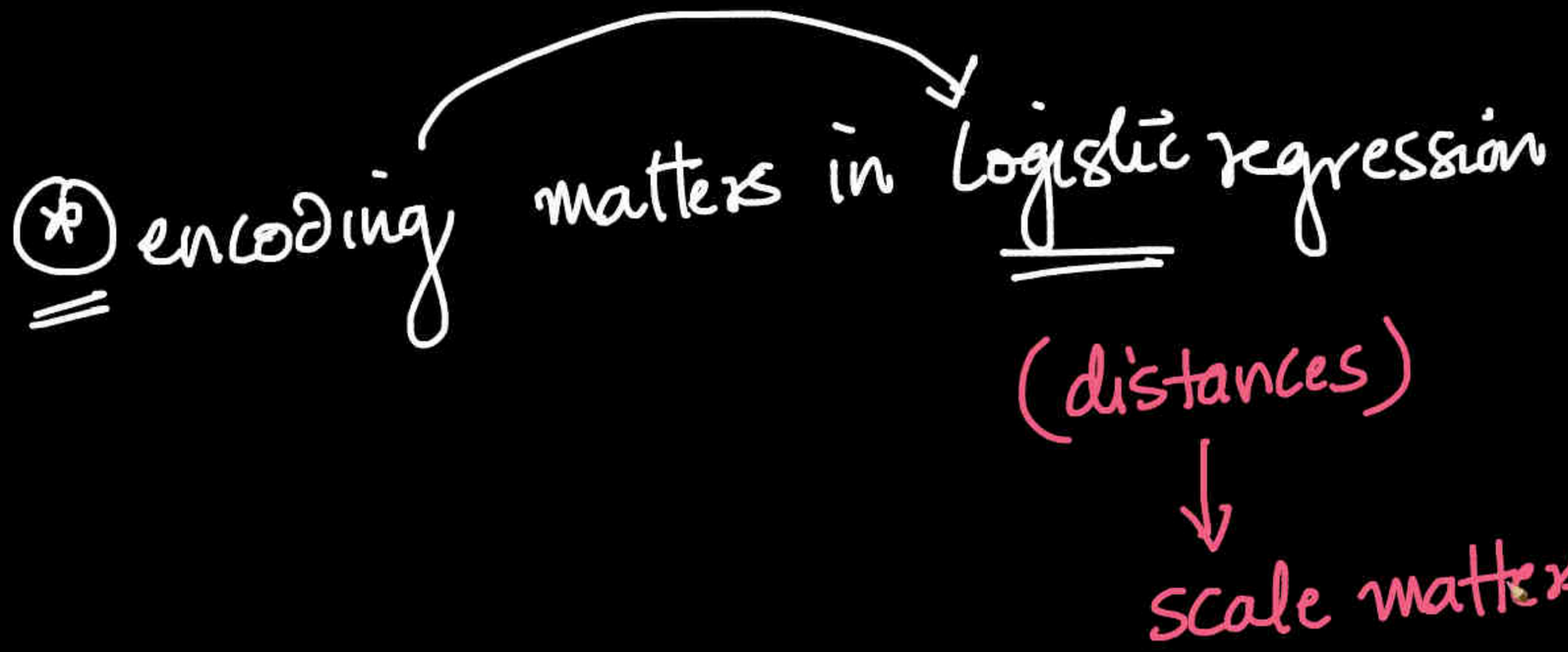
```
[221] df = pd.read_csv('HR-Employee-Attrition.csv')
      df.head()
```

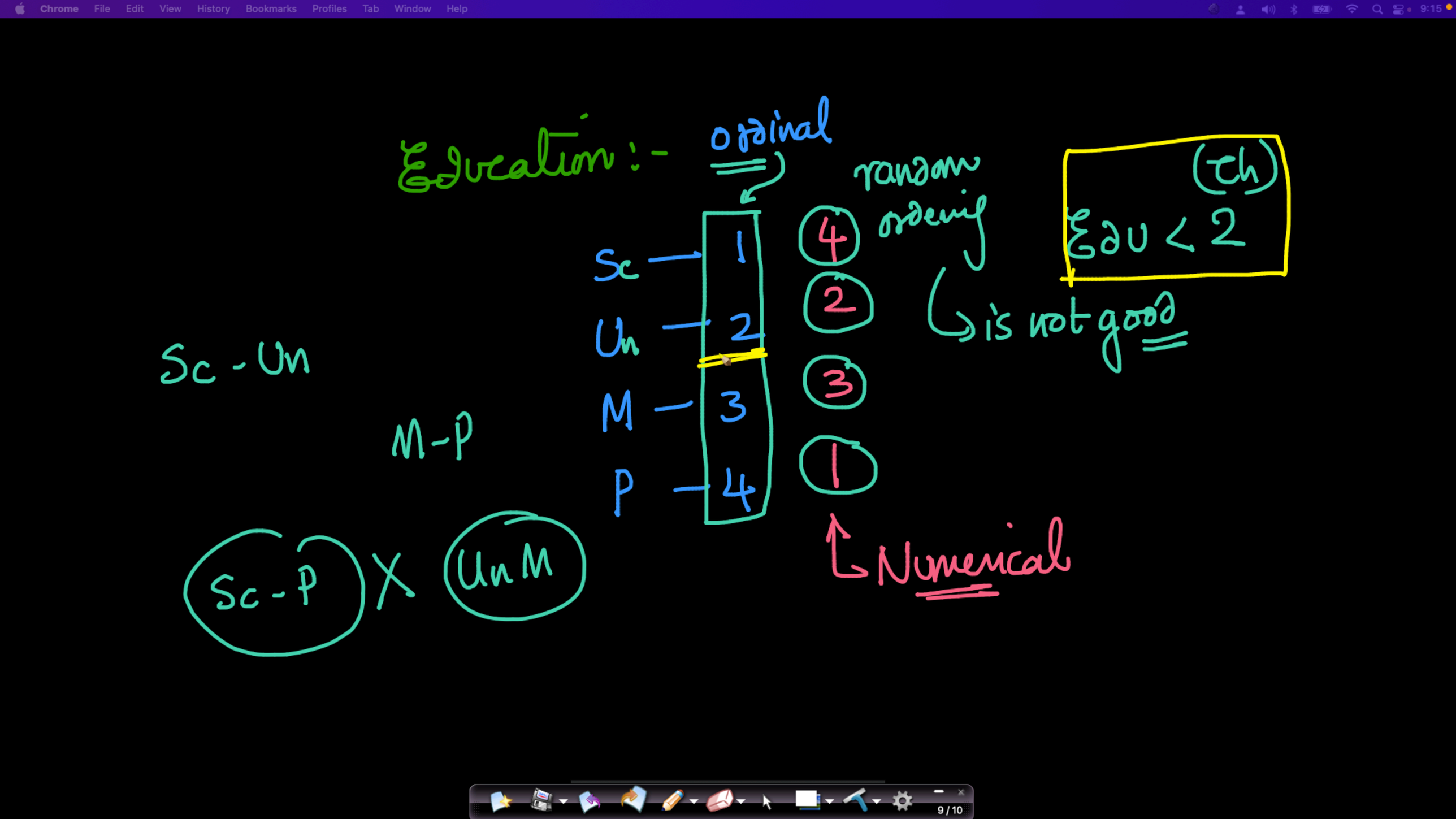
	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	E
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	

5 rows × 35 columns

```
df.info()
```






Education :-
original
Sc
Un
M
P
1
2
3
4
random ordering
4
2
3
1
is not good
(th)
Edu < 2
Sc - Un
M - P
Sc - P
X
Un M
Numerical

Logistic regr

Education → OHE ✓

↳ Target Encoding ✓

```
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

[108]
```
# Business problems
# 1.Find employees who are at high risk of attrition and  take remedial steps [ML Model]
# 2. Identify factors leading to attrition [Feature importance]
```

[109]
```
# Data preprocessing

# y_i's as 1 and 0 instead of Yes and No
df['Attrition'].value_counts()
# imbalanced data (~1:6 ratio): Need to account for this while building ML models
```

```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

```
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

[108]
```
# Business problems
# 1.Find employees who are at high risk of attrition and  take remedial steps [ML Model]
# 2. Identify factors leading to attrition [Feature importance]
```

[109]
```
# Data preprocessing

# y_i's as 1 and 0 instead of Yes and No
df['Attrition'].value_counts()
# imbalanced data (~1:6 ratio): Need to account for this while building ML models
```

```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

```
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

```
[108] # Business problems
      # 1.Find employees who are at high risk of attrition and  take remedial steps [ML Model]
      # 2. Identify factors leading to attrition [Feature importance]
```

```
[109] # Data preprocessing

      # y_i's as 1 and 0 instead of Yes and No
      df['Attrition'].value_counts()
      # imbalanced data (~1:6 ratio): Need to account for this while building ML models
```

```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

```
[108] # Business problems
      # 1.Find employees who are at high risk of attrition and  take remedial steps [ML Model]
      # 2. Identify factors leading to attrition [Feature importance]
```

```
# Data preprocessing

# y_i's as 1 and 0 instead of Yes and No
df['Attrition'].value_counts()
# imbalanced data (~1:6 ratio): Need to account for this while building ML models
```

class-weights ; SMOTE ; ...

```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

```
[110] df.loc[df['Attrition']=='Yes', 'Attrition'] = 1
      df.loc[df['Attrition']=='No', 'Attrition'] = 0
```

```
[111] df['Attrition'].value_counts()
```

```
0    1233
1     237
Name: Attrition, dtype: int64
```

```
# Data preprocessing

# y_i's as 1 and 0 instead of Yes and No
df['Attrition'].value_counts()
# imbalanced data (~1:6 ratio): Need to account for this while building ML models
```

```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

```
[110] df.loc[df['Attrition']=='Yes', 'Attrition'] = 1
      df.loc[df['Attrition']=='No', 'Attrition'] = 0
```

```
[111] df['Attrition'].value_counts()
```

```
0    1233
1     237
Name: Attrition, dtype: int64
```

```
[112] # Sklearn DT doesnot handle categorical values natively.
      # Options: OHE or Target Encoding (our choice)
      # Converting all non numerical features into Target encoded features
```

[110] df.loc[df['Attrition']=='Yes', 'Attrition'] = 1
df.loc[df['Attrition']=='No', 'Attrition'] = 0

[111] df['Attrition'].value_counts()

```
0    1233
1     237
Name: Attrition, dtype: int64
```

[112] # Sklearn DT doesnot handle categorical values natively.
Options: OHE or Target Encoding (our choice)
Converting all non numerical features into Target encoded features

Q: Why not Label Encoding in Sklearn?

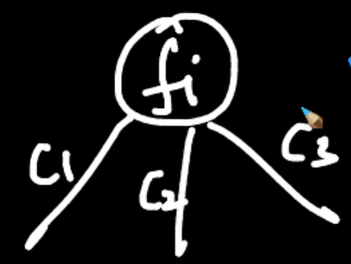
[113] !pip install category-encoders

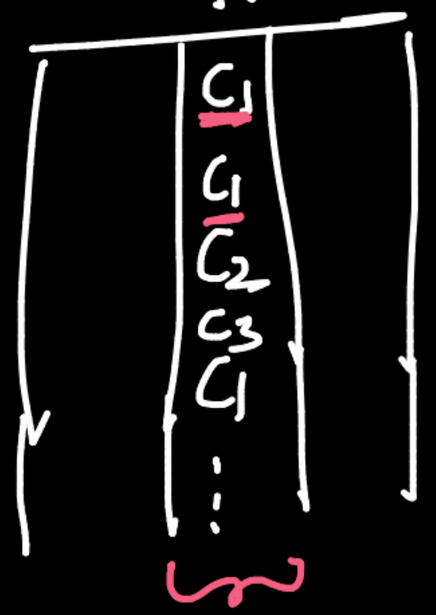
```
Requirement already satisfied: category-encoders in /usr/local/lib/python3.7/dist-packages (2.4.1)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.7/dist-packages (from category-encoder
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.7/dist-packages (from category-encoders) (1.2
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.7/dist-packages (from category-encoders) (0.5.
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.7/dist-packages (from category-encoders)
```

open source

Can you fix this?

sklearn → cannot handles categorical features natively





→ DT

CART

R → DT

fi: Categorical-features for DT

↳ numeric

avoid → OHE → explode #features

if many categories → Target encoding $P(y_i=1 \mid f_i=c_1)$

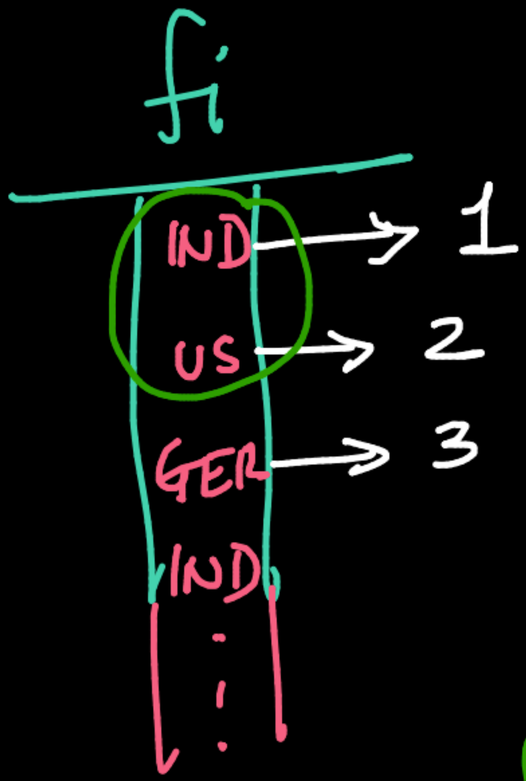
→ Label-encoding X

Label-encoding:-

Country: non-ordinal

creating random order

↳ random splits



randomly

$fi \leq 2$

sklearn.preprocessing.LabelEncoder

class sklearn.preprocessing.**LabelEncoder** [source]

Encode target labels with value between 0 and n_classes-1.

This transformer should be used to encode target values, *i.e.* `y`, and not the input `X`.

Read more in the User Guide.

New in version 0.12.

Attributes:

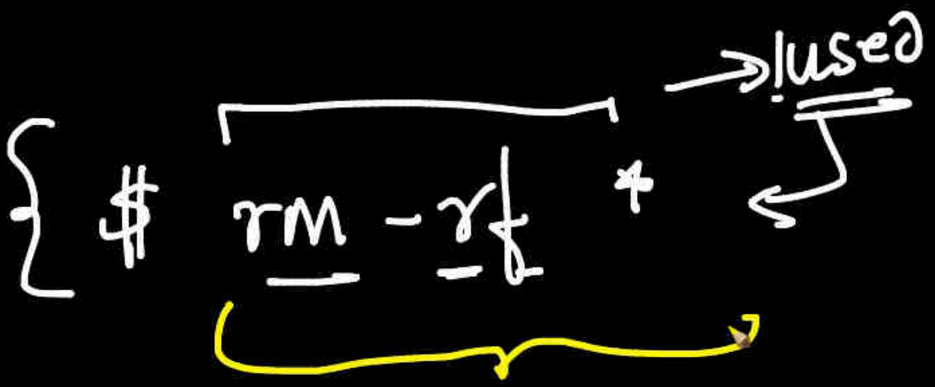
classes_ : *ndarray of shape (n_classes,)*
Holds the label for each class.

0
1
2

See also:

`OrdinalEncoder`
Encode categorical features using an ordinal encoding scheme.

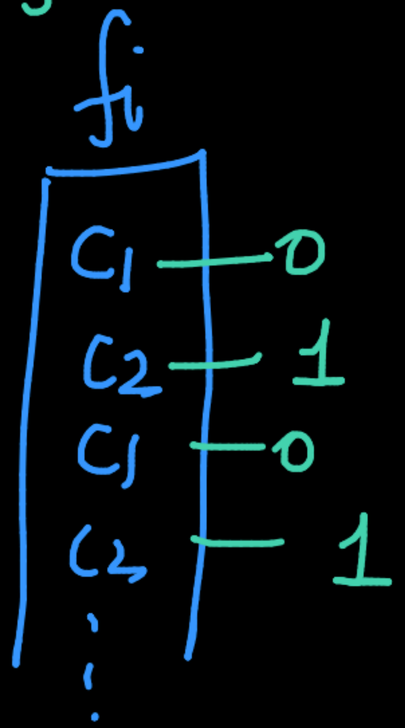
`OneHotEncoder`
Enco


{ # rm - r[ ] + 1 used

encode

5yxsvrndf → y_i → 0 / 1

label-encoding ✓



label-encoding can be done ..

Chrome File Edit View History Bookmarks Profiles Tab Window Help 9:35

DT and RF.ipynb - Colaborator | sklearn.tree.DecisionTreeClass | sklearn.ensemble.RandomFore | sklearn.preprocessing.LabelEr

colab.research.google.com/drive/1d2HJ9AKulf83i7pwmDrL-2FOCncFubRF#scrollTo=Oa6CGasjdbd8

+ Code + Text

RAM Disk

```
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from patsy>=0.5.1->category-encoders)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.20.0->ca
```

[114]
```
from pandas.api.types import is_numeric_dtype
from category_encoders import TargetEncoder

# COnvert all non-nuemric fetures into Target-encoded features
HR_col = list(df.columns)
HR_col.remove('Attrition')
for col in HR_col:
  if is_numeric_dtype(df[col])== False:
    encoder = TargetEncoder()
    df[col] = encoder.fit_transform(df[col], df['Attrition'])
```

[115]
```
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	E
0	41	1	0.149569	1102	0.206278	1	2	0.146865	1	
1	49	0	0.249097	279	0.138398	8	1	0.146865	1	

23 / 23

CE

fi

$c_1 \rightarrow$ 

$c_2 \rightarrow$ 

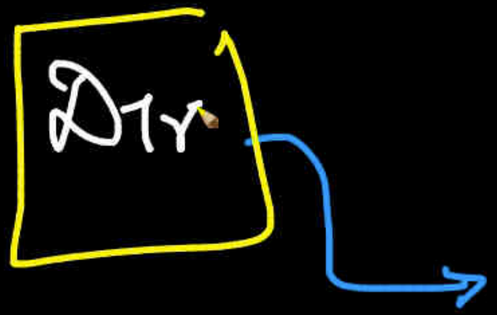
D → D_{TR} → Target Encoding ✓

D → D_{CV} ✓

D → D_{te} ✓

[no-leakage]

✓



f_i : C_1, C_2, C_3

$f_i : C_1 \rightarrow$ 0.15

$f_i : C_2 \rightarrow$ 0.60

$f_i : C_3 \rightarrow$ 0.02

D_{cv} or D_{te}

replace

$f_i : c_1 \rightarrow 0.15$

$f_i : c_2 \rightarrow 0.6$

$f_i : c_3 \rightarrow 0.02$

Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.20.0->ca

```
HR_col = list(df.columns)
HR_col.remove('Attrition')
for col in HR_col:
  if is_numeric_dtype(df[col])== False:
    print(col)
```

[114]
```
from pandas.api.types import is_numeric_dtype
from category_encoders import TargetEncoder

# COnvert all non-nuemric features into Target-encoded features
HR_col = list(df.columns)
HR_col.remove('Attrition')
for col in HR_col:
  if is_numeric_dtype(df[col])== False:
    encoder = TargetEncoder()
    df[col] = encoder.fit_transform(df[col], df['Attrition'])
```

[115]
```
df.head()
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help

DT and RF.ipynb - Colaborator | sklearn.tree.DecisionTreeClass | sklearn.ensemble.RandomFore | sklearn.preprocessing.LabelEr

colab.research.google.com/drive/1d2HJ9AKulf83i7pwmDrL-2FOCncFubRF#scrollTo=0kvclMDBL1HW

+ Code + Text

RAM
Disk

```
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.20.0->ca
```

```
HR_col = list(df.columns)
HR_col.remove('Attrition')
for col in HR_col:
  if is_numeric_dtype(df[col])== False:
    print(col)
```

```
BusinessTravel
Department
EducationField
Gender
JobRole
MaritalStatus
Over18
OverTime
```

— non-numeric

Educ: ordinal → numerical ✓

```
[114] from pandas.api.types import is_numeric_dtype
      from category_encoders import TargetEncoder

      # COnvert all non-nuemric features into Target-encoded features
      HR_col = list(df.columns)
      HR_col.remove('Attrition')
      for col in HR_col:
```

28 / 28

Chrome File Edit View History Bookmarks Profiles Tab Window Help 9:41

DT and RF.ipynb - Colaborator | sklearn.tree.DecisionTreeClass | sklearn.ensemble.RandomFore | sklearn.preprocessing.LabelEr

colab.research.google.com/drive/1d2HJ9AKulf83i7pwmDrL-2FOCncFubRF#scrollTo=0kvclMDBL1HW

Update

+ Code + Text

RAM
Disk

```
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.20.0->ca
```

```
HR_col = list(df.columns)
HR_col.remove('Attrition')
for col in HR_col:
  if is_numeric_dtype(df[col])== False:
    print(col)
```

```
BusinessTravel
Department
EducationField
Gender
JobRole
MaritalStatus
Over18
OverTime
```

D → Target encoding X

DTr ✓

```
[114] from pandas.api.types import is_numeric_dtype
      from category_encoders import TargetEncoder

      # COnvert all non-nuemric features into Target-encoded features
      HR_col = list(df.columns)
      HR_col.remove('Attrition')
      for col in HR_col:
```

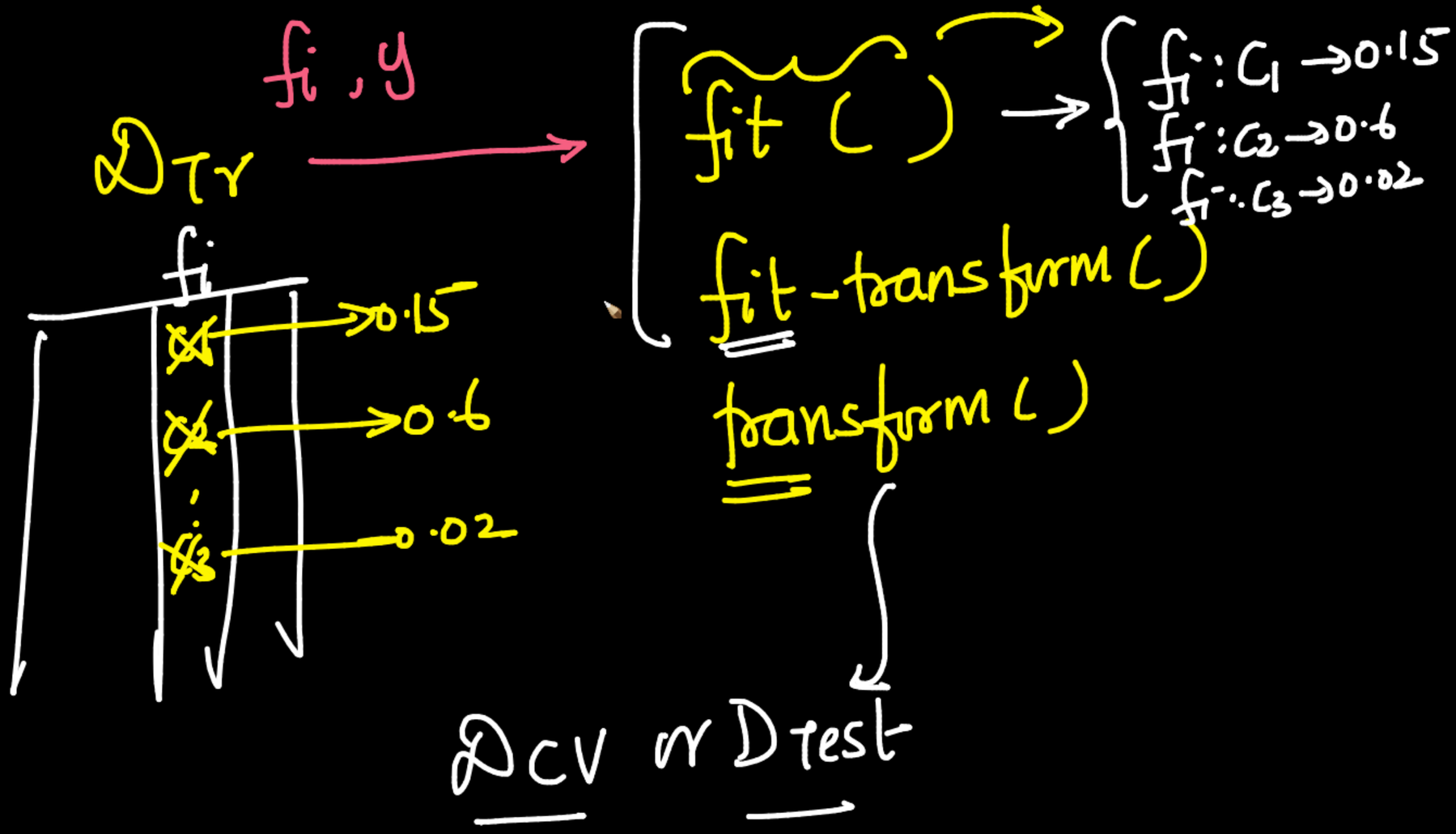
29 / 29

```
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.20.0->ca
```

```
HR_col = list(df.columns)
HR_col.remove('Attrition')
for col in HR_col:
  if is_numeric_dtype(df[col])== False:
    print(col)
```

```
BusinessTravel
Department
EducationField
Gender
JobRole
MaritalStatus
Over18
OverTime
```

```
[114] from pandas.api.types import is_numeric_dtype
      from category_encoders import TargetEncoder

      # COnvert all non-nuemric features into Target-encoded features
      HR_col = list(df.columns)
      HR_col.remove('Attrition')
      for col in HR_col:
```



References

1

A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems, from

https://dl.acm.org/citation.cfm?id=507538

Methods

`fit` (X, y, **kwargs)	Fit encoder according to X and y.
`fit_transform` (X[, y])	Encoders that utilize the target must make sure that the traini
`get_feature_names` ()	Returns the names of all transformed / added columns.
`get_params` ([deep])	Get parameters for this estimator.
`set_params` (**params)	Set the parameters of this estimator.
`transform` (X[, y, override_return_df])	Perform the transformation to new categorical data.

fit_target_encoding	
target_encode	

`fit`(*X*, *y*, ***kwargs*) [source]

Fit encoder according to X and y.

```
from pandas.api.types import is_numeric_dtype
from category_encoders import TargetEncoder

# Convert all non-numeric features into Target-encoded features
HR_col = list(df.columns)
HR_col.remove('Attrition')
for col in HR_col:
  if is_numeric_dtype(df[col])== False:
    encoder = TargetEncoder()
    df[col] = encoder.fit_transform(df[col], df['Attrition'])
```

```
[115] df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	E
0	41	1	0.149569	1102	0.206278	1	2	0.146865	1	
1	49	0	0.249097	279	0.138398	8	1	0.146865	1	
2	37	1	0.149569	1373	0.138398	2	2	0.134146	1	
3	33	0	0.249097	1392	0.138398	3	4	0.146865	1	
4	27	0	0.149569	591	0.138398	2	1	0.135776	1	

```
from pandas.api.types import is_numeric_dtype
from category_encoders import TargetEncoder

# Convert all non-numeric features into Target-encoded features
HR_col = list(df.columns)
HR_col.remove('Attrition')
for col in HR_col:
  if is_numeric_dtype(df[col])== False:
    encoder = TargetEncoder()
    df[col] = encoder.fit_transform(df[col], df['Attrition'])
```

```
[115] df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	E
0	41	1	0.149569	1102	0.206278	1	2	0.146865	1	
1	49	0	0.249097	279	0.138398	8	1	0.146865	1	
2	37	1	0.149569	1373	0.138398	2	2	0.134146	1	
3	33	0	0.249097	1392	0.138398	3	4	0.146865	1	
4	27	0	0.149569	591	0.138398	2	1	0.135776	1	

```
#Check if any non-numeric column exists in the DF
for col in HR_col:
  if is_numeric_dtype(df[col])== False:
    print(col)
```

```
[117] # Remove useless columns
      df['EmployeeNumber'].value_counts()
```

```
1       1
1391    1
1389    1
1387    1
1383    1
       ..
659     1
657     1
656     1
655     1
2068    1
Name: EmployeeNumber, Length: 1470, dtype: int64
```

```
[118] df['EmployeeCount'].value_counts()
```

```
1    1470
```

```
1          1
1391       1
1389       1
1387       1
1383       1
          ..
659        1
657        1
656        1
655        1
2068       1
Name: EmployeeNumber, Length: 1470, dtype: int64
```

```
df['EmployeeCount'].value_counts()
```

```
1    1470
Name: EmployeeCount, dtype: int64
```

```
[119] df['StandardHours'].value_counts()
```

```
80    1470
Name: StandardHours, dtype: int64
```

```
[120] df['Over18'].value_counts()
```

```
[118] df['EmployeeCount'].value_counts()
```

```
1    1470
Name: EmployeeCount, dtype: int64
```

```
[119] df['StandardHours'].value_counts()
```

```
80    1470
Name: StandardHours, dtype: int64
```


```
df['Over18'].value_counts()
```

```
0.161224    1470
Name: Over18, dtype: int64
```

```
[121] y = df['Attrition']
      y = y.astype('int')
```

```
[122] df.drop(['Attrition', 'EmployeeCount', 'EmployeeNumber',
               'StandardHours', 'Over18'], axis=1, inplace=True)
      print('Size of Full dataset is: {}'.format(df.shape))
```

```
[118] df['EmployeeCount'].value_counts()
```

```
1    1470
Name: EmployeeCount, dtype: int64
```

```
[119] df['StandardHours'].value_counts()
```

```
80    1470
Name: StandardHours, dtype: int64
```

```
df['Over18'].value_counts()
```

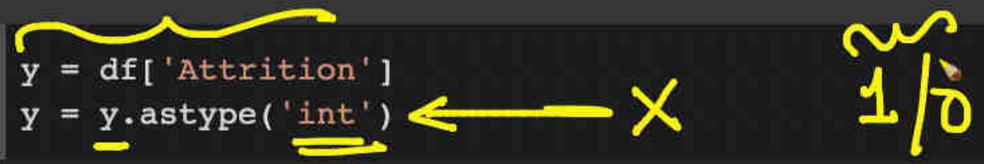
```
0.161224    1470
Name: Over18, dtype: int64
```

```
[121] y = df['Attrition']
      y = y.astype('int')
```

```
[122] df.drop(['Attrition', 'EmployeeCount', 'EmployeeNumber',
               'StandardHours', 'Over18'], axis=1, inplace=True)
      print('Size of Full dataset is: {}'.format(df.shape))
```

```
[120] df['Over18'].value_counts()

      0.161224    1470
      Name: Over18, dtype: int64
```

```
      y = df['Attrition']
      y = y.astype('int')
```




```
[122] df.drop(['Attrition', 'EmployeeCount', 'EmployeeNumber',
                'StandardHours', 'Over18'], axis=1, inplace=True)
      print('Size of Full dataset is: {}'.format(df.shape))

      X = df
```

```
      Size of Full dataset is: (1470, 30)
```

```
[123] # Train, CV, test split
      from sklearn.model_selection import train_test_split
      #0.6, 0.2, 0.2 split

      X_tr_cv, X_test, y_tr_cv, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

      X_train, X_val, y_train, y_val = train_test_split(X_tr_cv, y_tr_cv, test_size=0.25, random_state=42) # 0.25 x 0.8 =
```

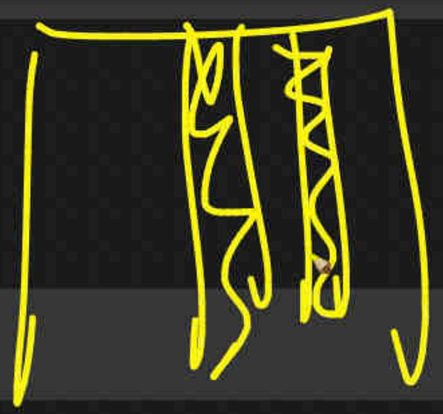
```
df['Over18'].value_counts()
```

```
0.161224    1470
Name: Over18, dtype: int64
```

```
y = df['Attrition']
y = y.astype('int')
```

```
df.drop(['Attrition', 'EmployeeCount', 'EmployeeNumber',
         'StandardHours', 'Over18'], axis=1, inplace=True)
print('Size of Full dataset is: {}'.format(df.shape))

X = df
```

```
Size of Full dataset is: (1470, 30)
```



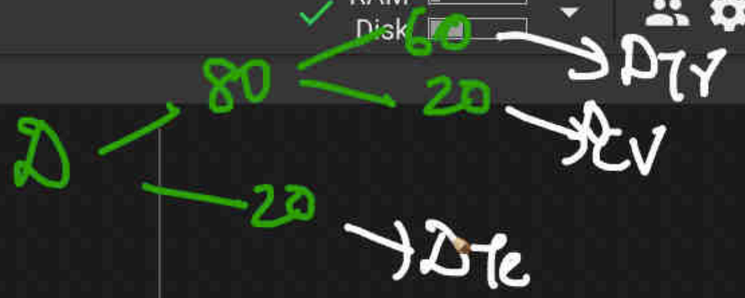
```
# Train, CV, test split
from sklearn.model_selection import train_test_split
#0.6, 0.2, 0.2 split

X_tr_cv, X_test, y_tr_cv, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

X_train, X_val, y_train, y_val = train_test_split(X_tr_cv, y_tr_cv, test_size=0.25, random_state=42) # 0.25 x 0.8 =
```

```
[120] df['Over18'].value_counts()

      0.161224    1470
      Name: Over18, dtype: int64
```

```
      y = df['Attrition']
      y = y.astype('int')
```

```
[122] df.drop(['Attrition', 'EmployeeCount', 'EmployeeNumber',
                'StandardHours', 'Over18'], axis=1, inplace=True)
      print('Size of Full dataset is: {}'.format(df.shape))

      X = df

      Size of Full dataset is: (1470, 30)
```

```
[123] # Train, CV, test split
      from sklearn.model_selection import train_test_split
      #0.6, 0.2, 0.2 split

      X_tr_cv, X_test, y_tr_cv, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

      X_train, X_val, y_train, y_val = train_test_split(X_tr_cv, y_tr_cv, test_size=0.25, random_state=42) # 0.25 x 0.8 =
```



[123]
```
# Train, CV, test split
from sklearn.model_selection import train_test_split
#0.6, 0.2, 0.2 split

X_tr_cv, X_test, y_tr_cv, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

X_train, X_val, y_train, y_val = train_test_split(X_tr_cv, y_tr_cv, test_size=0.25, random_state=42) # 0.25 x 0.8 =
```

[124]
```
print("Number transactions X_train dataset: ", X_train.shape)
print("Number transactions y_train dataset: ", y_train.shape)
print("Number transactions X_CV dataset: ", X_val.shape)
print("Number transactions y_CV dataset: ", y_val.shape)
print("Number transactions X_test dataset: ", X_test.shape)
print("Number transactions y_test dataset: ", y_test.shape)
```

```
Number transactions X_train dataset:  (882, 30)
Number transactions y_train dataset:  (882,)
Number transactions X_CV dataset:  (294, 30)
Number transactions y_CV dataset:  (294,)
Number transactions X_test dataset:  (294, 30)
Number transactions y_test dataset:  (294,)
```

Decision Trees

[123]
```
# Train, CV, test split
from sklearn.model_selection import train_test_split
#0.6, 0.2, 0.2 split

X_tr_cv, X_test, y_tr_cv, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

X_train, X_val, y_train, y_val = train_test_split(X_tr_cv, y_tr_cv, test_size=0.25, random_state=42) # 0.25 x 0.8 =
```

[124]
```
print("Number transactions X_train dataset: ", X_train.shape)
print("Number transactions y_train dataset: ", y_train.shape)
print("Number transactions X_CV dataset: ", X_val.shape)
print("Number transactions y_CV dataset: ", y_val.shape)
print("Number transactions X_test dataset: ", X_test.shape)
print("Number transactions y_test dataset: ", y_test.shape)
```

```
Number transactions X_train dataset:  (882, 30)
Number transactions y_train dataset:  (882,)
Number transactions X_CV dataset:  (294, 30)
Number transactions y_CV dataset:  (294,)
Number transactions X_test dataset:  (294, 30)
Number transactions y_test dataset:  (294,)
```

Decision Trees

```
# Hyper-pram tuning + DT model
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score

train_scores = []
val_scores = []

l=1
u=20
d=1
w=1.0

for depth in np.arange(l,u,d):
  clf = DecisionTreeClassifier(random_state=0, max_depth=depth, class_weight={ 0:0.1, 1:w } )
  clf.fit(X_train, y_train)
  train_y_pred = clf.predict(X_train)
  val_y_pred = clf.predict(X_val)
  train_score = f1_score(y_train, train_y_pred)
  val_score = f1_score(y_val, val_y_pred)
  train_scores.append(train_score)
  val_scores.append(val_score)
```

```
[209] import matplotlib.pyplot as plt
```

```
# Hyper-pram tuning + DT model
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score

train_scores = []
val_scores = []

l=1
u=20
d=1
w=1.0

for depth in np.arange(l,u,d):
  clf = DecisionTreeClassifier(random_state=0, max_depth=depth, class_weight={ 0:0.1, 1:w } )
  clf.fit(X_train, y_train)
  train_y_pred = clf.predict(X_train)
  val_y_pred = clf.predict(X_val)
  train_score = f1_score(y_train, train_y_pred)
  val_score = f1_score(y_val, val_y_pred)
  train_scores.append(train_score)
  val_scores.append(val_score)
```

1,2,3,4, . . . 20

1 → 20

```
[209] import matplotlib.pyplot as plt
```

```
# Hyper-pram tuning + DT model
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score

train_scores = []
val_scores = []

l=1
u=20
d=1
w=1.0

for depth in np.arange(l,u,d):
  clf = DecisionTreeClassifier(random_state=0, max_depth=depth, class_weight={ 0:0.1, 1:w } )
  clf.fit(X_train, y_train)
  train_y_pred = clf.predict(X_train)
  val_y_pred = clf.predict(X_val)
  train_score = f1_score(y_train, train_y_pred)
  val_score = f1_score(y_val, val_y_pred)
  train_scores.append(train_score)
  val_scores.append(val_score)
```

depth = 1, 2, ... 20

1:1

1:10

hyper-param

1:0

```
[209] import matplotlib.pyplot as plt
```

Combinations

depth = 1

depth = 1

"

depth = 2

"

Class weights : 0.1 : 0.6

" : 0.1 : 1.0

" : 0.1 : 2.0

"

C.W = 0.1 : 0.6

0.1 : 1.0

"

⋮

⋮

GBDT → automate

GridSearch
RandomSearch

on HP

```
l=1
u=20
d=1
w=1.0

for depth in np.arange(l,u,d):
  clf = DecisionTreeClassifier(random_state=0, max_depth=depth, class_weight={ 0:0.1, 1:w } )
  clf.fit(X_train, y_train)
  train_y_pred = clf.predict(X_train)
  val_y_pred = clf.predict(X_val)
  train_score = f1_score(y_train, train_y_pred)
  val_score = f1_score(y_val, val_y_pred)
  train_scores.append(train_score)
  val_scores.append(val_score)
```

tunable

[209]
```
import matplotlib.pyplot as plt

plt.figure()
plt.plot(list(np.arange(l,u,d)), train_scores, label="train")
plt.plot(list(np.arange(l,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("lambda")
plt.ylabel("F1-Score")
plt.grid()
```

```
l=1
u=20
d=1
w=1.0

for depth in np.arange(l,u,d):
  clf = DecisionTreeClassifier(random_state=0, max_depth=depth, class_weight={ 0:0.1, 1:w } )
  clf.fit(X_train, y_train)
  train_y_pred = clf.predict(X_train)
  val_y_pred = clf.predict(X_val)
  train_score = f1_score(y_train, train_y_pred)
  val_score = f1_score(y_val, val_y_pred)
  train_scores.append(train_score)
  val_scores.append(val_score)
```

```
[209] import matplotlib.pyplot as plt

    plt.figure()
    plt.plot(list(np.arange(l,u,d)), train_scores, label="train")
    plt.plot(list(np.arange(l,u,d)), val_scores, label="val")
    plt.legend(loc='lower right')
    plt.xlabel("lambda")
    plt.ylabel("F1-Score")
    plt.grid()
```

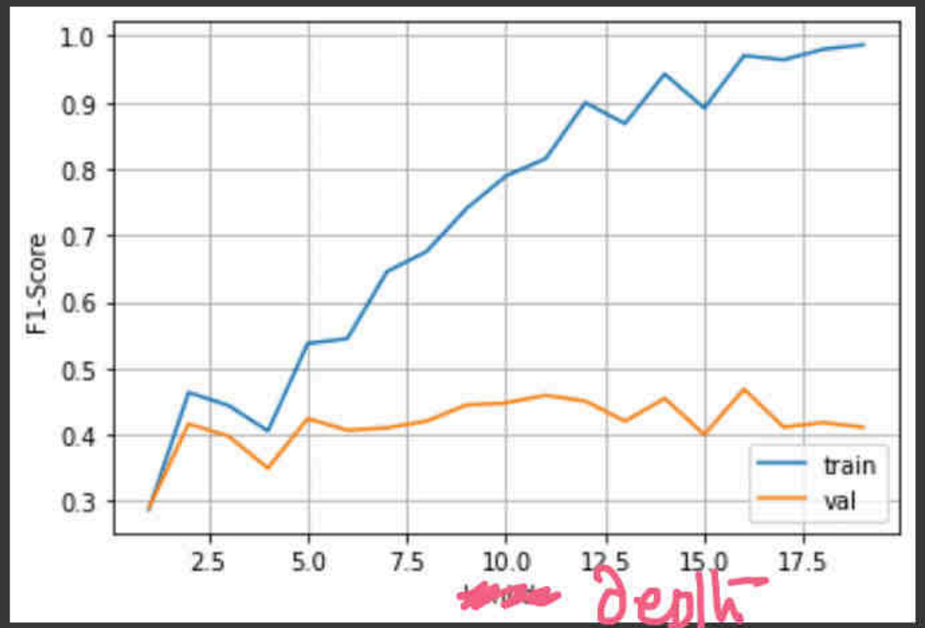
```
l=1
u=20
d=1
w=1.0

for depth in np.arange(l,u,d):
  clf = DecisionTreeClassifier(random_state=0, max_depth=depth, class_weight={ 0:0.1, 1:w } )
  clf.fit(X_train, y_train)
  train_y_pred = clf.predict(X_train)
  val_y_pred = clf.predict(X_val)
  train_score = f1_score(y_train, train_y_pred)
  val_score = f1_score(y_val, val_y_pred)
  train_scores.append(train_score)
  val_scores.append(val_score)
```

```
[209] import matplotlib.pyplot as plt

    plt.figure()
    plt.plot(list(np.arange(l,u,d)), train_scores, label="train")
    plt.plot(list(np.arange(l,u,d)), val_scores, label="val")
    plt.legend(loc='lower right')
    plt.xlabel("lambda")
    plt.ylabel("F1-Score")
    plt.grid()
```

```
l=1
u=20
d=1
w=1.0

for depth in np.arange(l,u,d):
  clf = DecisionTreeClassifier(random_state=0, max_depth=depth, class_weight={ 0:0.1, 1:w } )
  clf.fit(X_train, y_train)
  train_y_pred = clf.predict(X_train)
  val_y_pred = clf.predict(X_val)
  train_score = f1_score(y_train, train_y_pred)
  val_score = f1_score(y_val, val_y_pred)
  train_scores.append(train_score)
  val_scores.append(val_score)
```

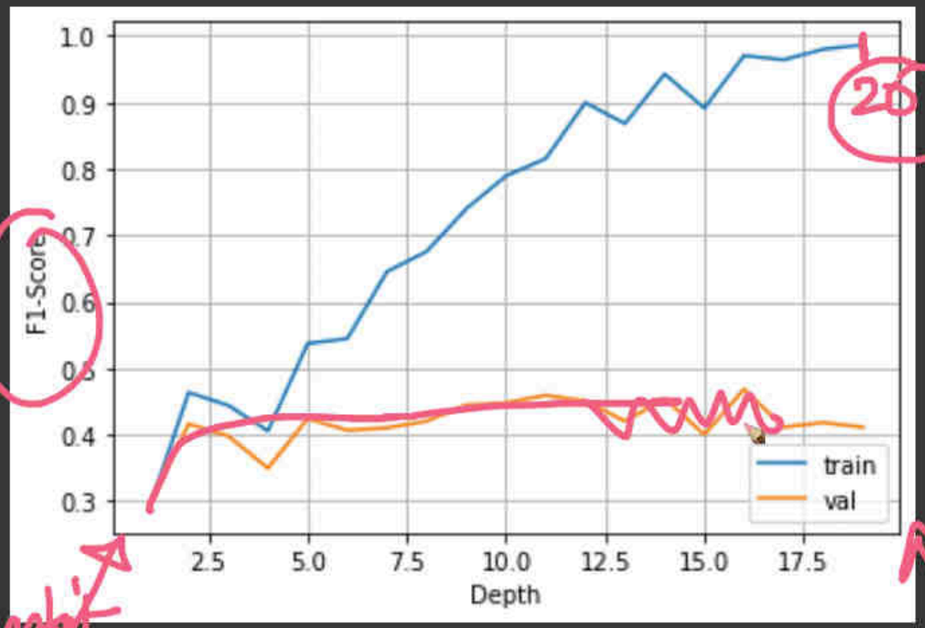
[209]
```
import matplotlib.pyplot as plt

plt.figure()
plt.plot(list(np.arange(l,u,d)), train_scores, label="train")
plt.plot(list(np.arange(l,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("lambda")
plt.ylabel("F1-Score")
plt.grid()
```

```
plt.plot(list(np.arange(l,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("lambda")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```




```
[210] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

      best_idx = np.argmax(val_scores)
```

```
plt.plot(list(np.arange(1,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("Depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```



800 train pts 30 feat

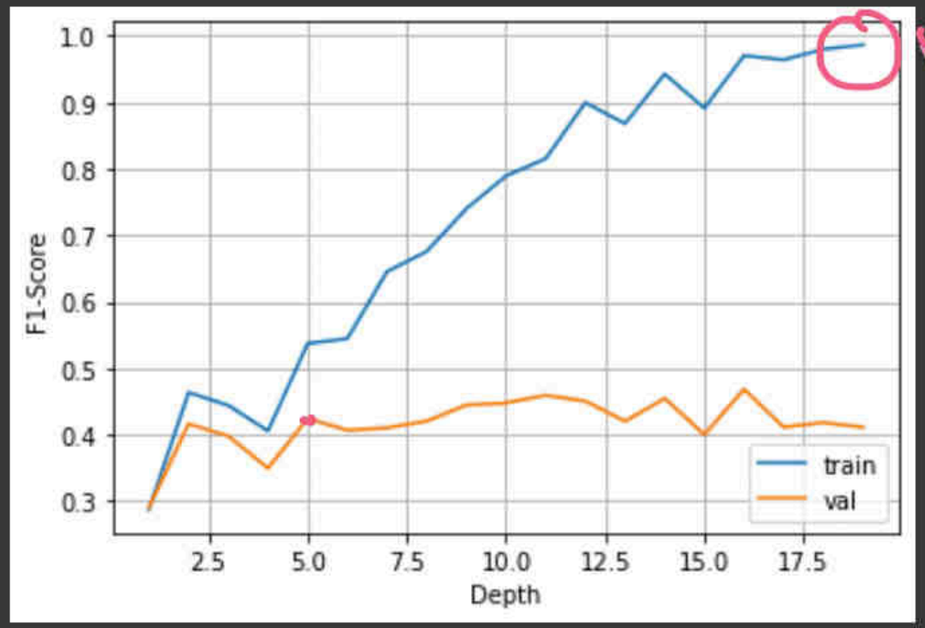
20

underfit

overfit

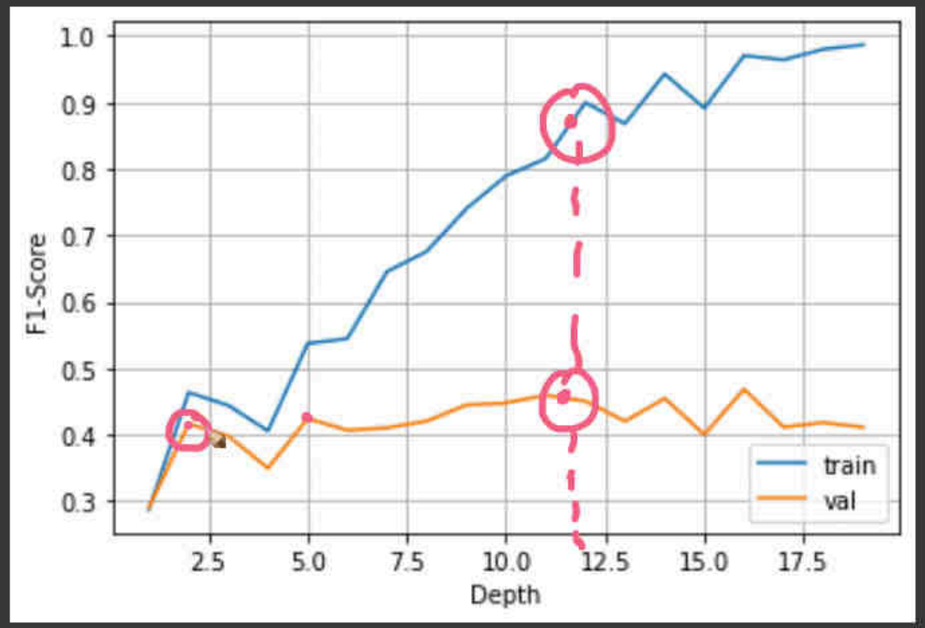
```
[210] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

      best_idx = np.argmax(val_scores)
```

```
plt.plot(list(np.arange(l,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("Depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```



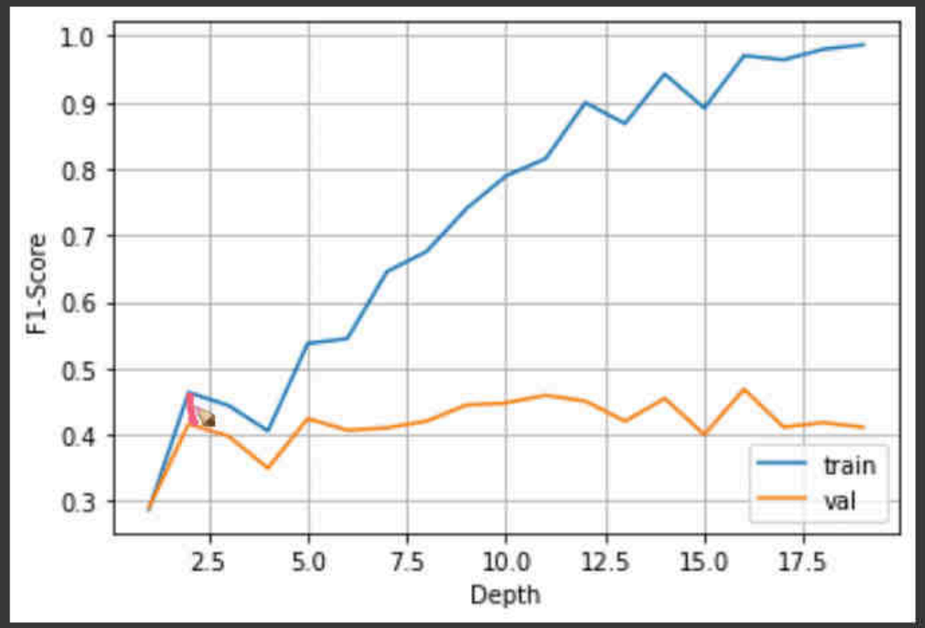

```
[210] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

      best_idx = np.argmax(val_scores)
```

```
plt.plot(list(np.arange(1,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("Depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```



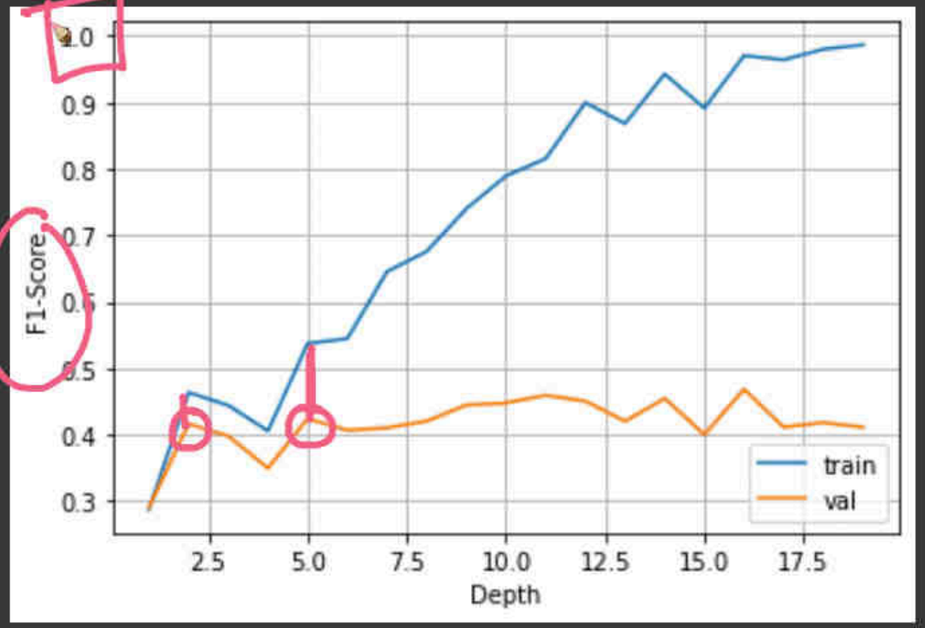

```
[210] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

      best_idx = np.argmax(val_scores)
```

```
plt.plot(list(np.arange(l,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("Depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```



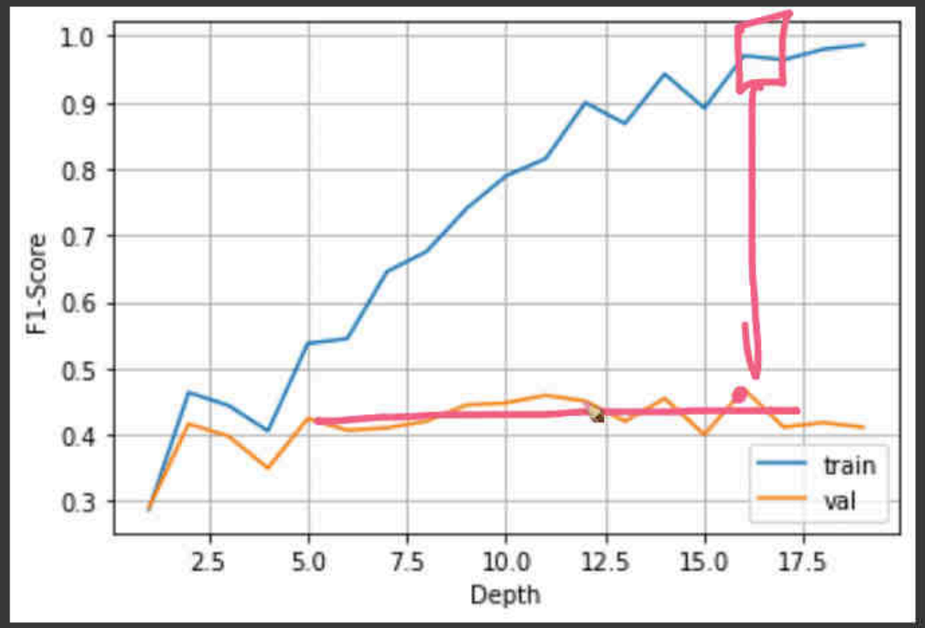

```
[210] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

      best_idx = np.argmax(val_scores)
```

```
plt.plot(list(np.arange(1,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("Depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```



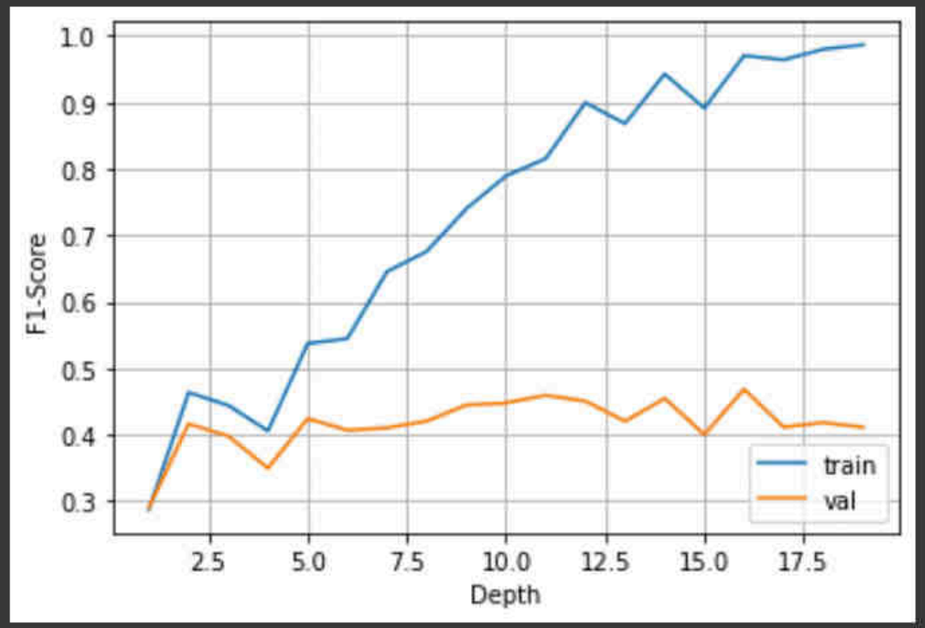

```
[210] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

      best_idx = np.argmax(val_scores)
```

```
plt.plot(list(np.arange(1,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("Depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```



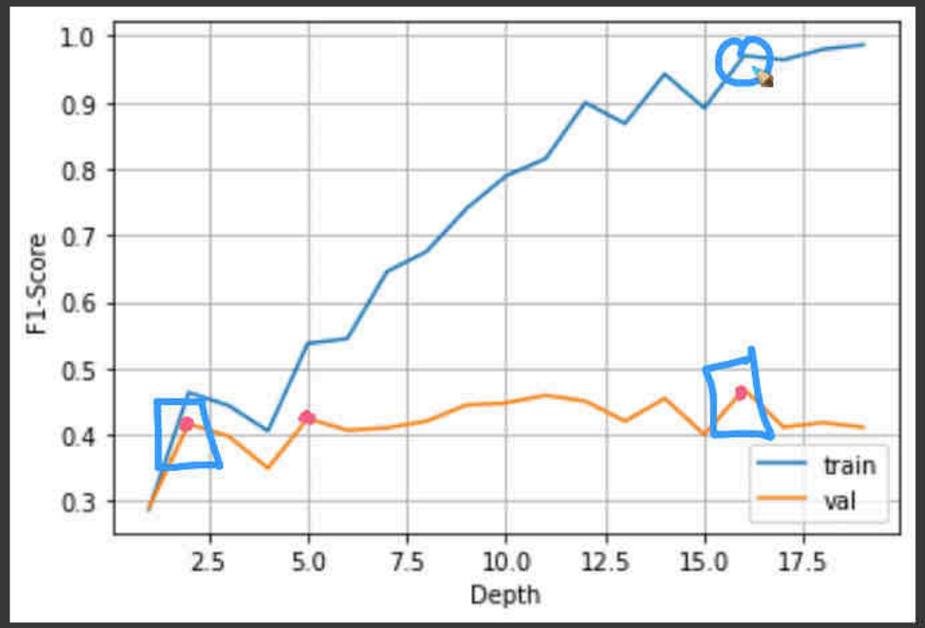

```
[210] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

      best_idx = np.argmax(val_scores)
```

```
plt.plot(list(np.arange(1,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("Depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```




```
[210] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

      best_idx = np.argmax(val_scores)
```

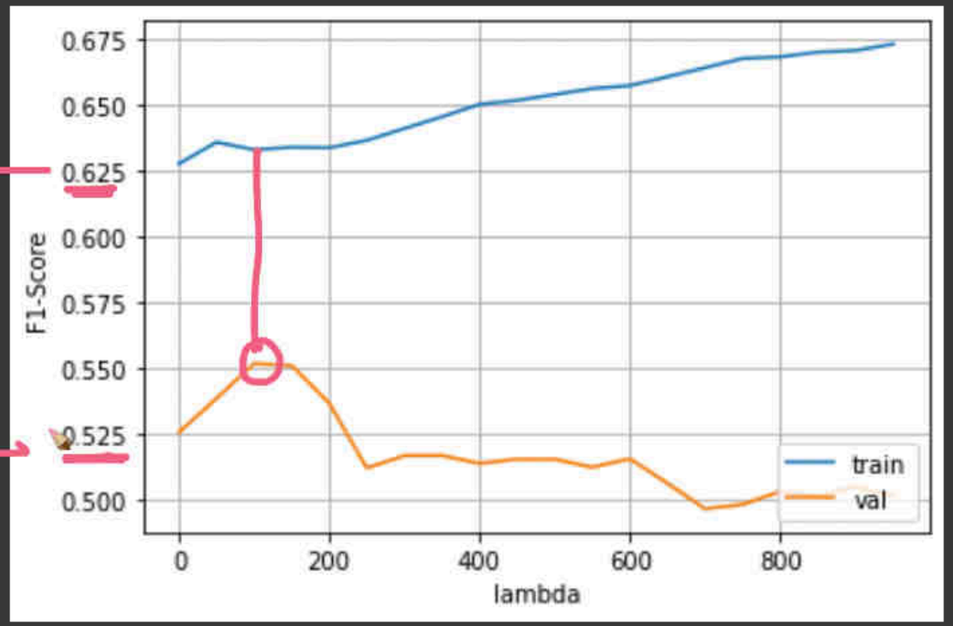
```
plt.plot(list(np.arange(1,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("Depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```



800

```
[210] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

      best_idx = np.argmax(val_scores)
```

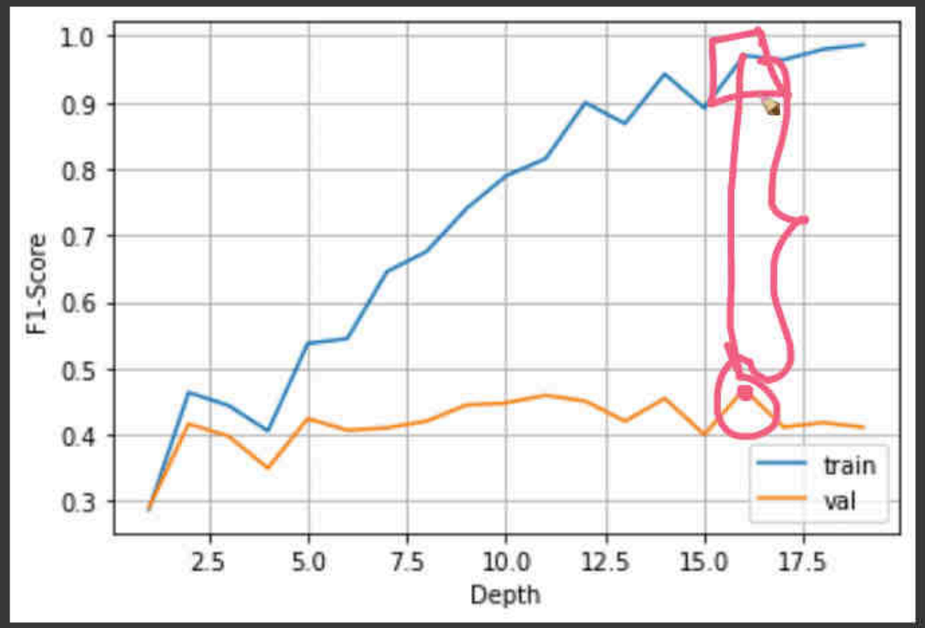
```
plt.grid()
plt.show()
```



Logistic Reg

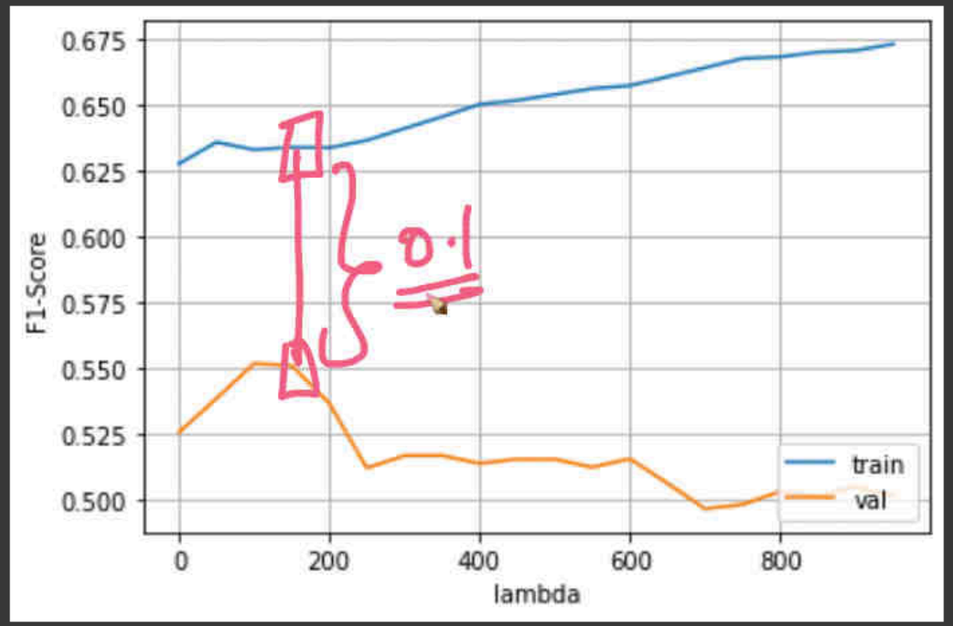
```
# minority class needs more weighting

# Hyper-pram tuning
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.metrics import f1_score
```

```
plt.plot(list(np.arange(l,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("Depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```



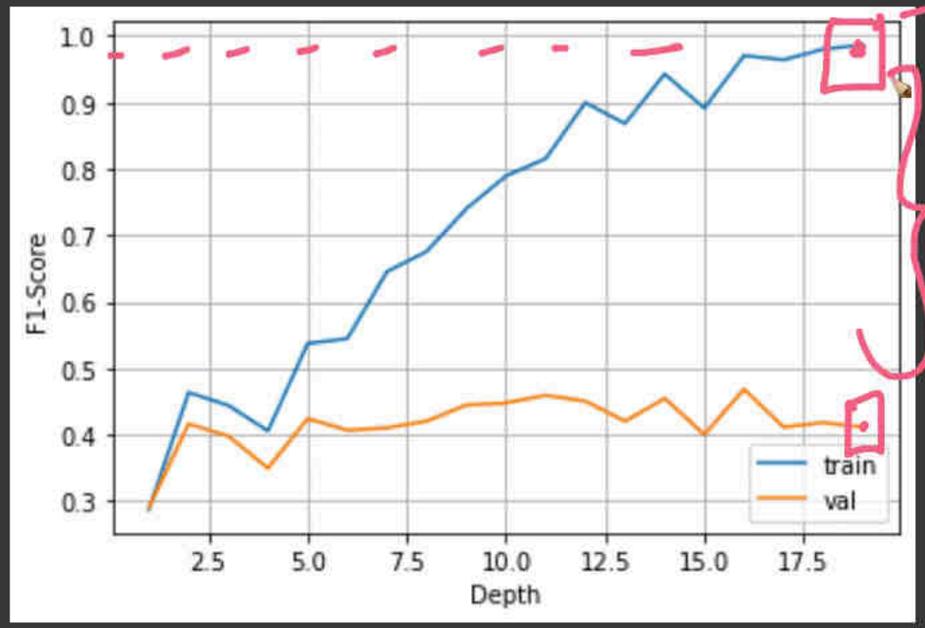

```
[210] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

      best_idx = np.argmax(val_scores)
```

```
[ ]  plt.grid()
     plt.show()
```



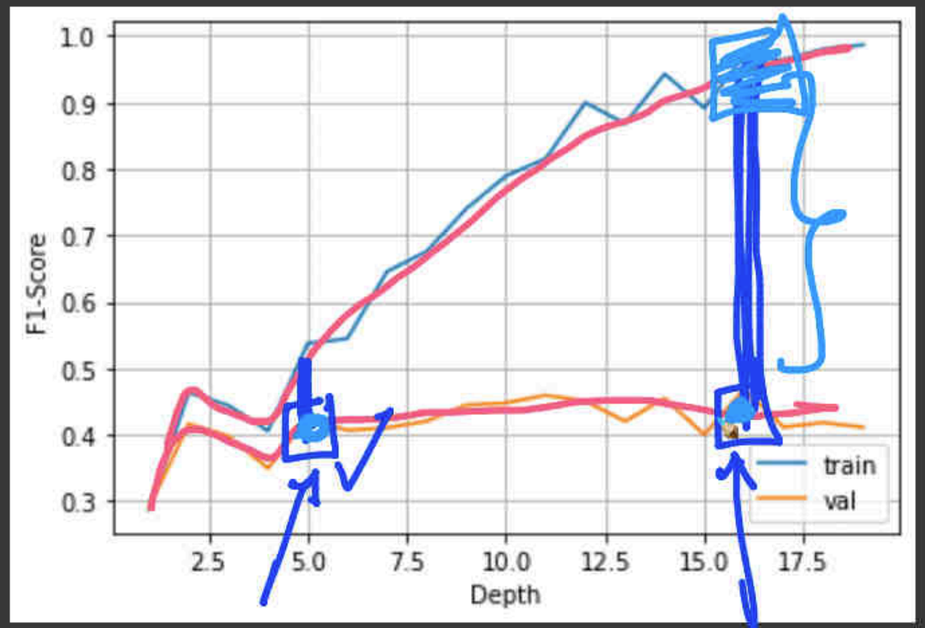

```
[ ]  # minority class needs more weighting

     # Hyper-pram tuning
     from sklearn.linear_model import LogisticRegression
     from sklearn.pipeline import make_pipeline
     from sklearn.metrics import f1_score
```

```
plt.plot(list(np.arange(l,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("Depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```




```
[210] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

      best_idx = np.argmax(val_scores)
```

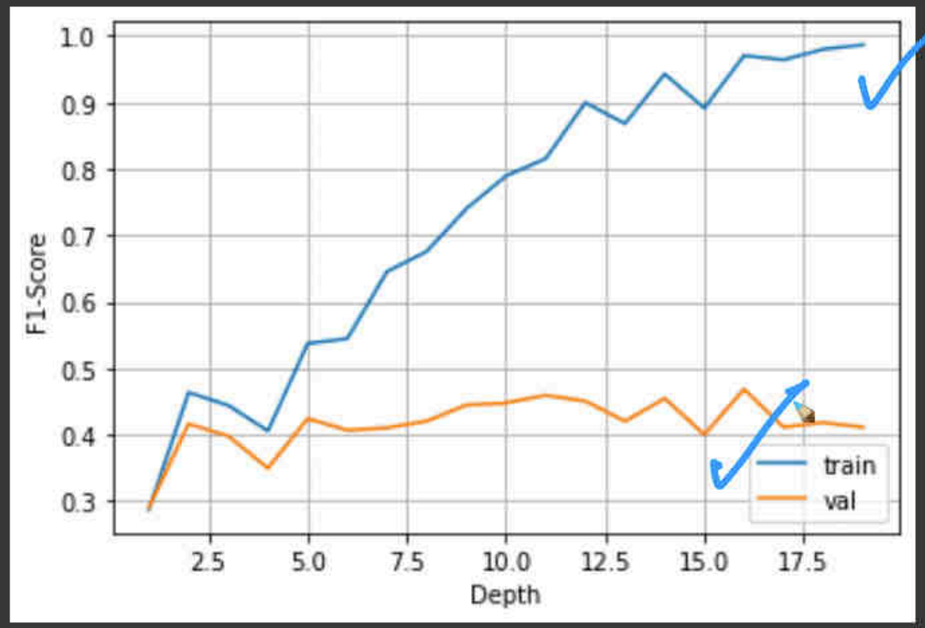
```
plt.plot(list(np.arange(1,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("Depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```



808

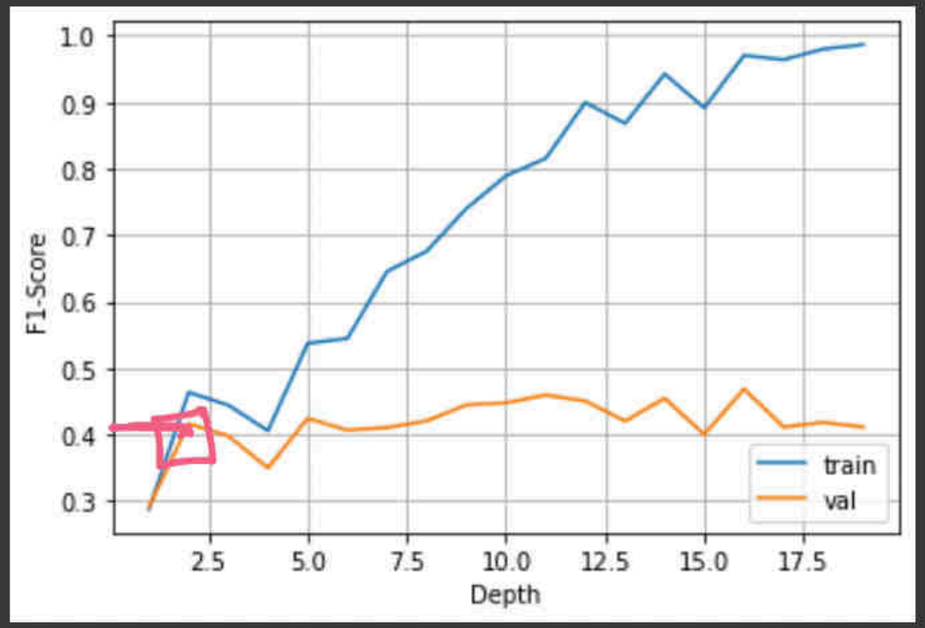
```
[210] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

      best_idx = np.argmax(val_scores)
```

```
plt.plot(list(np.arange(1,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("Depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```



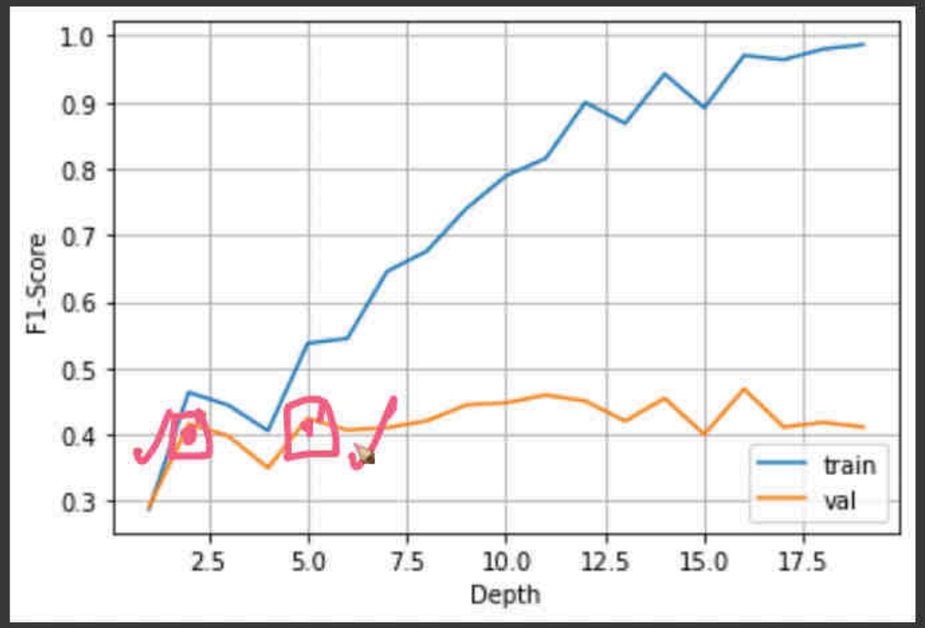

```
[210] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

      best_idx = np.argmax(val_scores)
```

```
plt.legend(loc='lower right')
plt.xlabel("Depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```

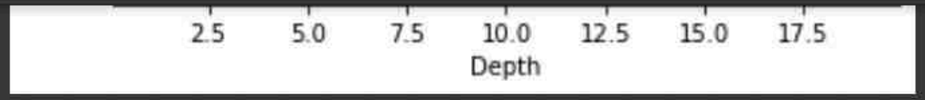



```
[210] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

      best_idx = np.argmax(val_scores)
```

```
plt.legend(loc='lower right')
plt.xlabel("Depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```



[210]
```
# Model with depth_best
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

best_idx = np.argmax(val_scores)
```



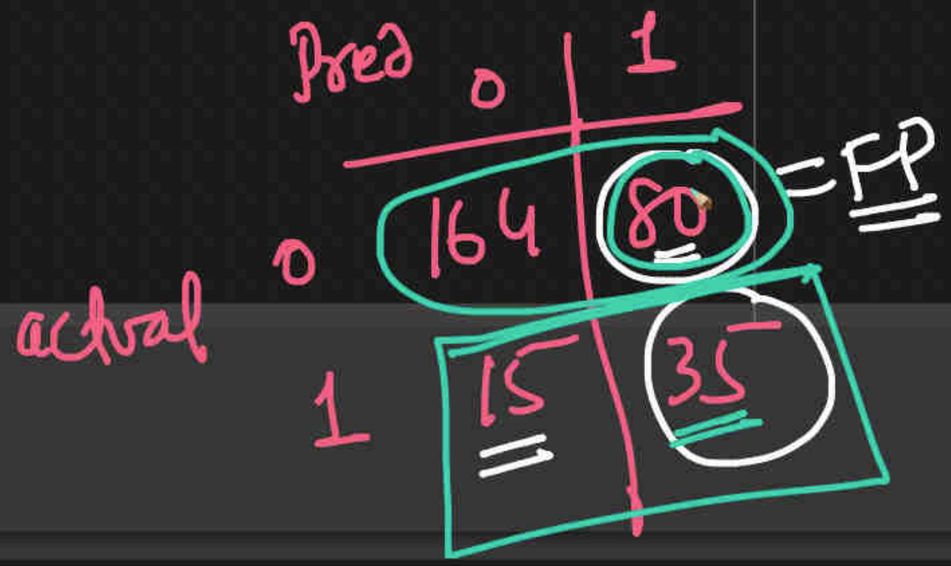

```
# Model with depth_best
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

best_idx = np.argmax(val_scores)
l_best = 5 #l+d*best_idx
print(l_best)
clf = DecisionTreeClassifier(random_state=0, max_depth=l_best, class_weight={ 0:0.1, 1:w } )
clf.fit(X_train, y_train)

y_pred_val = clf.predict(X_val)
val_score = f1_score(y_val, y_pred_val)

print(val_score)

confusion_matrix(y_val, y_pred_val)
```

```
5
0.42424242424242425
array([[164,  80],
       [ 15,  35]])
```

```
[211] # Feature importance
```

```
best_idx = np.argmax(val_scores)
l_best = 2 #l+d*best_idx
print(l_best)
clf = DecisionTreeClassifier(random_state=0, max_depth=l_best, class_weight={ 0:0.1, 1:w } )
clf.fit(X_train, y_train)

y_pred_val = clf.predict(X_val)
val_score = f1_score(y_val, y_pred_val)

print(val_score)

confusion_matrix(y_val, y_pred_val)
```

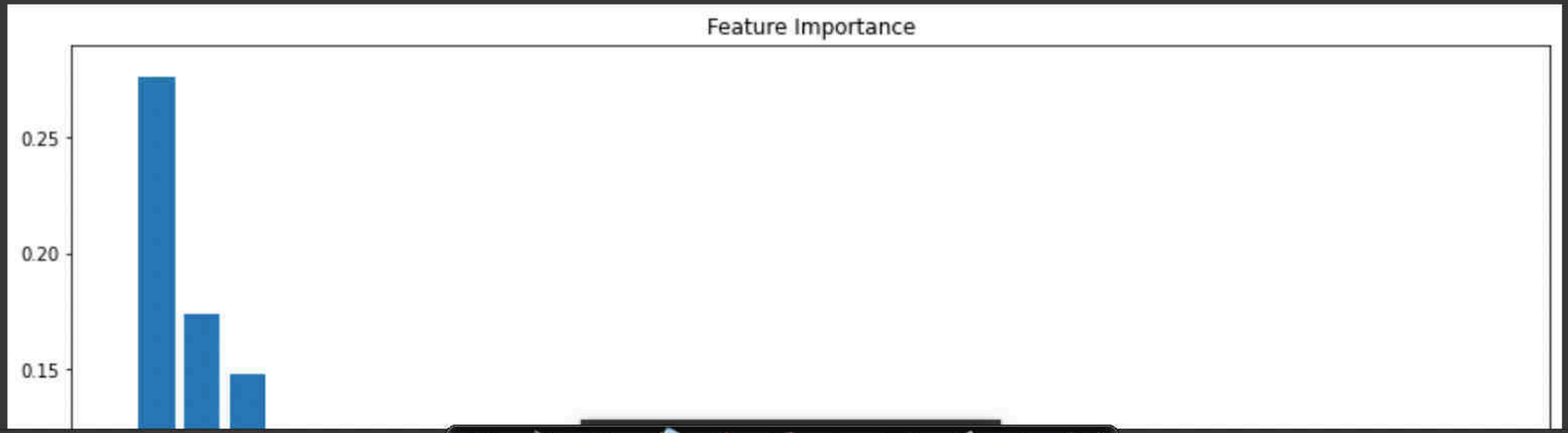
```
2
0.41618497109826585
array([[157,  87],
       [ 14,  36]])
```




```
[211] # Feature importance
      importances = clf.feature_importances_
      indices = np.argsort(importances)[::-1] # Sort feature importances in descending order
      names = [X_train.columns[i] for i in indices] # Rearrange feature names so they match the sorted feature importance
      plt.figure(figsize=(15, 7)) # Create plot
      plt.title("Feature Importance") # Create plot title
```

```
array([[157,  87],
       [ 14,  36]])
```

```python
# Feature importance
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1] # Sort feature importances in descending order
names = [X_train.columns[i] for i in indices] # Rearrange feature names so they match the sorted feature importance
plt.figure(figsize=(15, 7)) # Create plot
plt.title("Feature Importance") # Create plot title
plt.bar(range(X_train.shape[1]), importances[indices]) # Add bars
plt.xticks(range(X_train.shape[1]), names, rotation=90) # Add feature names as x-axis labels
plt.show() # Show plot
```



scikit learn
Prev Up Next
scikit-learn 1.1.0
Other versions
Please cite us if you use the software.
sklearn.tree.DecisionTreeClassifier
Examples using sklearn.tree.DecisionTreeClassifier

ccp_alpha : *non-negative float, default=0.0*

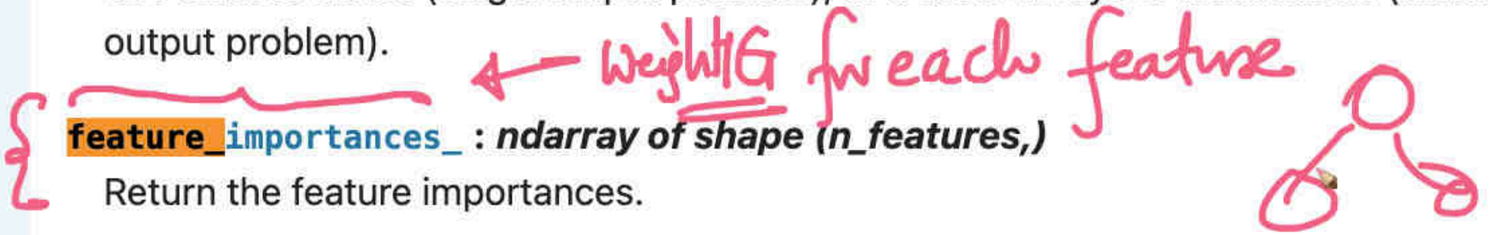
Complexity parameter used for Minimal Cost-Complexity Pruning. The subtree with the largest cost complexity that is smaller than `ccp_alpha` will be chosen. By default, no pruning is performed. See Minimal Cost-Complexity Pruning for details.

New in version 0.22.

Attributes:

classes_ : *ndarray of shape (n_classes,) or list of ndarray*

The classes labels (single output problem), or a list of arrays of class labels (multi-output problem).



feature_importances_ : *ndarray of shape (n_features,)*

Return the feature importances.

max_features_ : *int*

The inferred value of max_features.

n_classes_ : *int or list of int*

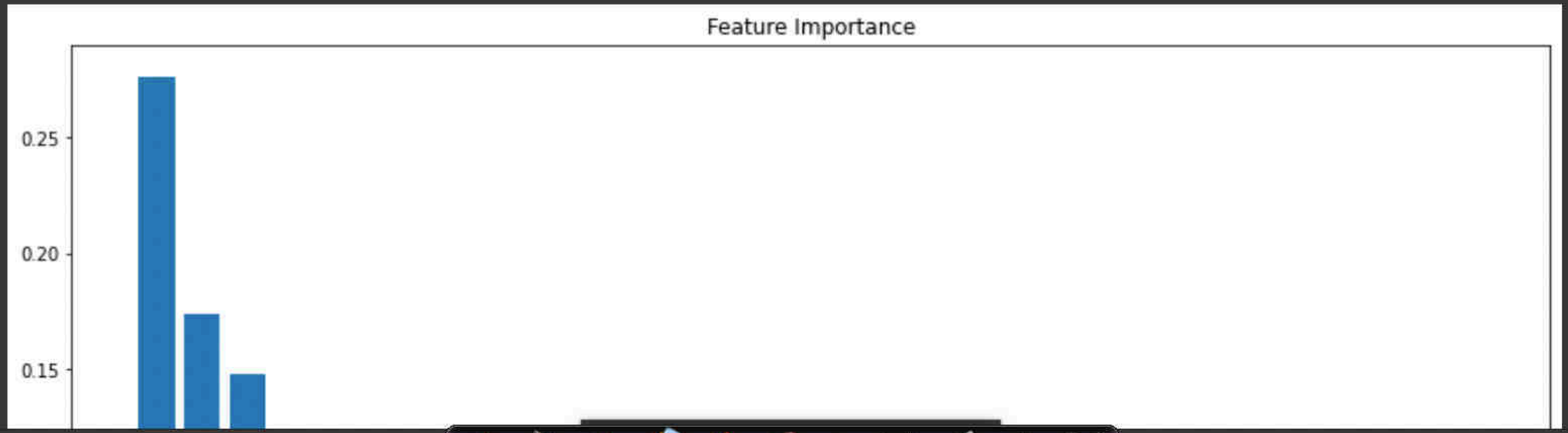
The number of classes (for single output problems), or a list containing the number of classes for each output (for multi-output problems).

n_features_ : *int*

DEPRECATED: The attribute `n_features_` is deprecated in 1.0 and will be removed in

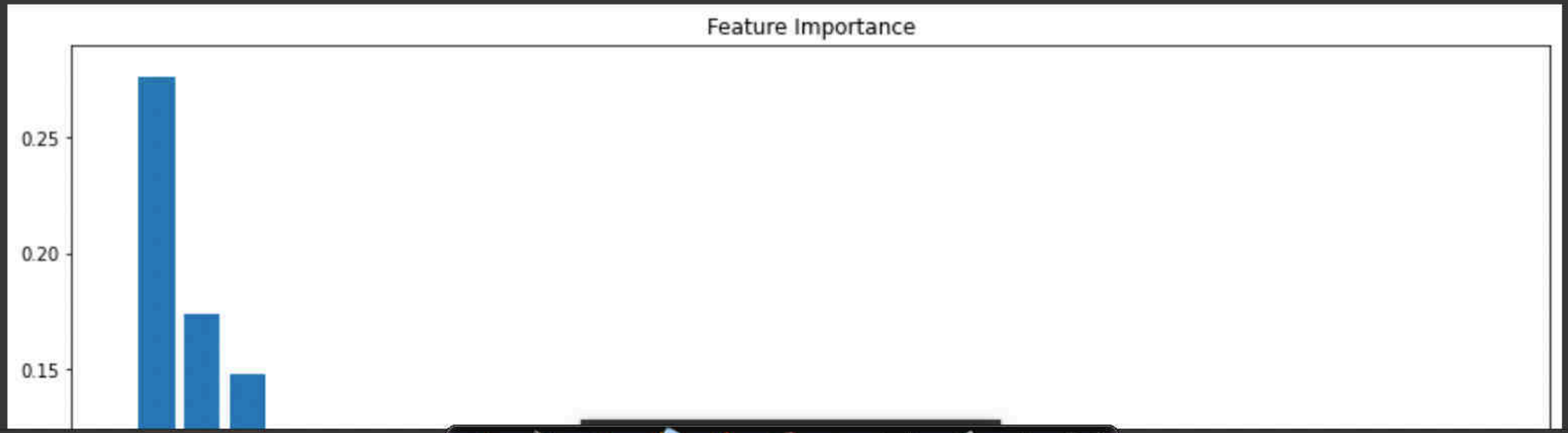
```
array([[157,  87],
       [ 14,  36]])
```

```python
# Feature importance
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1] # Sort feature importances in descending order
names = [X_train.columns[i] for i in indices] # Rearrange feature names so they match the sorted feature importance
plt.figure(figsize=(15, 7)) # Create plot
plt.title("Feature Importance") # Create plot title
plt.bar(range(X_train.shape[1]), importances[indices]) # Add bars
plt.xticks(range(X_train.shape[1]), names, rotation=90) # Add feature names as x-axis labels
plt.show() # Show plot
```



```
array([[157,  87],
       [ 14,  36]])
```

```
# Feature importance
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1] # Sort feature importances in descending order
names = [X_train.columns[i] for i in indices] # Rearrange feature names so they match the sorted feature importance
plt.figure(figsize=(15, 7)) # Create plot
plt.title("Feature Importance") # Create plot title
plt.bar(range(X_train.shape[1]), importances[indices]) # Add bars
plt.xticks(range(X_train.shape[1]), names, rotation=90) # Add feature names as x-axis labels
plt.show() # Show plot
```

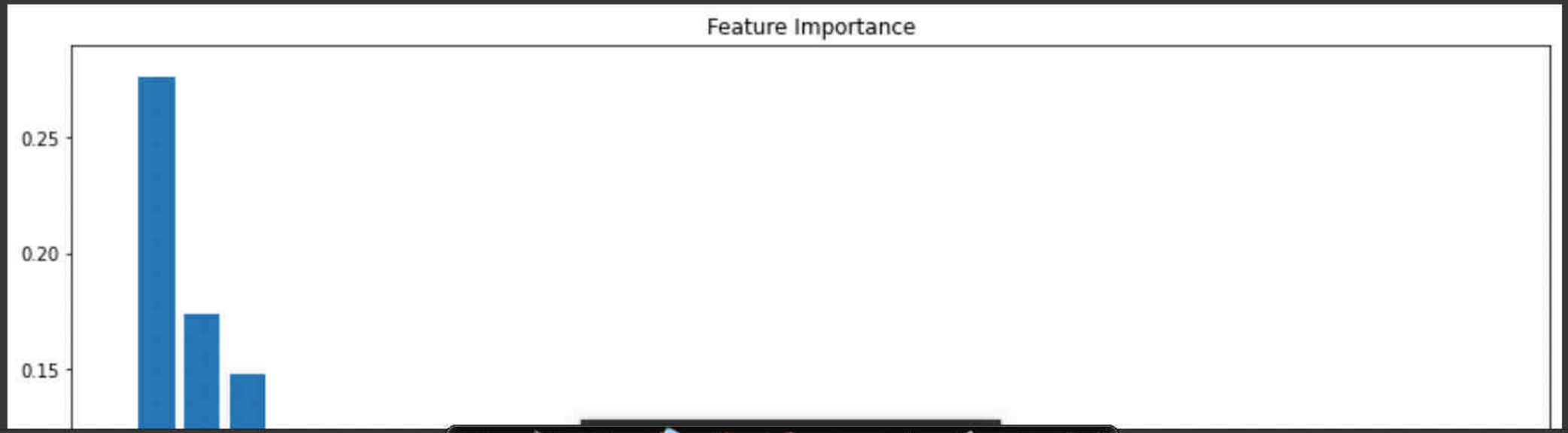


```
array([[157,  87],
       [ 14,  36]])
```

f1 f2 ... fd

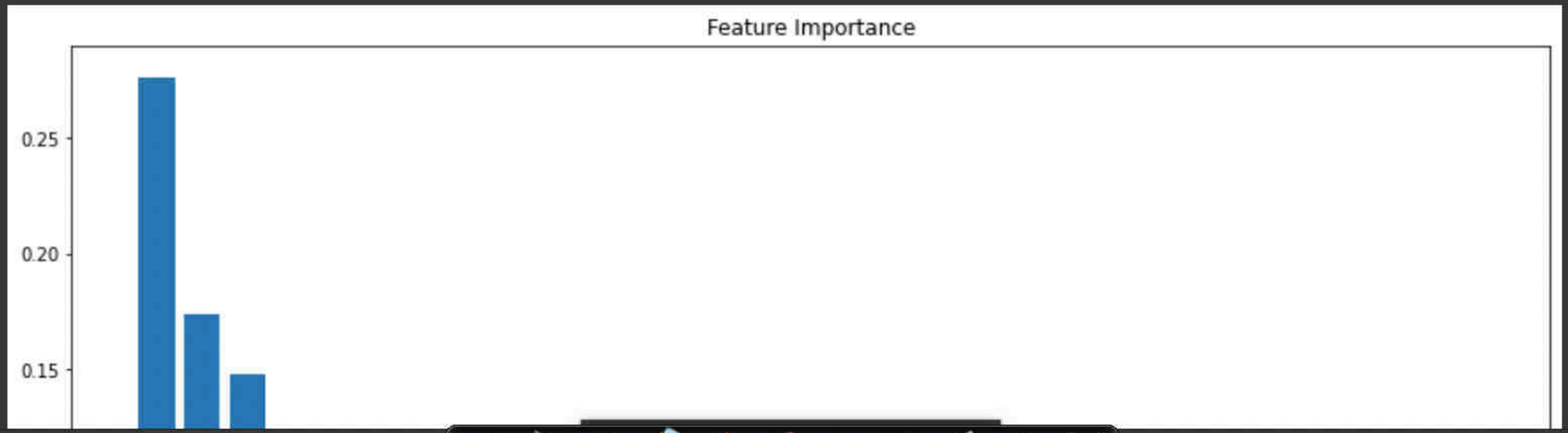
0.26 0.24 ...

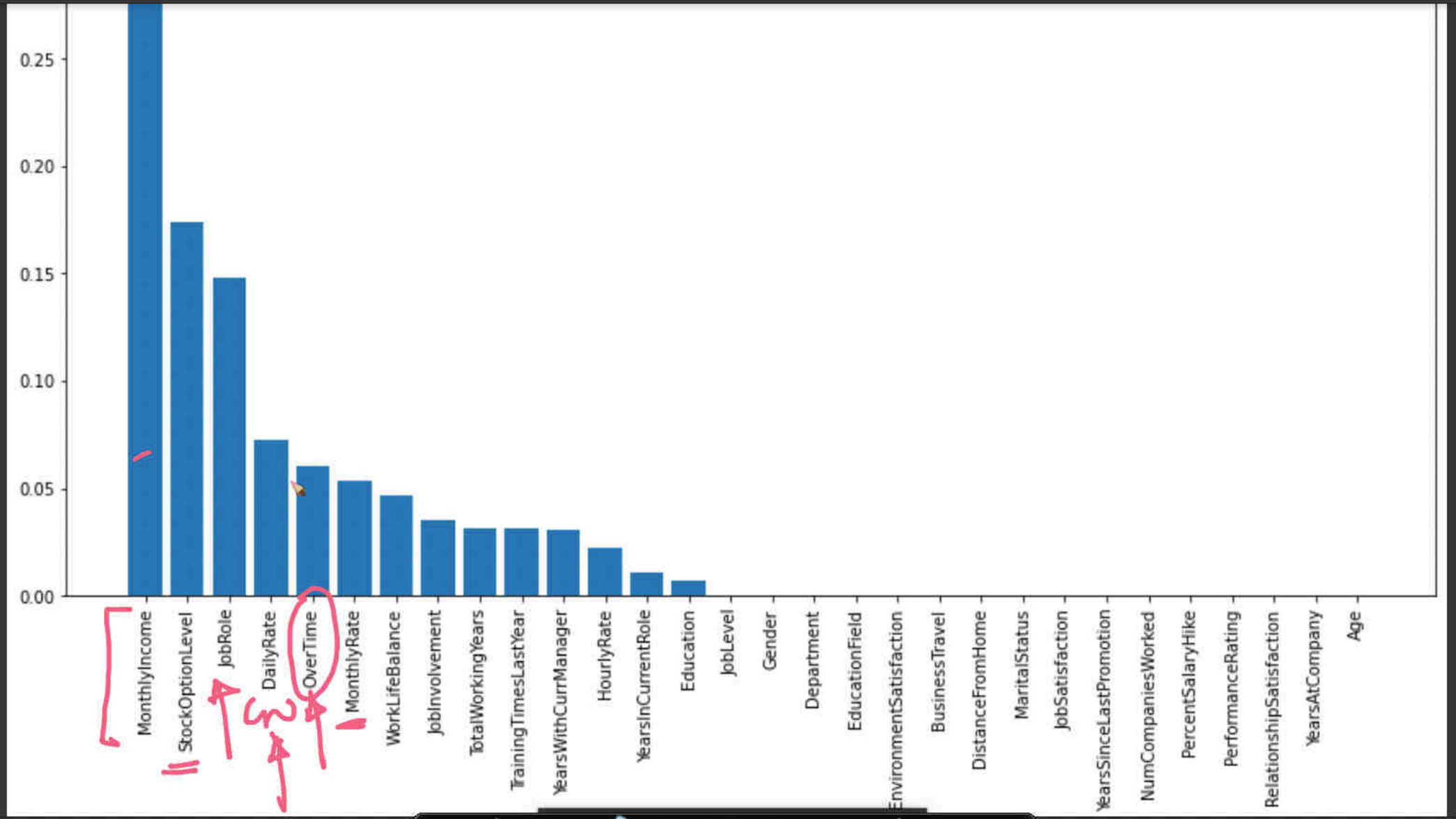
```
# Feature importance
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1] # Sort feature importances in descending order
names = [X_train.columns[i] for i in indices] # Rearrange feature names so they match the sorted feature importance
plt.figure(figsize=(15, 7)) # Create plot
plt.title("Feature Importance") # Create plot title
plt.bar(range(X_train.shape[1]), importances[indices]) # Add bars
plt.xticks(range(X_train.shape[1]), names, rotation=90) # Add feature names as x-axis labels
plt.show() # Show plot
```

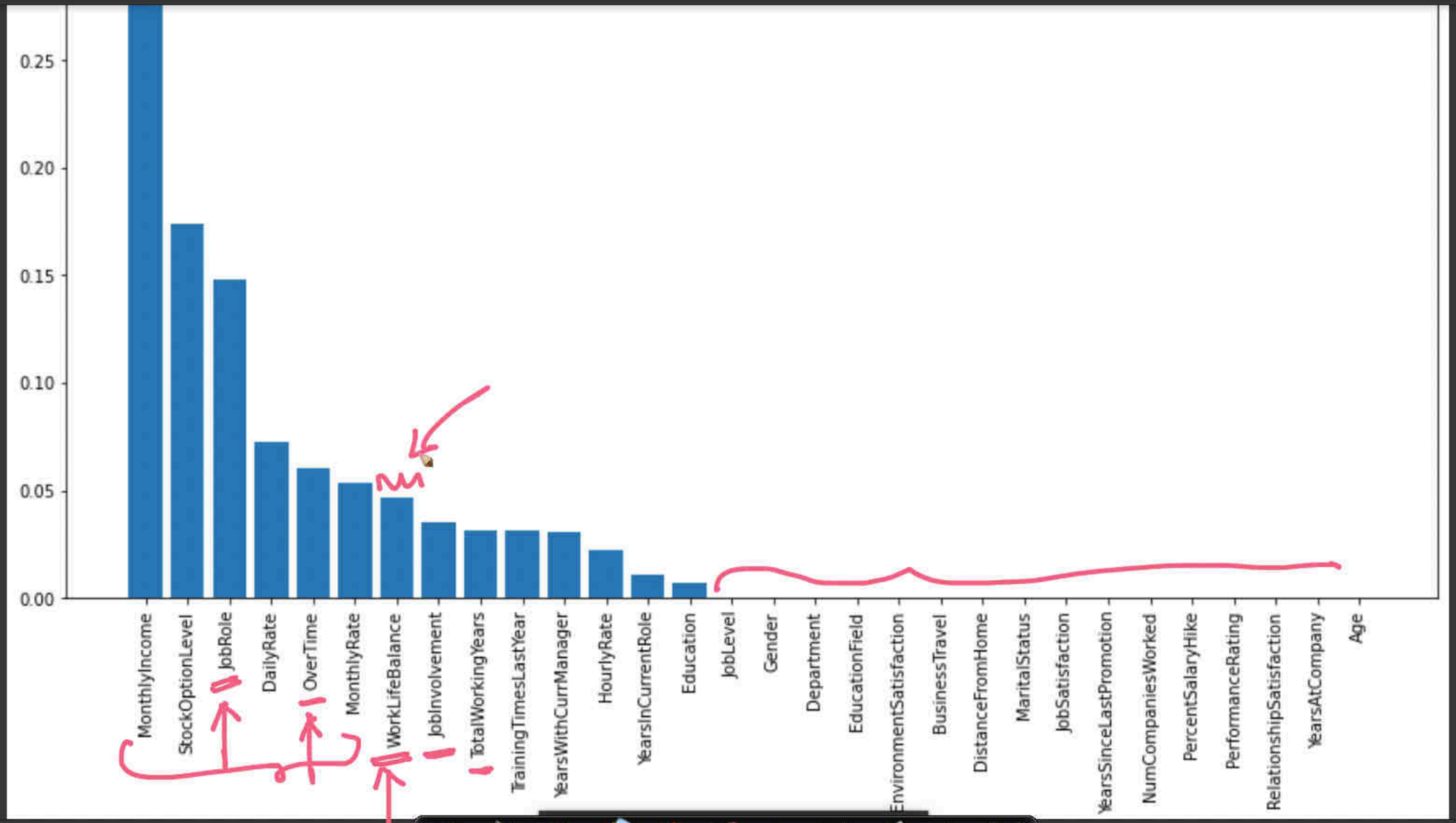


```
array([[157,  87],
       [ 14,  36]])
```

```
# Feature importance
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1] # Sort feature importances in descending order
names = [X_train.columns[i] for i in indices] # Rearrange feature names so they match the sorted feature importance
plt.figure(figsize=(15, 7)) # Create plot
plt.title("Feature Importance") # Create plot title
plt.bar(range(X_train.shape[1]), importances[indices]) # Add bars
plt.xticks(range(X_train.shape[1]), names, rotation=90) # Add feature names as x-axis labels
plt.show() # Show plot
```






0.25
0.20
0.15
0.10
0.05
0.00
MonthlyIncome
StockOptionLevel
JobRole
DailyRate
OverTime
MonthlyRate
WorkLifeBalance
JobInvolvement
TotalWorkingYears
TrainingTimesLastYear
YearsWithCurrManager
HourlyRate
YearsInCurrentRole
Education
JobLevel
Gender
Department
EducationField
EnvironmentSatisfaction
BusinessTravel
DistanceFromHome
MaritalStatus
JobSatisfaction
YearsSinceLastPromotion
NumCompaniesWorked
PercentSalaryHike
PerformanceRating
RelationshipSatisfaction
YearsAtCompany
Age

Chrome File Edit View History Bookmarks Profiles Tab Window Help 10:19

DT and RF.ipynb - Colaborator | sklearn.tree.DecisionTreeClass | sklearn.ensemble.RandomFore | sklearn.preprocessing.LabelEr | Target Encoder — Category En | Imbalanced Data.ipynb - Colab

colab.research.google.com/drive/1d2HJ9AKulf83i7pwmDrL-2FOCncFubRF#scrollTo=xl_3aNxqqkuu

Update

+ Code + Text

RAM Disk

[216]
```
train_scores = []
val_scores = []

l=1
u=20
d=1
w=3.0
num_learners=100
row_sampling_rate = 0.75

for depth in np.arange(l,u,d):
  clf = RandomForestClassifier(max_depth=depth, max_samples=row_sampling_rate, n_estimators=num_learners, random_st
  clf.fit(X_train, y_train)
  train_y_pred = clf.predict(X_train)
  val_y_pred = clf.predict(X_val)
  train_score = f1_score(y_train, train_y_pred)
  val_score = f1_score(y_val, val_y_pred)
  train_scores.append(train_score)
  val_scores.append(val_score)
```

M1

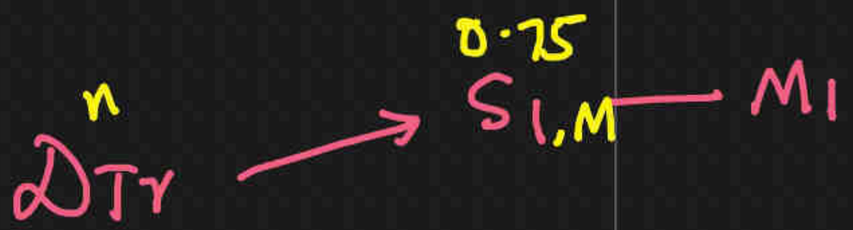
0.1 3.0

0.1 1.0

[217]
```
import matplotlib.pyplot as plt

plt.figure()
```

80 / 82

```
[216] train_scores = []
      val_scores = []

      l=1
      u=20
      d=1
      w=3.0
      num_learners=100
      row_sampling_rate = 0.75


      for depth in np.arange(l,u,d):
        clf = RandomForestClassifier(max_depth=depth, max_samples=row_sampling_rate, n_estimators=num_learners, random_st
        clf.fit(X_train, y_train)
        train_y_pred = clf.predict(X_train)
        val_y_pred = clf.predict(X_val)
        train_score = f1_score(y_train, train_y_pred)
        val_score = f1_score(y_val, val_y_pred)
        train_scores.append(train_score)
        val_scores.append(val_score)
```




```
[217] import matplotlib.pyplot as plt

      plt.figure()
```

```
[216] train_scores = []
      val_scores = []

      l=1
      u=20
      d=1
      w=3.0
      num_learners=100
      row_sampling_rate = 0.75


      for depth in np.arange(l,u,d):
        clf = RandomForestClassifier(max_depth=depth, max_samples=row_sampling_rate, n_estimators=num_learners, random_st
        clf.fit(X_train, y_train)
        train_y_pred = clf.predict(X_train)
        val_y_pred = clf.predict(X_val)
        train_score = f1_score(y_train, train_y_pred)
        val_score = f1_score(y_val, val_y_pred)
        train_scores.append(train_score)
        val_scores.append(val_score)
```

Xg boost

```
[217] import matplotlib.pyplot as plt

      plt.figure()
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help 10:21

DT and RF.ipynb - Colaborator | sklearn.tree.DecisionTreeClass | sklearn.ensemble.RandomFore | sklearn.preprocessing.LabelEn | Target Encoder — Category En | Imbalanced Data.ipynb - Colab

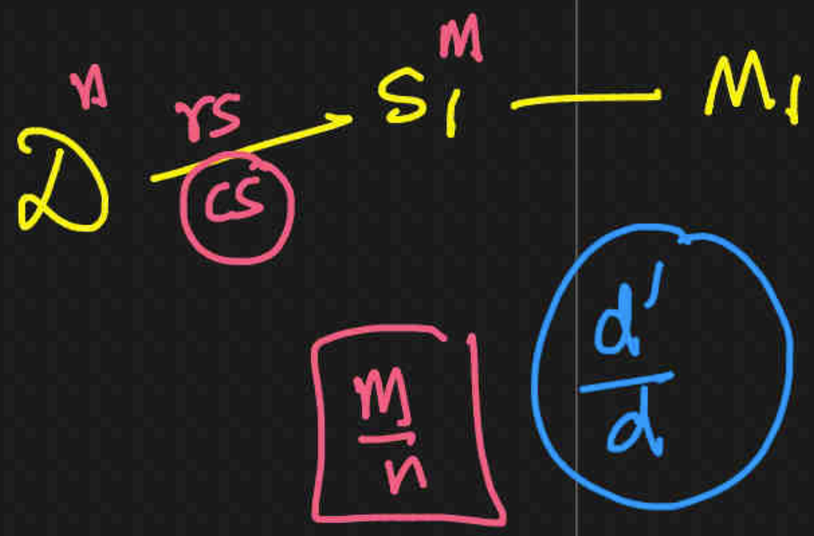
colab.research.google.com/drive/1d2HJ9AKulf83i7pwmDrL-2FOCncFubRF#scrollTo=xl_3aNxqqkuu

Update

+ Code + Text

RAM Disk

```
[216] train_scores = []
      val_scores = []

      l=1
      u=20
      d=1
      w=3.0
      num_learners=100
      row_sampling_rate = 0.75


      for depth in np.arange(l,u,d):
        clf = RandomForestClassifier(max_depth=depth, max_samples=row_sampling_rate, n_estimators=num_learners, random_st
        clf.fit(X_train, y_train)
        train_y_pred = clf.predict(X_train)
        val_y_pred = clf.predict(X_val)
        train_score = f1_score(y_train, train_y_pred)
        val_score = f1_score(y_val, val_y_pred)
        train_scores.append(train_score)
        val_scores.append(val_score)
```




```
[217] import matplotlib.pyplot as plt

      plt.figure()
```

83 / 85

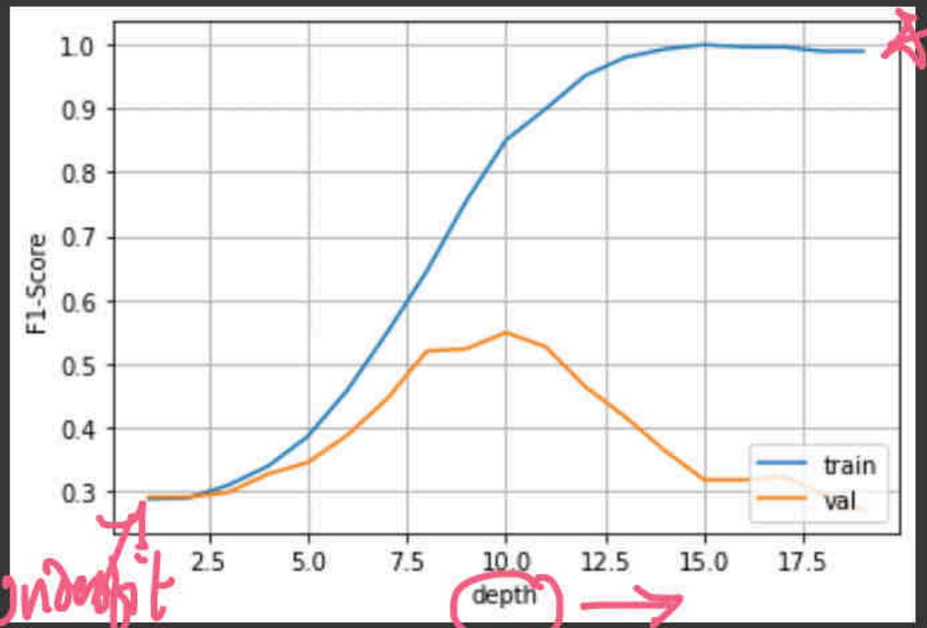
```
train_scores = []
val_scores = []

l=1
u=20
d=1
w=3.0
num_learners=100
row_sampling_rate = 0.75


for depth in np.arange(l,u,d):
  clf = RandomForestClassifier(max_depth=depth, max_samples=row_sampling_rate, n_estimators=num_learners, random_st
  clf.fit(X_train, y_train)
  train_y_pred = clf.predict(X_train)
  val_y_pred = clf.predict(X_val)
  train_score = f1_score(y_train, train_y_pred)
  val_score = f1_score(y_val, val_y_pred)
  train_scores.append(train_score)
  val_scores.append(val_score)
```

```
[217] import matplotlib.pyplot as plt

      plt.figure()
```

```
train_scores = []
val_scores = []

l=1
u=20
d=1
w=3.0
num_learners=100
row_sampling_rate = 0.75


for depth in np.arange(l,u,d):
  clf = RandomForestClassifier(max_depth=depth, max_samples=row_sampling_rate, n_estimators=num_learners, random_st
  clf.fit(X_train, y_train)
  train_y_pred = clf.predict(X_train)
  val_y_pred = clf.predict(X_val)
  train_score = f1_score(y_train, train_y_pred)
  val_score = f1_score(y_val, val_y_pred)
  train_scores.append(train_score)
  val_scores.append(val_score)
```

```
[217] import matplotlib.pyplot as plt

      plt.figure()
```

```
plt.figure()
plt.plot(list(np.arange(l,u,d)), train_scores, label="train")
plt.plot(list(np.arange(l,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```



overfit

Underfit

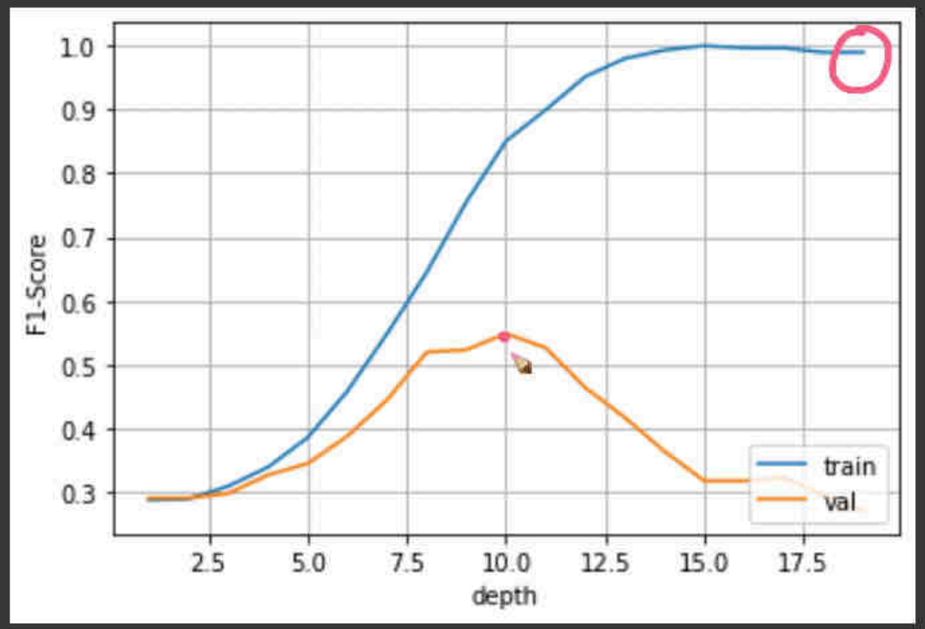
depth ↑

m/n ↓

#base learners ↑

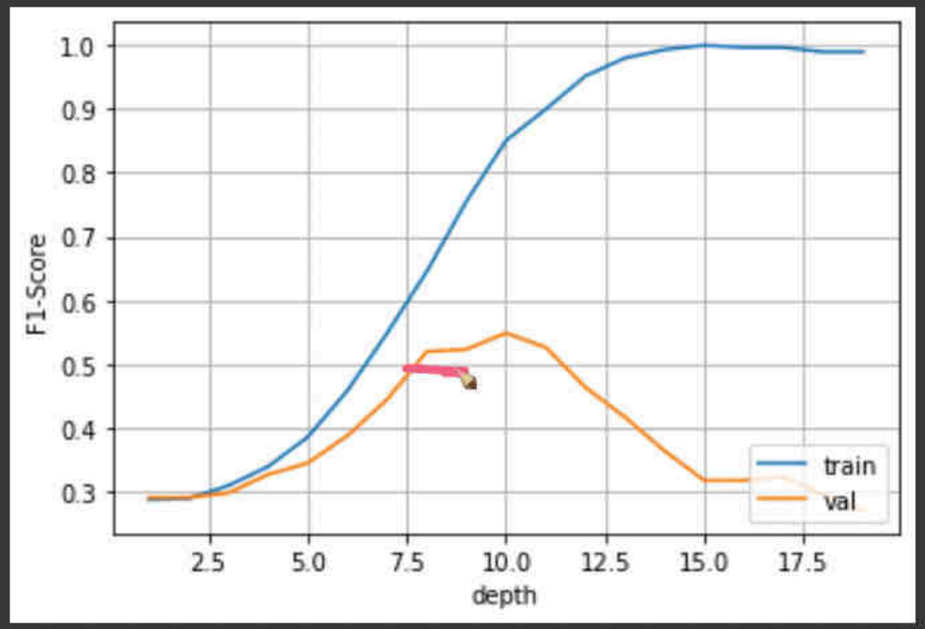
```
[218] # Model with depth best
```

```
plt.figure()
plt.plot(list(np.arange(1,u,d)), train_scores, label="train")
plt.plot(list(np.arange(1,u,d)), val_scores, label="val")
plt.legend(loc='lower right')
plt.xlabel("depth")
plt.ylabel("F1-Score")
plt.grid()
plt.show()
```



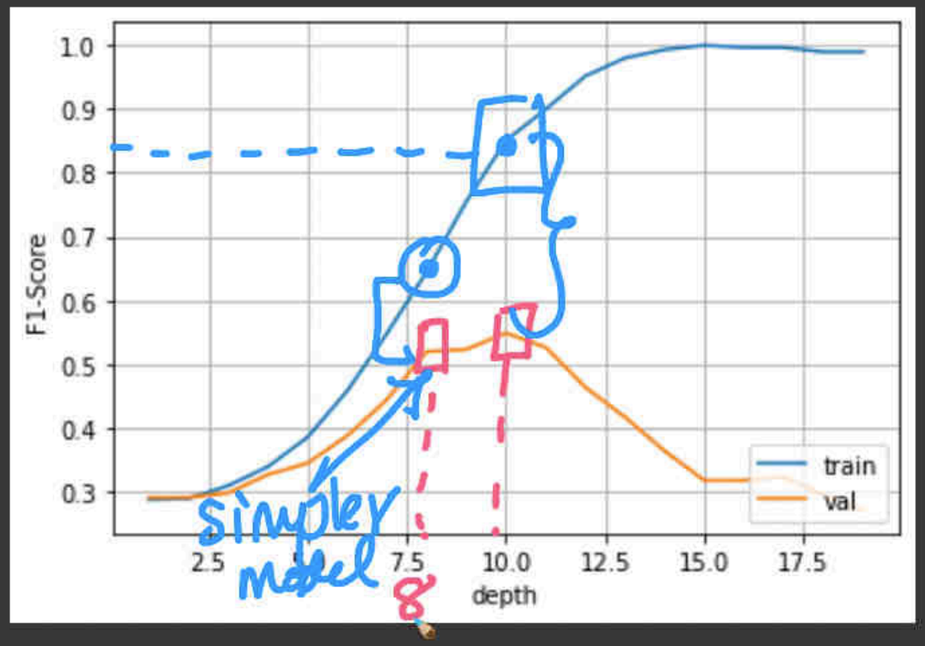

```
[218] # Model with depth best
```

```
[242] plt.plot(list(np.arange(1,u,d)), train_scores, label="train")
      plt.plot(list(np.arange(1,u,d)), val_scores, label="val")
      plt.legend(loc='lower right')
      plt.xlabel("depth")
      plt.ylabel("F1-Score")
      plt.grid()
      plt.show()
```



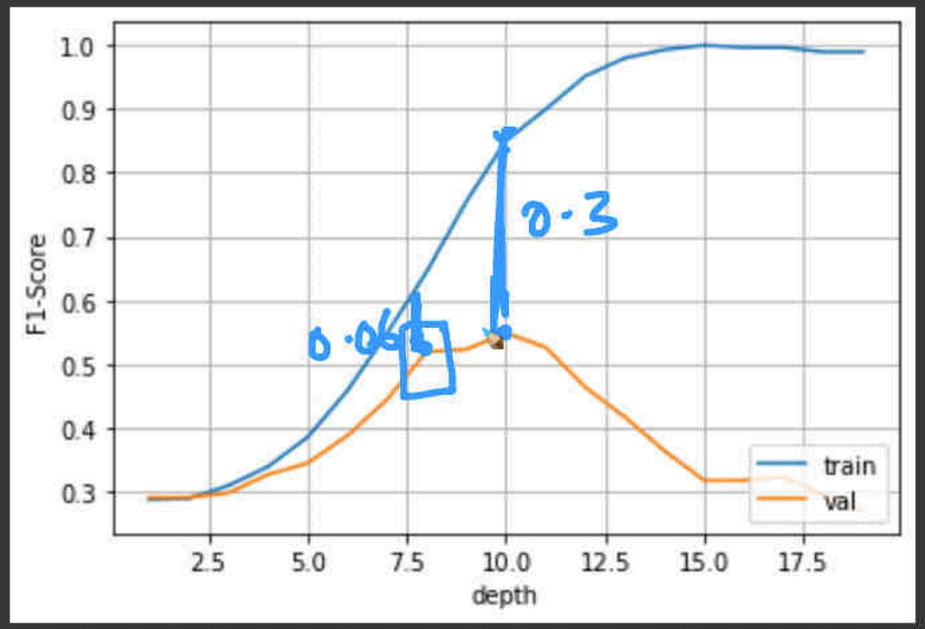

```
[218] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re
```

```
[242] plt.plot(list(np.arange(1,u,d)), val_scores, label="val")
      plt.legend(loc='lower right')
      plt.xlabel("depth")
      plt.ylabel("F1-Score")
      plt.grid()
      plt.show()
```




```
[218] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re
```

```
[242] plt.plot(list(np.arange(1,u,d)), val_scores, label="val")
      plt.legend(loc='lower right')
      plt.xlabel("depth")
      plt.ylabel("F1-Score")
      plt.grid()
      plt.show()
```




```
[218] # Model with depth_best
      from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re
```

depth

```
# Model with depth_best
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, precision_re

best_idx = np.argmax(val_scores)
l_best = 8 #l+d*best_idx
clf = RandomForestClassifier(max_depth=l_best, max_samples = row_sampling_rate, n_estimators=num_learners, random_s
clf.fit(X_train, y_train)

y_pred_val = clf.predict(X_val)
val_score = f1_score(y_val, y_pred_val)

print(val_score)

confusion_matrix(y_val, y_pred_val)
```

```
0.5194805194805195
array([[180,  64],
       [ 10,  40]])
```

```
[219] # Feature importance
      importances = clf.feature_importances_
      indices = np.argsort(importances)[::-1] # Sort feature importances in descending order
      names = [X_train.columns[i] for i in indices] # Rearrange feature names so they match the sorted feature importance
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help 10:26

DT and RF.ipynb - Colaborator | sklearn.tree.DecisionTreeClass | sklearn.ensemble.RandomFore | sklearn.preprocessing.LabelEr | Target Encoder — Category En | Imbalanced Data.ipynb - Colab

colab.research.google.com/drive/1d2HJ9AKulf83i7pwmDrL-2FOCncFubRF#scrollTo=QX-DZt3bsmqh

Update

+ Code + Text

RAM Disk

[239]
0s

```
print(val_score)

confusion_matrix(y_val, y_pred_val)
```

```
2
0.41618497109826585
array([[157,  87],
       [ 14,  36]])
```

[211]
0s

```
# Feature importance
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1] # Sort feature importances in descending order
names = [X_train.columns[i] for i in indices] # Rearrange feature names so they match the sorted feature importance
plt.figure(figsize=(15, 7)) # Create plot
plt.title("Feature Importance") # Create plot title
plt.bar(range(X_train.shape[1]), importances[indices]) # Add bars
plt.xticks(range(X_train.shape[1]), names, rotation=90) # Add feature names as x-axis labels
plt.show() # Show plot
```

92 / 94

```
clf = RandomForestClassifier(max_depth=l_best, max_samples = row_sampling_rate, n_estimat
clf.fit(X_train, y_train)

y_pred_val = clf.predict(X_val)
val_score = f1_score(y_val, y_pred_val)

print(val_score)

confusion_matrix(y_val, y_pred_val)
```

```
0.5194805194805195
array([[180,  64],
       [ 10,  40]])
```

```
[219] # Feature importance
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1] # Sort feature importances in descending order
names = [X_train.columns[i] for i in indices] # Rearrange feature names so they match the sorted feature importance
plt.figure(figsize=(15, 7)) # Create plot
plt.title("Feature Importance") # Create plot title
plt.bar(range(X_train.shape[1]), importances[indices]) # Add bars
plt.xticks(range(X_train.shape[1]), names, rotation=90) # Add feature names as x-axis labels
plt.show() # Show plot
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help 10:27

DT and RF.ipynb - Colaborator | sklearn.tree.DecisionTreeClass | sklearn.ensemble.RandomFore | sklearn.preprocessing.LabelEr | Target Encoder — Category En | Imbalanced Data.ipynb - Colab

colab.research.google.com/drive/1d2HJ9AKulf83i7pwmDrL-2FOCncFubRF#scrollTo=QX-DZt3bsmqh

Update

+ Code + Text

RAM Disk

[239]
0s

```
y_pred_val = clf.predict(X_val)
val_score = f1_score(y_val, y_pred_val)

print(val_score)

confusion_matrix(y_val, y_pred_val)
```

```
2
0.41618497109826585
array([[157,  87],
       [ 14,  36]])
```

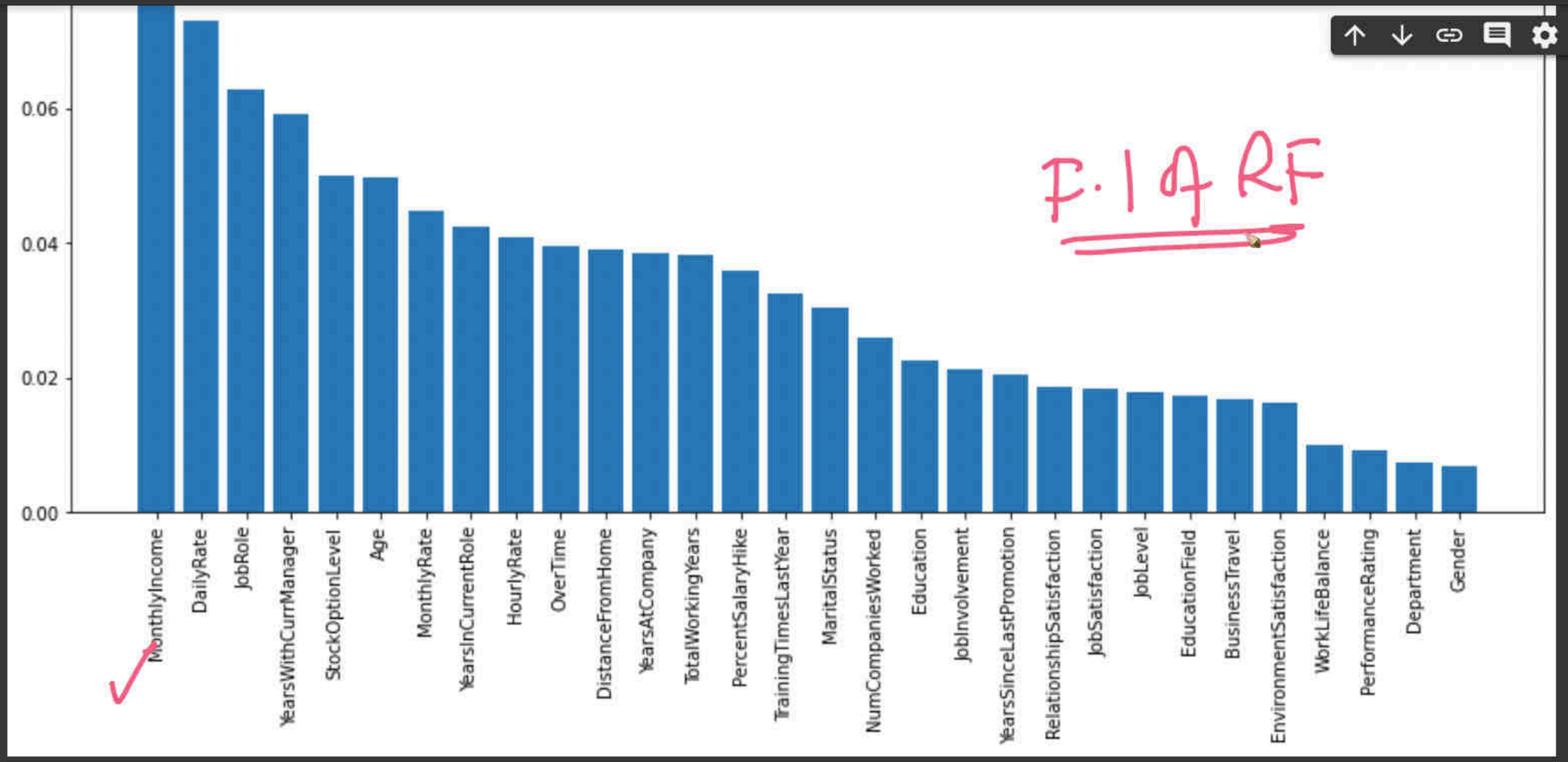
DT: 87, 14

RF: 64, 10

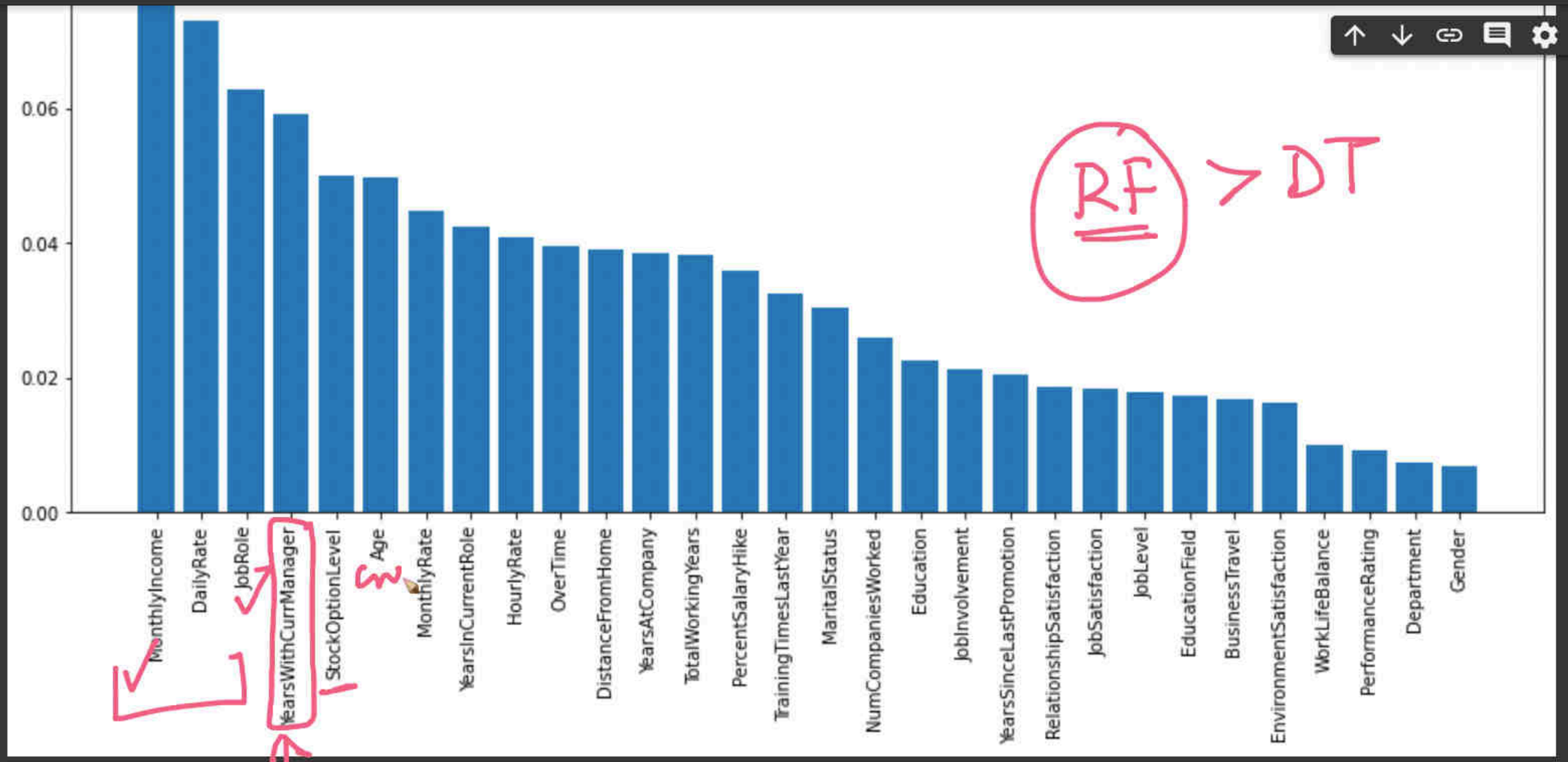
[211]
0s

```
# Feature importance
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1] # Sort feature importances in descending order
names = [X_train.columns[i] for i in indices] # Rearrange feature names so they match the sorted feature importance
plt.figure(figsize=(15, 7)) # Create plot
plt.title("Feature Importance") # Create plot title
plt.bar(range(X_train.shape[1]), importances[indices]) # Add bars
plt.xticks(range(X_train.shape[1]), names, rotation=90) # Add feature names as x-axis labels
plt.show() # Show plot
```

94 / 96



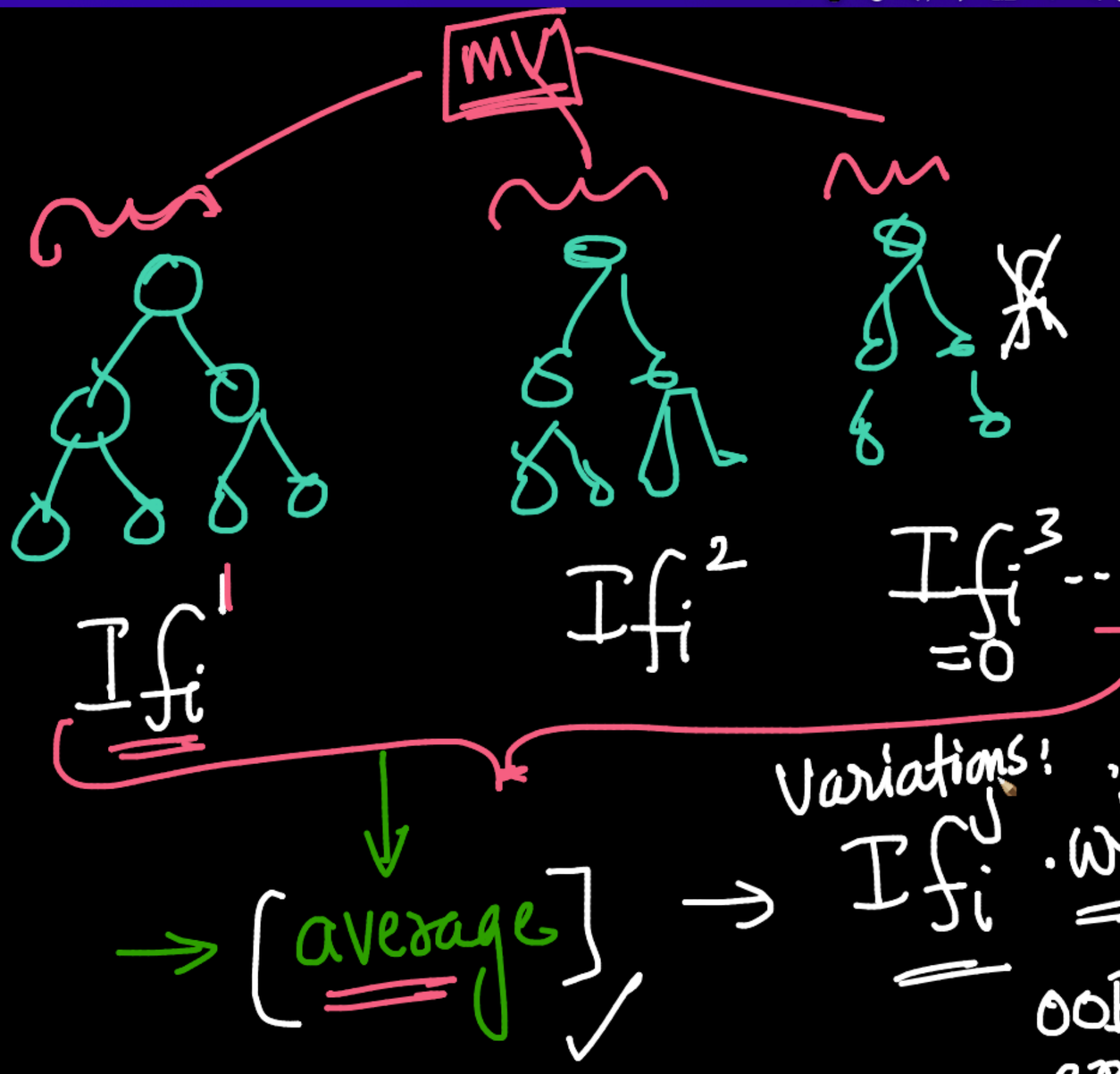
Other Concepts (covered after boosting)

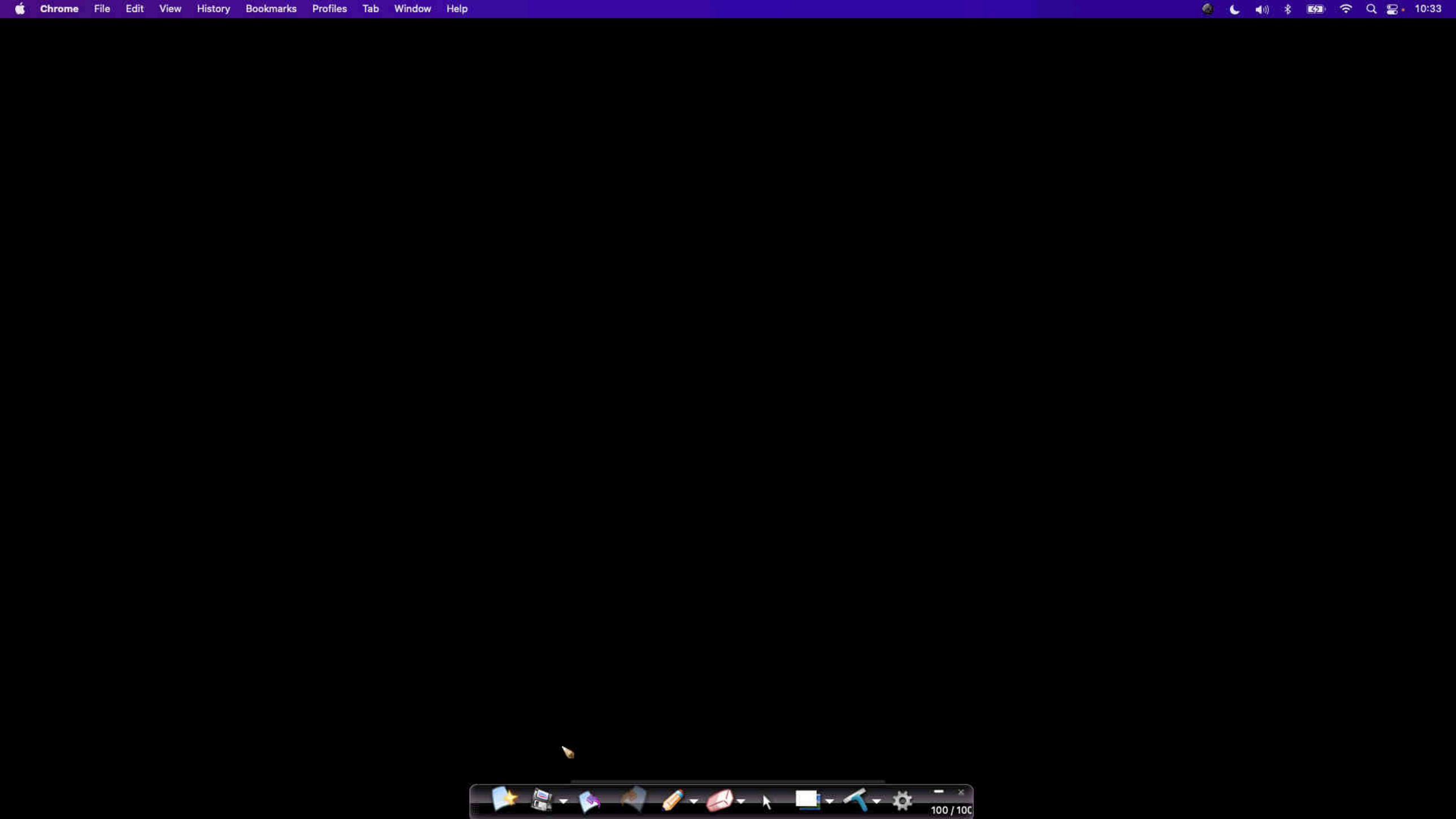


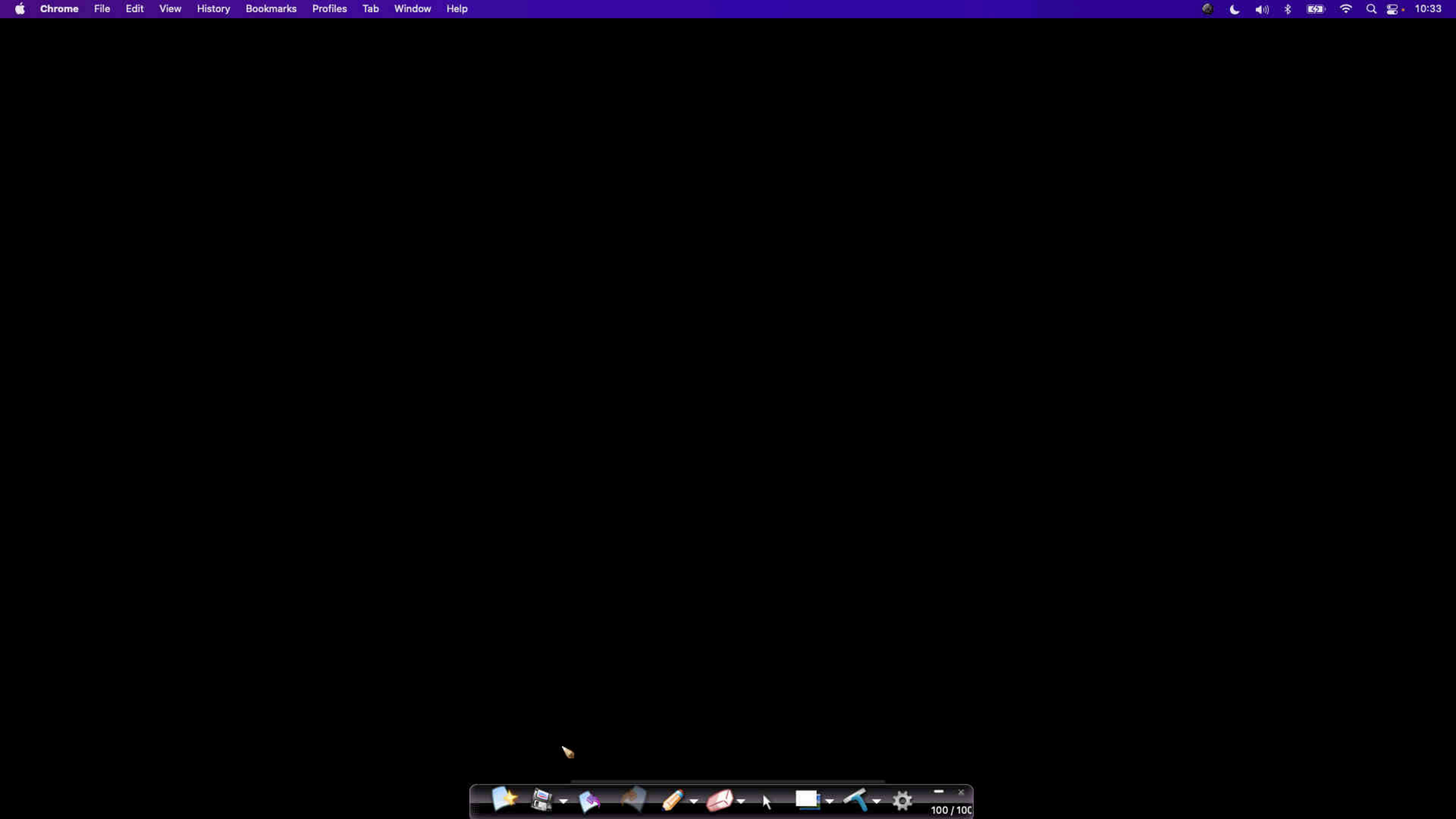
Other Concepts (covered after boosting)

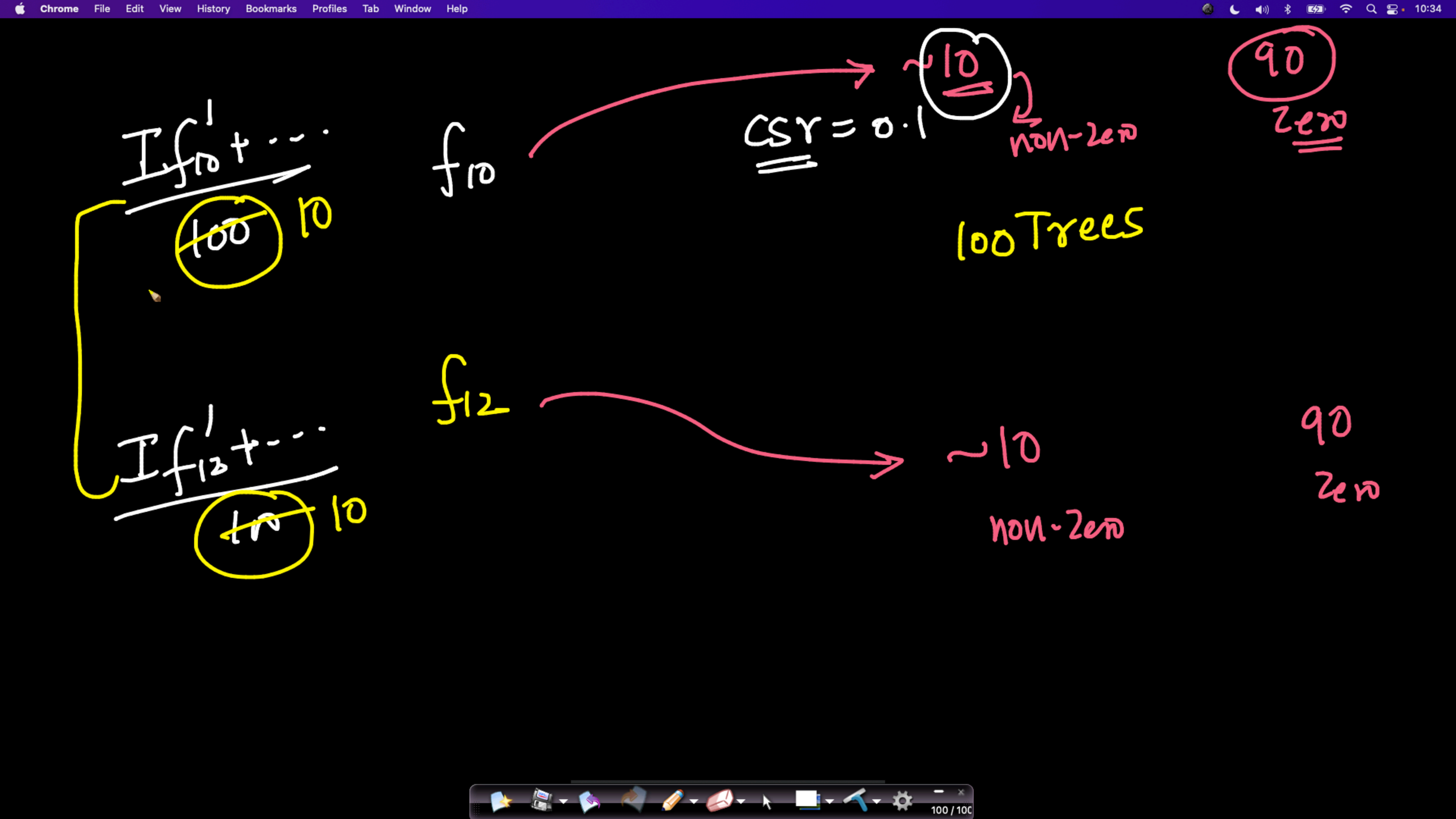
F·I in RF

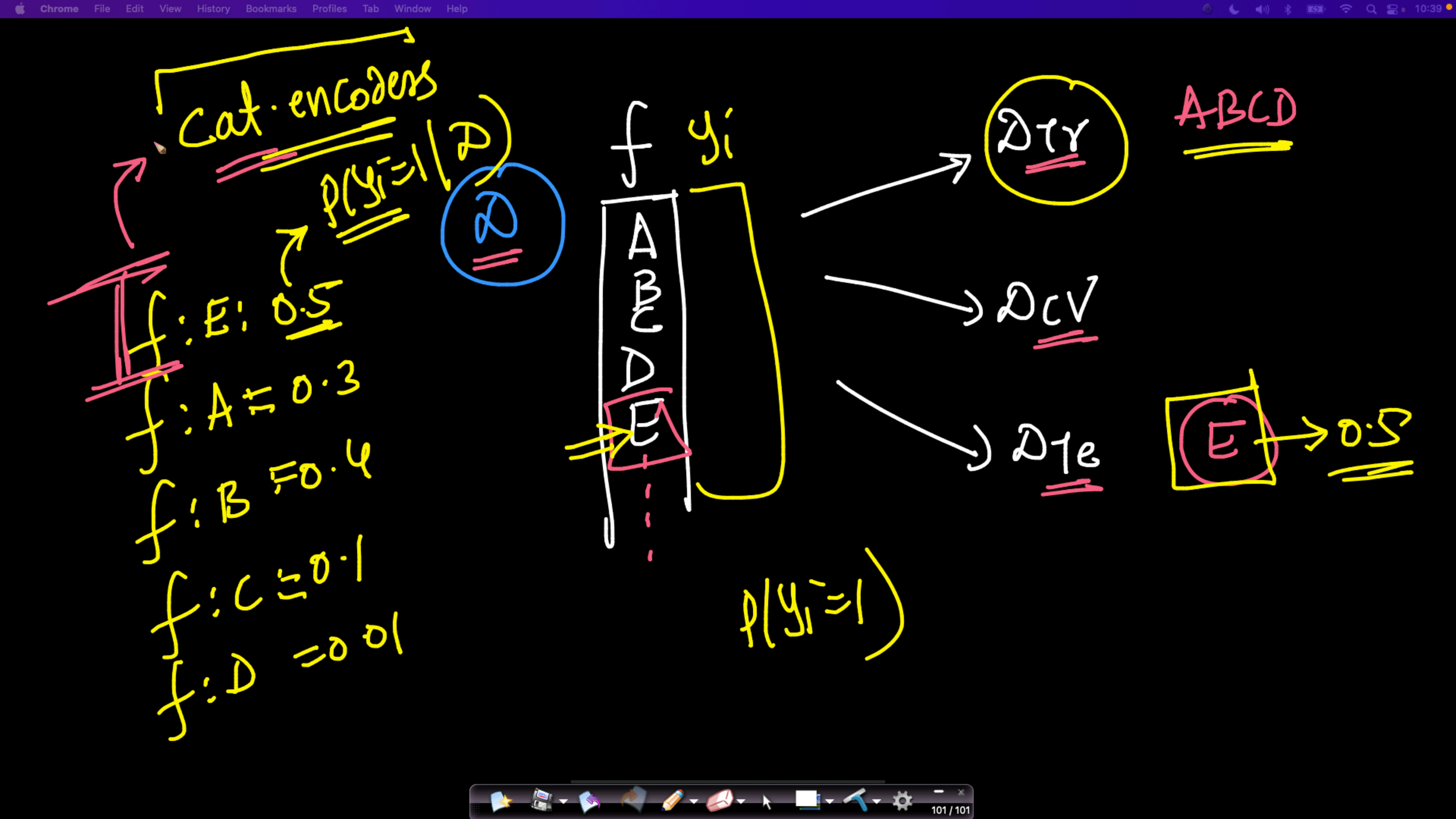
col. sampling




Chrome File Edit View History Bookmarks Profiles Tab Window Help
10:33
100 / 100


Chrome File Edit View History Bookmarks Profiles Tab Window Help
10:33
100 / 100


If f10 + ...
100
10
f10
~10
CSR = 0.1
non-zero
90
zero
100 Trees
If f12 + ...
10
f12
~10
non-zero
90
zero


Cat. encoders
P(yi=1|D)
D
f : E : 0.5
f : A = 0.3
f : B = 0.4
f : C = 0.1
f : D = 0.01
f
yi
A
B
E
D
E
D1Y
D1V
D1e
ABCD
E
0.5
P(yi=1)

fi

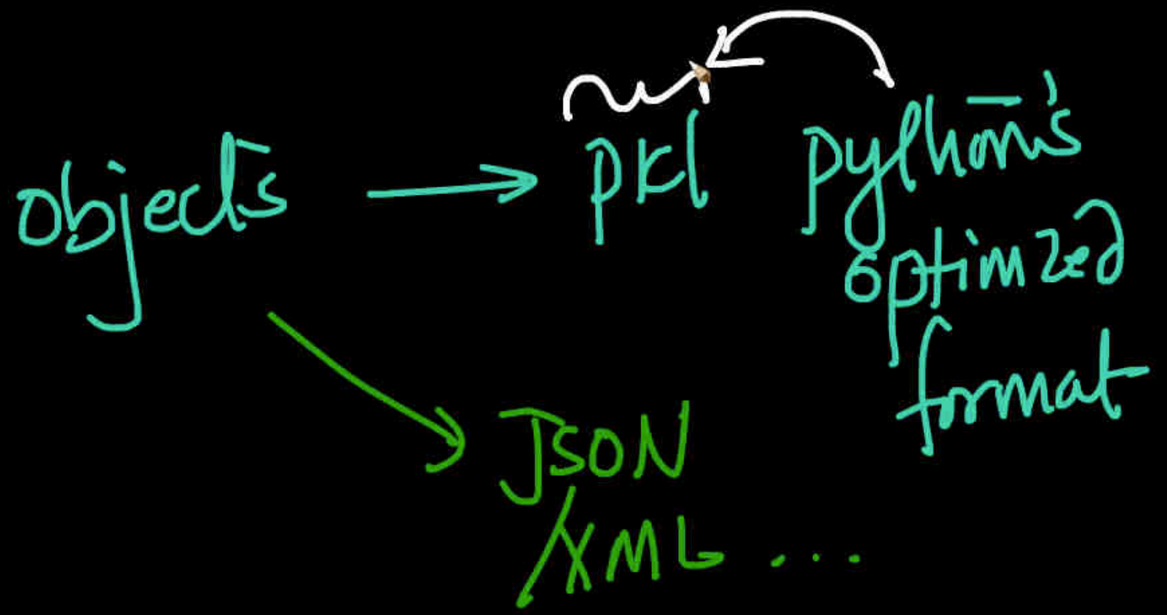
Target enloder : encoder ; - - -

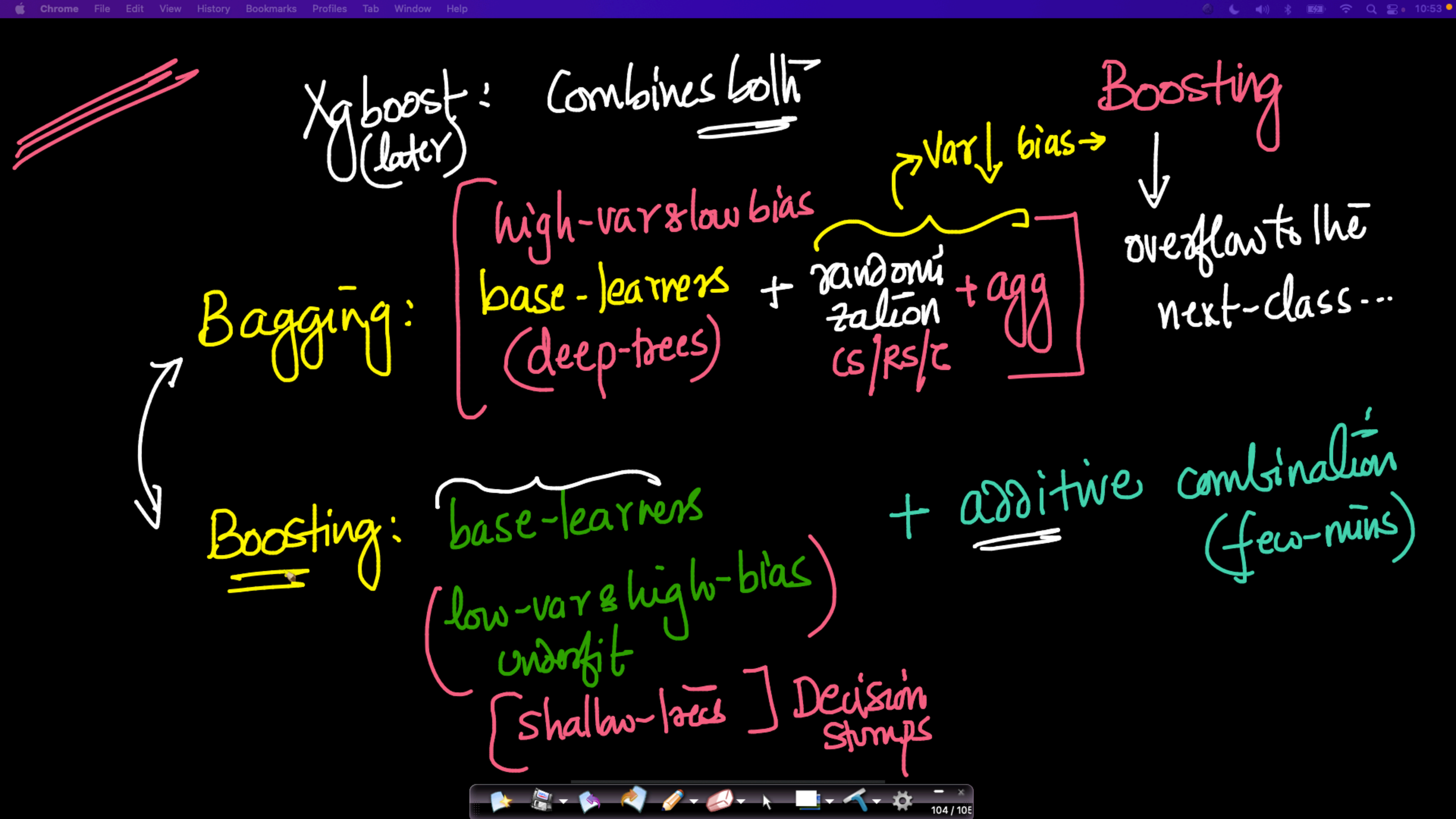
→ object

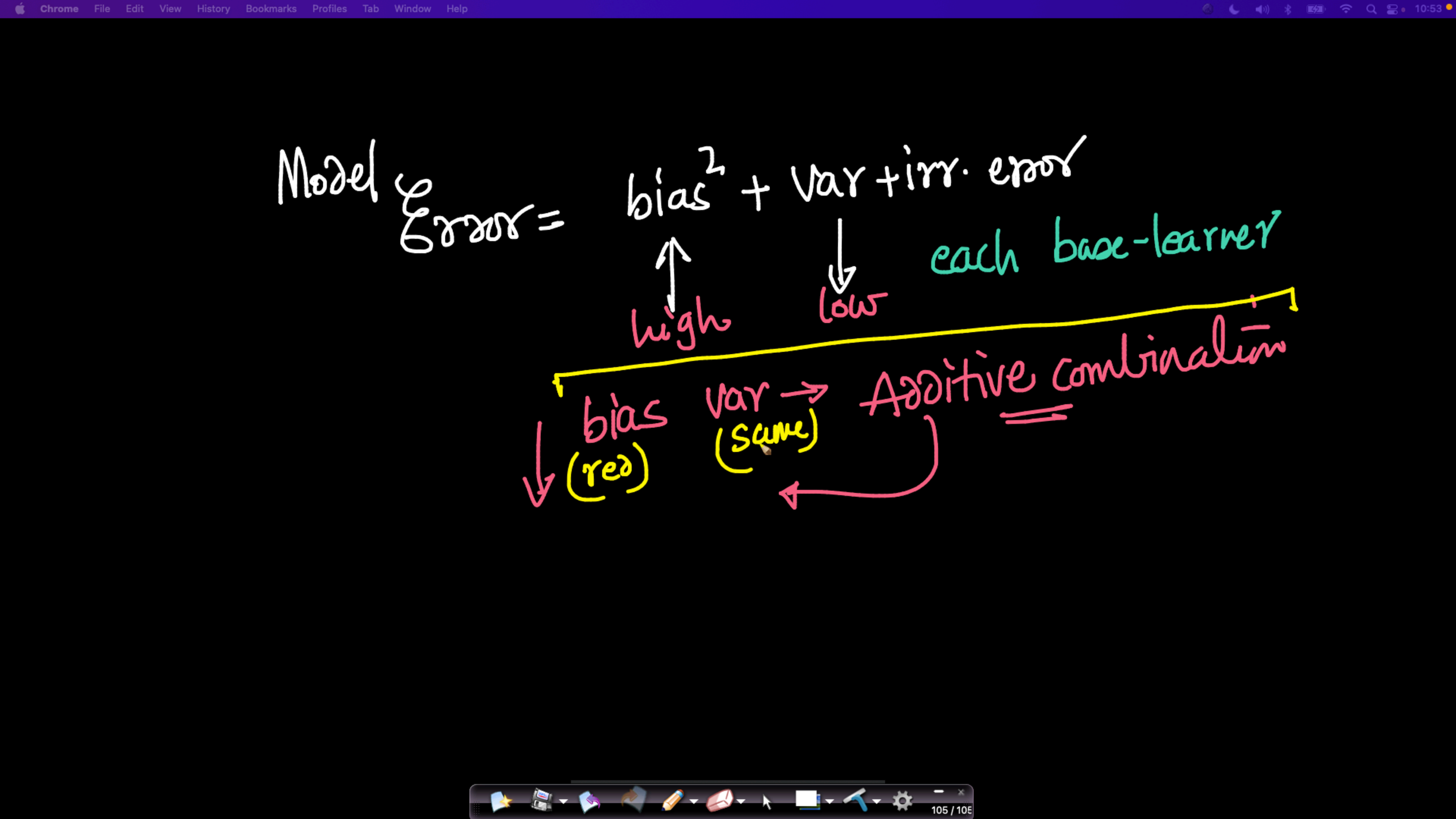
↓

{ strc it
toaisk

pid file




Xgboost: Combines both
(later)
Bagging:
high-var & low bias
base-learners
(deep-trees)
+ randomization
cs/rs/t
+ agg
Var↓ bias→
Boosting
overflow to the next-class...
Boosting:
base-learners
(low-var & high-bias)
underfit
[shallow-trees] Decision Stumps
+ additive combination
(few-runs)


Model Error = bias² + var + irr. error
high
low
each base-learner
bias (red)
var (same)
Additive combination

Chrome File Edit View History Bookmarks Profiles Tab Window Help

Core-idea: Frame-work

$$D_{tr} = \{(x_i, y_i)\}_{i=1}^{n}$$ (real)

easier to grasp

regression

Classify: post read

Stage 0 : M_0 = Mean-model

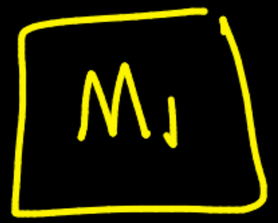
$$\hat{y}_i = \text{mean}(y_i s)$$

$h_0(x)$

$$y_i - \hat{y}_i^0 = \text{error}_i^0 \quad \forall i = 1 \to n$$

106 / 108

Stage 1:



Shallow DT (extreme: Decision Stump)

$h_1(x)$

$\{x_i, err_i\}$

y_i's

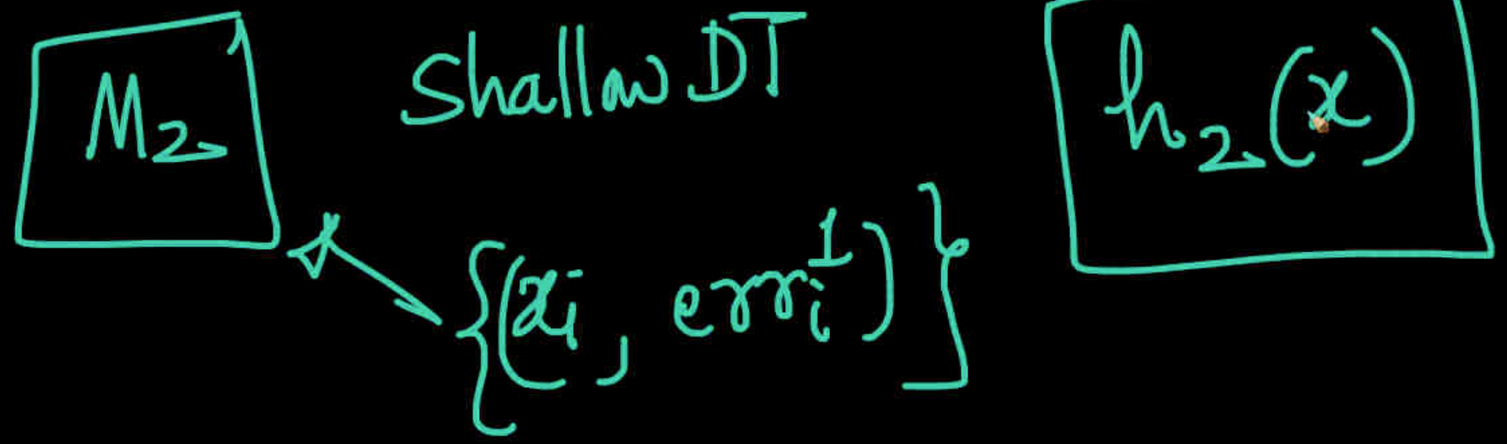
$$F_1(x) = \alpha_0 h_0(x) + \alpha_1 h_1(x)$$

model after stage 1

algo to find α_0 & α_1

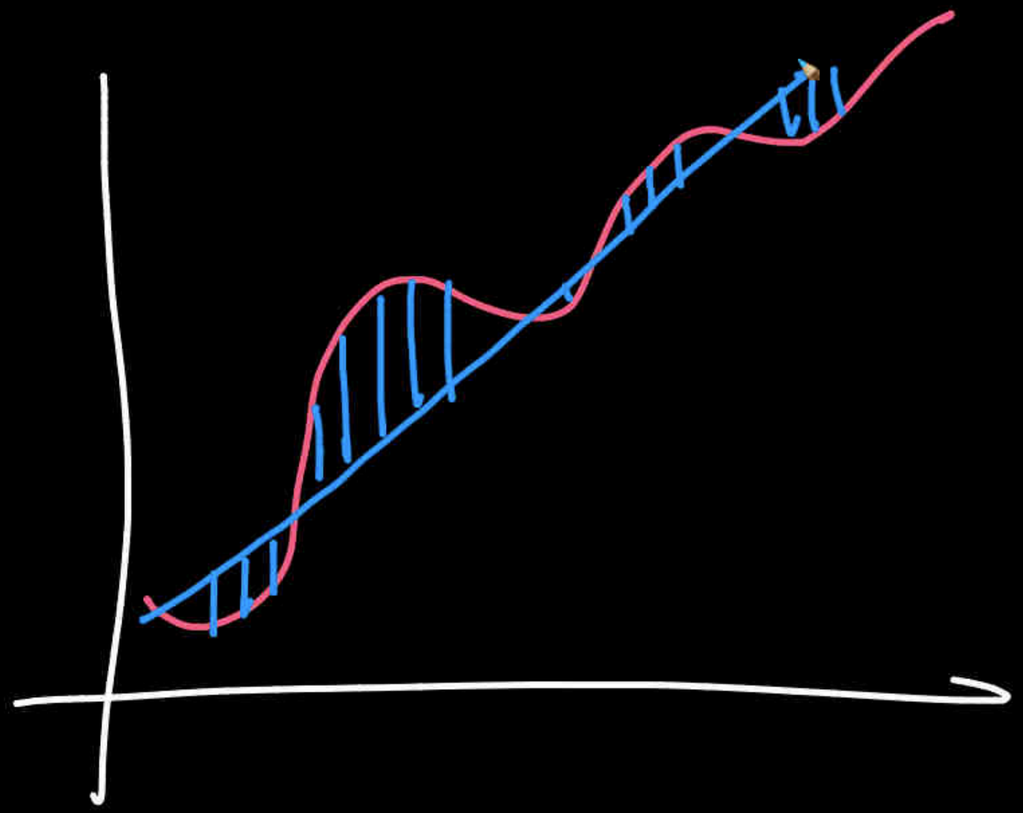
$$y_i - f_1(x_i) = err_i^1 \qquad \forall i : 1 \to n$$

Stage 2:



$$F_2(x) = \alpha_0 h_0(x) + \alpha_1 h_1(x) + \alpha_2 h_2(x)$$

Model after stage 2



repeat ....

{ M-Stages!

$$F_M(x) = \sum_{i=0}^{M} \alpha_i h_i(x)$$

mean for $h_0(x)$

shallow DT

optimization to find α_i

Boosting:- each α_i is learnt @ different time

$$F_m(x) = \sum_{i=0}^{M} \alpha_i h_i(x)$$

Boosting ⟹ additive weighted model

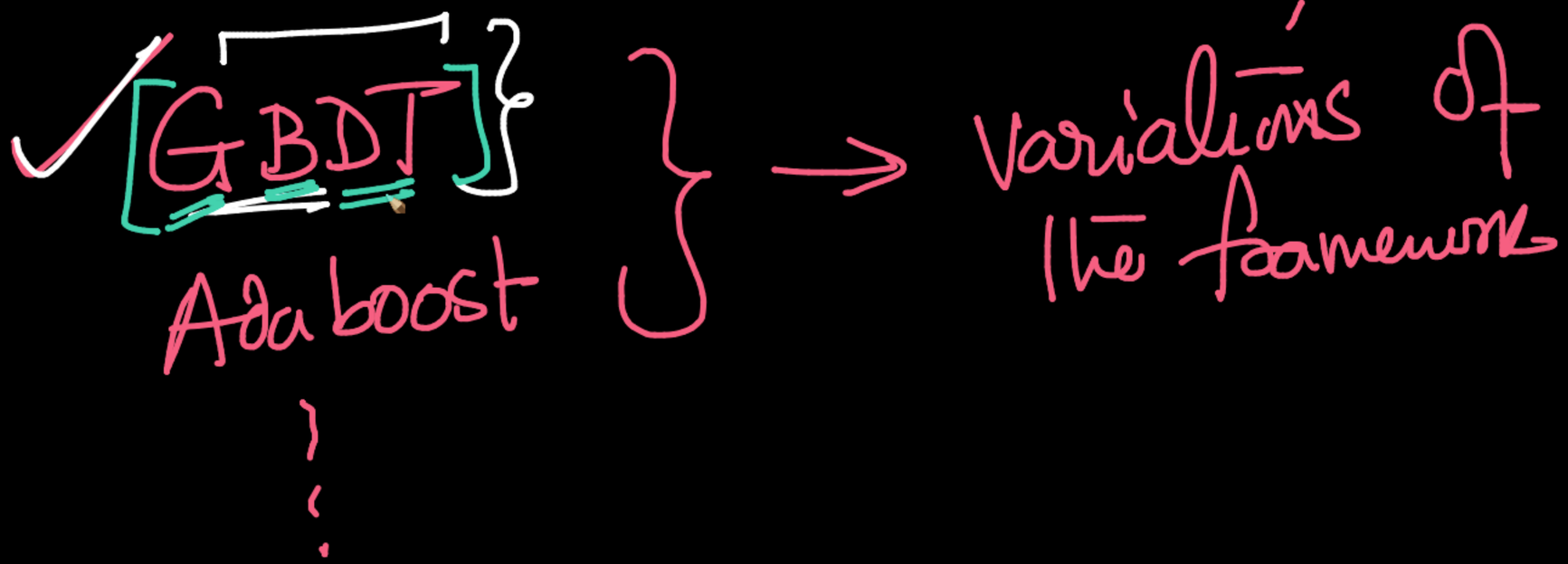
Lr. reg :-

$$F(x) = \sum_{i=1}^{d} w_i x_i + w_0$$

GD : all w_i's are learnt @ same time

err_i^j = errors after stage j

↑ "residuals"


GBDT
Adaboost
Variations of the framework

detour

$$F_K(x) = \sum_{i=0}^{K} \alpha_i h_i(x)$$

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

Residuals

Loss-fns:

Gradients

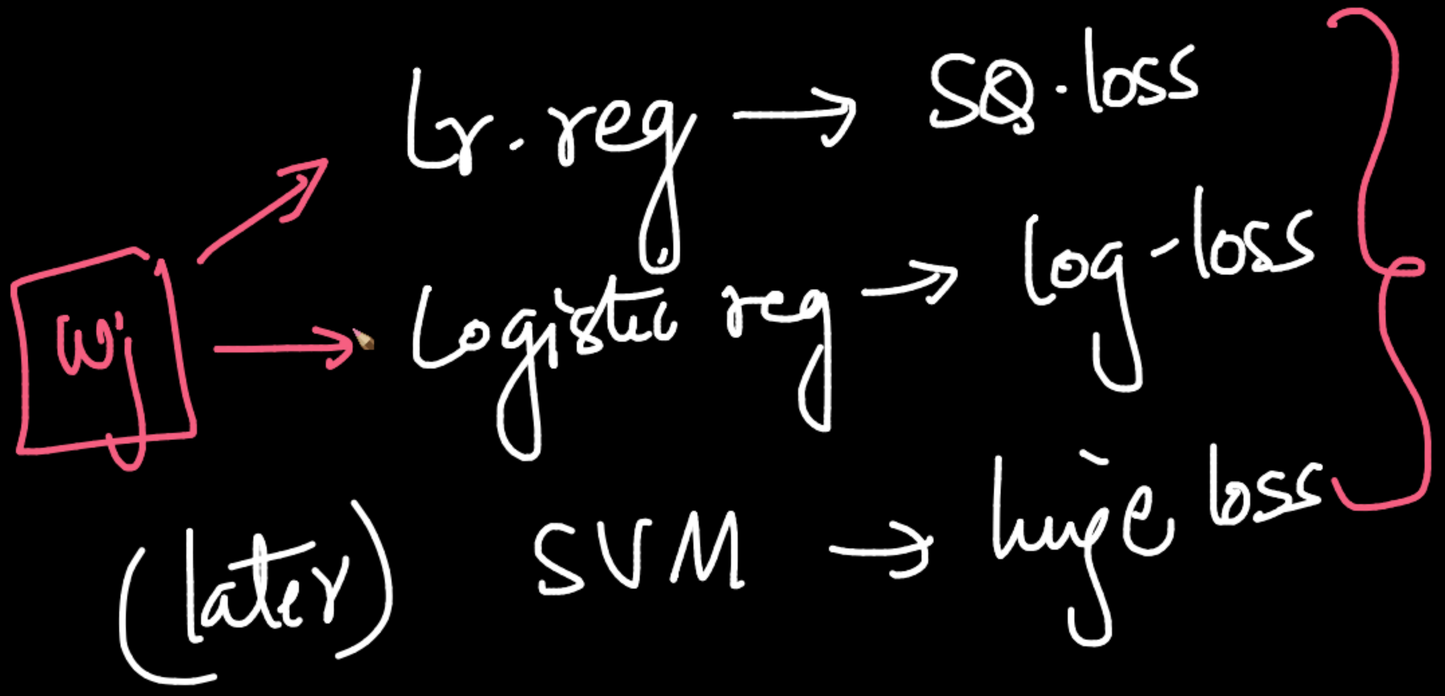
pseudo-residuals

GBDT

[any diff loss-function]

[sq-loss : reg]

Post read [log-loss : Classfn]


lr. reg → SQ. loss
Logistic reg → log-loss
(later) SVM → huge loss

e.g: sq. loss

$\hat{y}_i$: predicted

actual y_i

$$\mathcal{L}^{sq}(y_i, F_k(x_i)) = (y_i - F_k(x_i))^2$$

$$\frac{\partial \mathcal{L}^{sq}}{\partial F_k(x_i)} = (-1) \cdot 2 \cdot (y_i - F_k(x_i))$$

$$= -2 (y_i - F_k(x_i))$$

$$-\frac{\partial \mathcal{L}^{SQ}}{\partial F_k(x_i)} = 2\left(y_i - F_k(x_i)\right)$$

$$\text{error}_i^k = \text{res}_{i,k} \quad (\gamma_{ik})$$

pseudo residual

neg. grad = residual

Can prove for log-loss also → little more Math...

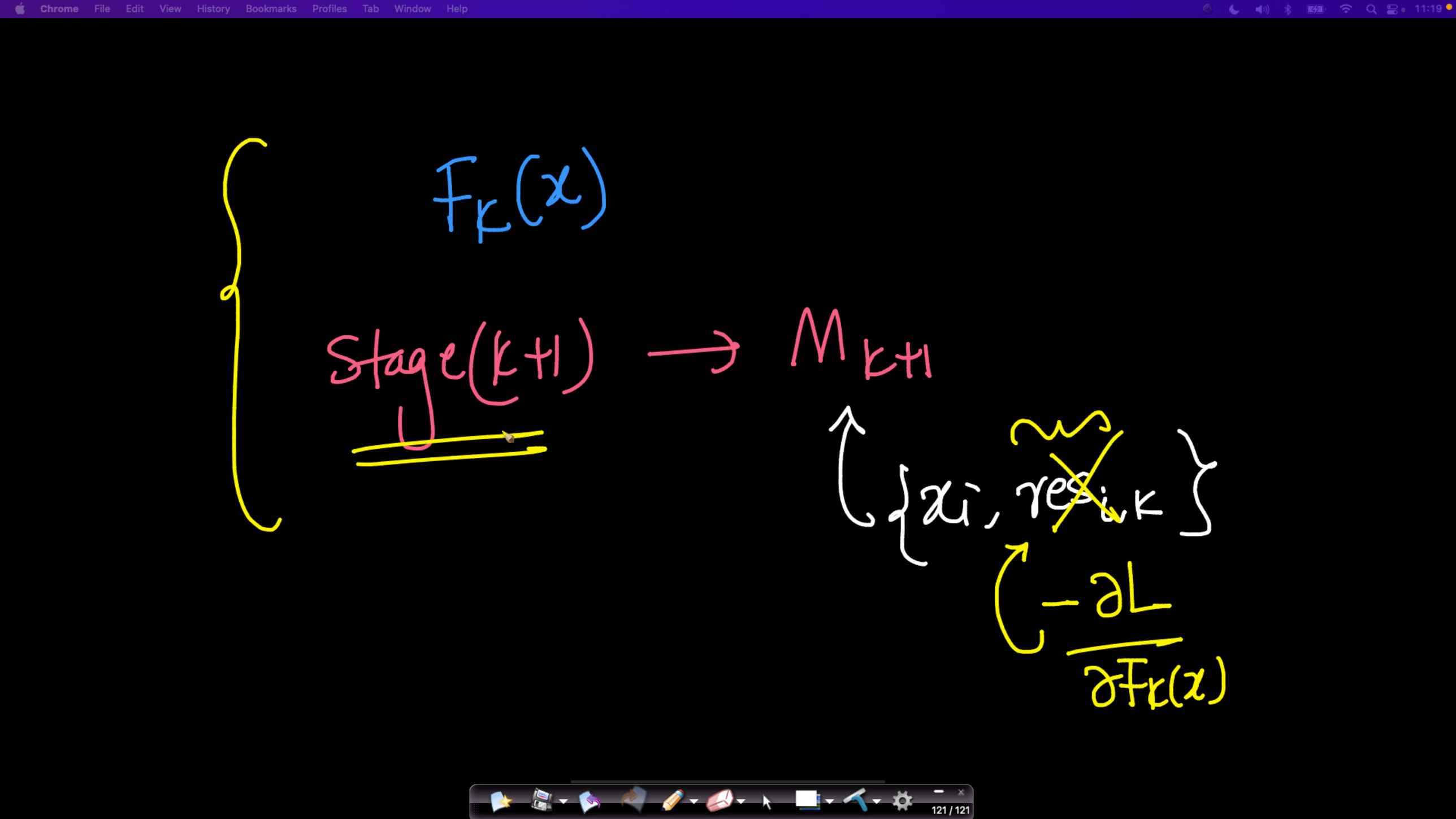
(Classfn)

$$\text{neg.grad} = \frac{-\partial L^{\log}}{\partial F_t(x_i)} = y_i - p_i$$

$\mathcal{L}$: differentiable - loss fn

$$-\frac{\partial \tilde{\mathcal{L}}}{\partial f_k(x_i)} = \text{residual}_{i,k}$$

→ error


$F_k(x)$
Stage(k+1)
M_{k+1}
$\{x_i, res_{i,k}\}$
$-\frac{\partial L}{\partial F_k(x)}$

To train $k+1^{th}$ base-model:

$$\left\{ x_i, -\frac{\partial L}{\partial F_k(x_i} \right\}$$

train on residuals
≈ train on
pseudo residuals

Boosting → neg. grad as pseudo residual

↓

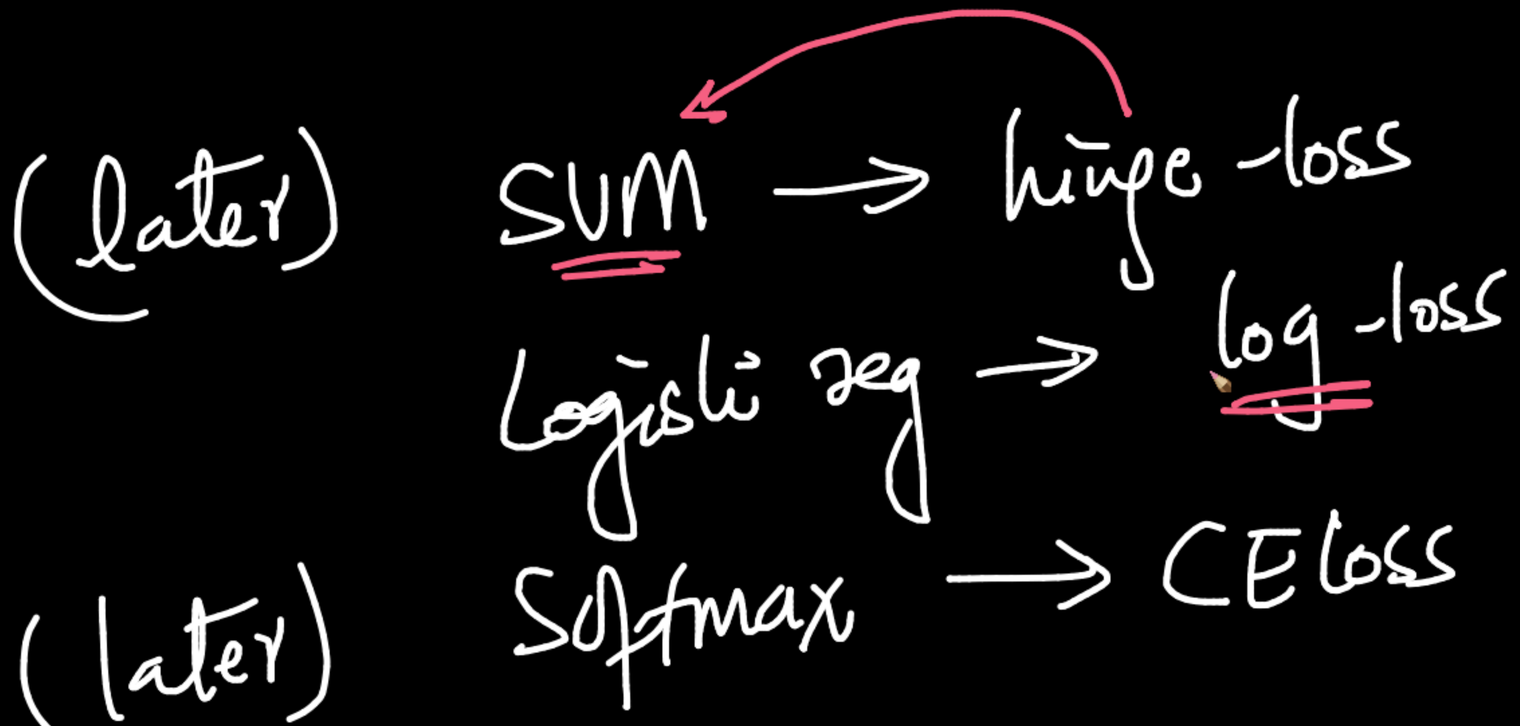
instead of the residual/error

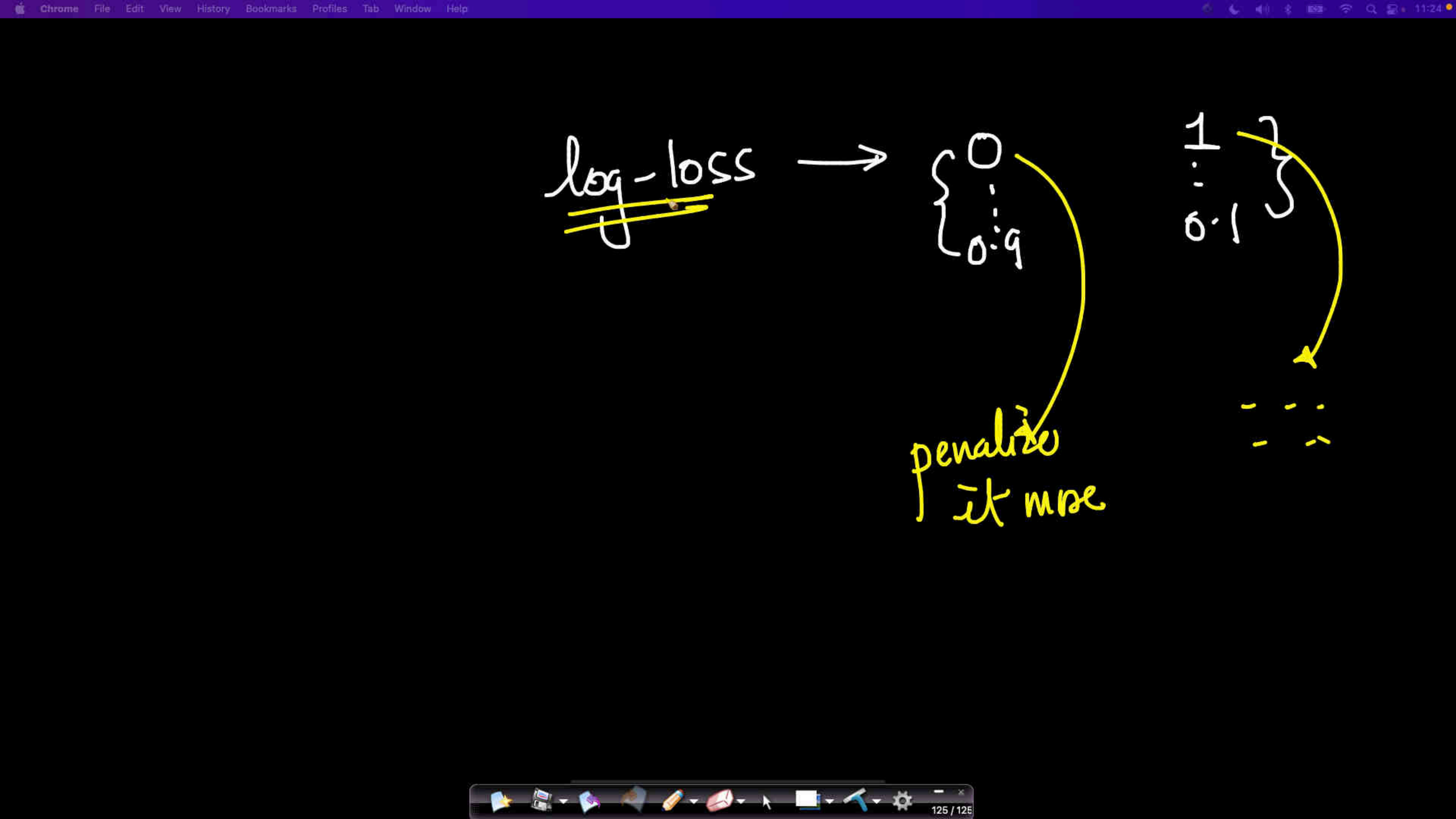
↓

{GBDT} → why is this popular

↳ Loss fn of my choice

Flexibility


(later)
SVM → hinge-loss
Logistic reg → log-loss
(later)
Softmax → CE loss


log-loss
0
0:9
1
0.1
penalize it more

Reg

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

- sq. loss → lr. reg
- [abs. loss]
- $(y_i - \hat{y}_i)^4$ → much more penalization of errors

however, i.e. when the set $\mathcal{H}$ is finite, we choose the candidate function h closest to the gradient of L for which the coefficient γ may then be calculated with the aid of line search on the above equations. Note that this approach is a heuristic and therefore doesn't yield an exact solution to the given problem, but rather an approximation. In pseudocode, the generic gradient boosting method is:[5][2]

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

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
    1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

    2. Fit a base learner (or weak learner, e.g. tree) closed under scaling $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
    3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

    4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

Gradient tree bo

solution to the given problem, but rather an approximation. In pseudocode, the generic gradient boosting method is:[5][2]

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

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
    1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

    2. Fit a base learner (or weak learner, e.g. tree) closed under scaling $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
    3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

    4. Update the model:

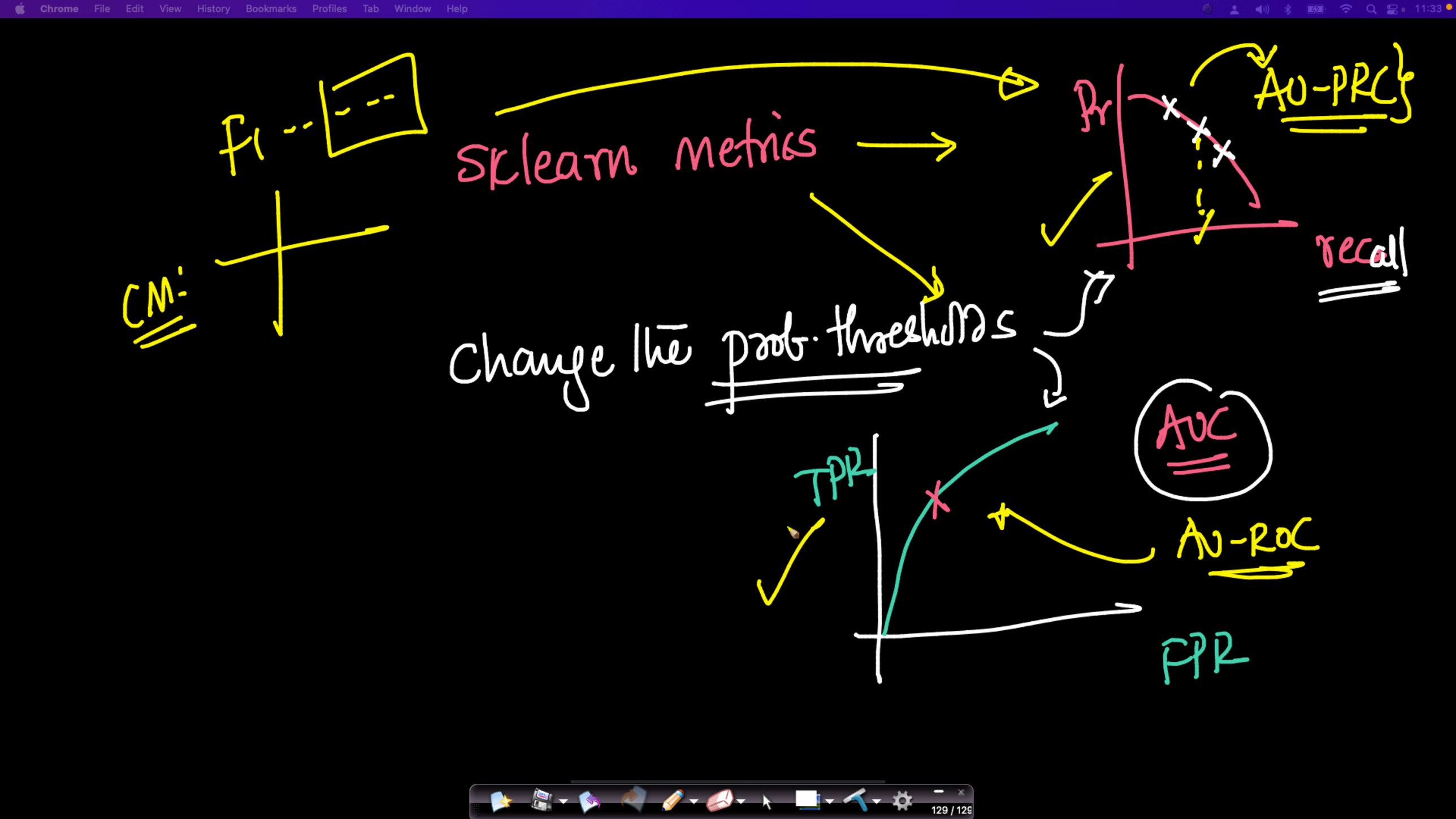
$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

Gradient tree boosting [edit]

Gradient boosting is typi… d size as base learners. For this special case,

Amazon


F1
CM:
sklearn metrics
Pr
Au-PRC
recall
Change the prob. thresholds
TPR
FPR
AUC
Au-RoC

$(x_i, y_i) \longrightarrow$ mean $h_0(x) \xrightarrow{err_i^0} h_1(x)$; $\alpha_0 h_0(x) + \alpha_1 h_1(x) = F_1(x)$

err_i^1

$h_2(x)$

$\alpha_0 h_0(x) + \alpha_1 h_1(x) + \alpha_2 h_2(x)$

$$err_i^j = y_i - F_j(x)$$

M base learners

```
clf = DecisionTreeClassifier(random_state=0, max_depth=1_best, class_weight={ 0:0.1, 1:w } )
clf.fit(X_train, y_train)

y_pred_val = clf.predict(X_val)
val_score = f1_score(y_val, y_pred_val)

print(val_score)

confusion_matrix(y_val, y_pred_val)
```

```
2
0.41618497109826585
array([[157,  87],
       [ 14,  36]])
```

```
# Feature importance
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1] # Sort feature importances in descending order
names = [X_train.columns[i] for i in indices] # Rearrange feature names so they match the sorted feature importance
plt.figure(figsize=(15, 7)) # Create plot
plt.title("Feature Importance") # Create plot title
plt.bar(range(X_train.shape[1]), importances[indices]) # Add bars
plt.xticks(range(X_train.shape[1]), names, rotation=90) # Add feature names as x-axis labels
plt.show() # Show plot
```

0s completed at 22:27