{ Plotting Packages:-
→ Solve simple DS

Task: Determine the eligibility for granting Home loan.

Objective of this notebook is:

1. To understand the patterns in the data.
2. How to Handle the categorical features.
3. How to deal with missing data.
4. Feature Engineering
5. Finding the most important features while taking the decision of granting a loan application.
6. Understanding the Normalization and standardisation of the data.

Load data and libraries

[1]
```
import numpy as np
import pandas as pd
from scipy import stats

import matplotlib.pyplot as plt
import seaborn as sns
```

Task: Determine the eligibility for granting Home loan.

Objective of this notebook is:

1. To understand the patterns in the data.
2. How to Handle the categorical features.
3. How to deal with missing data.
4. Feature Engineering
5. Finding the most important features while taking the decision of granting a loan application.
6. Understanding the Normalization and standardisation of the data.

Load data and libraries

[1]
```
import numpy as np
import pandas as pd
from scipy import stats

import matplotlib.pyplot as plt
import seaborn as sns
```

Home-loan:

Variables / features

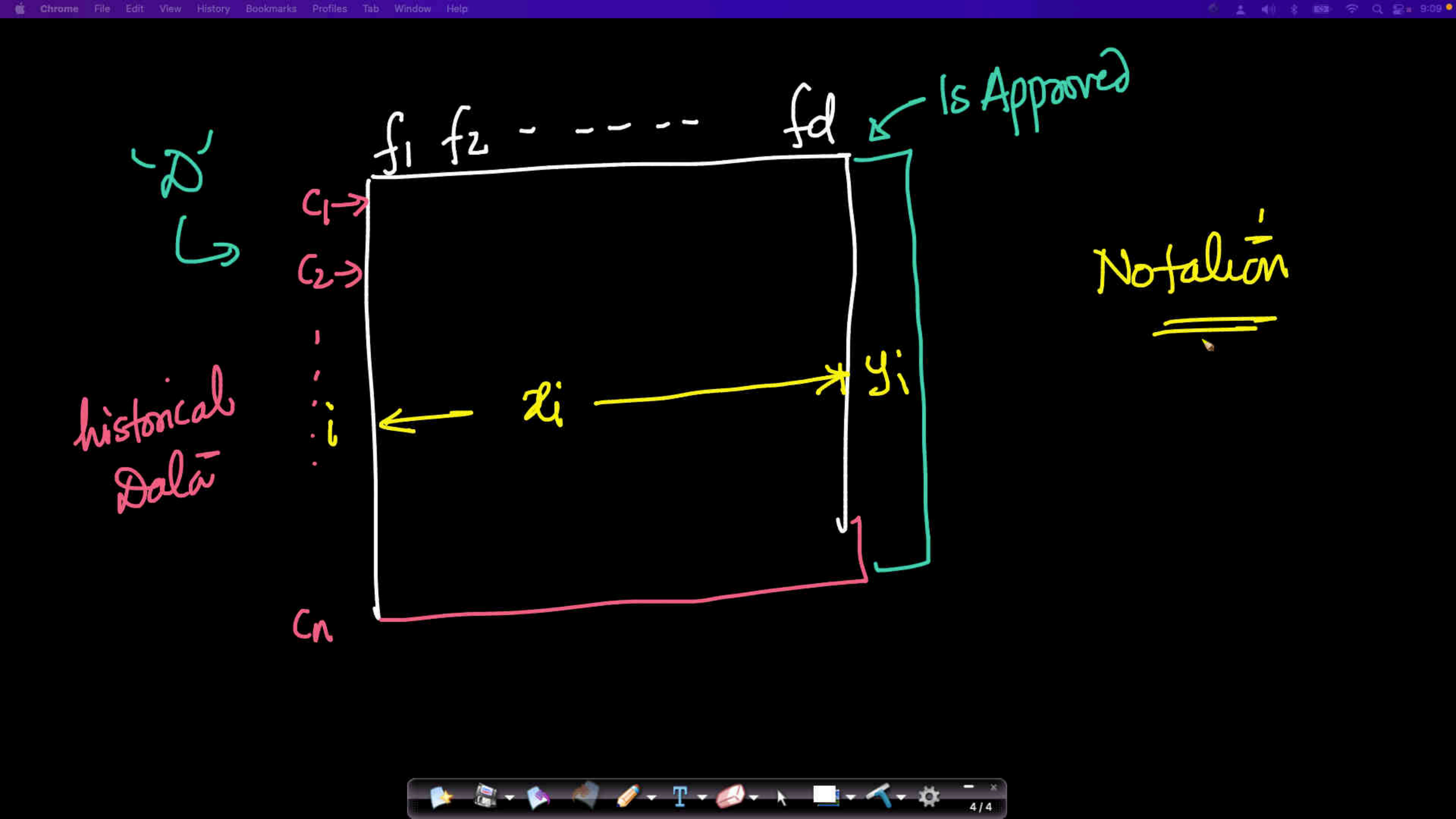
. . . .

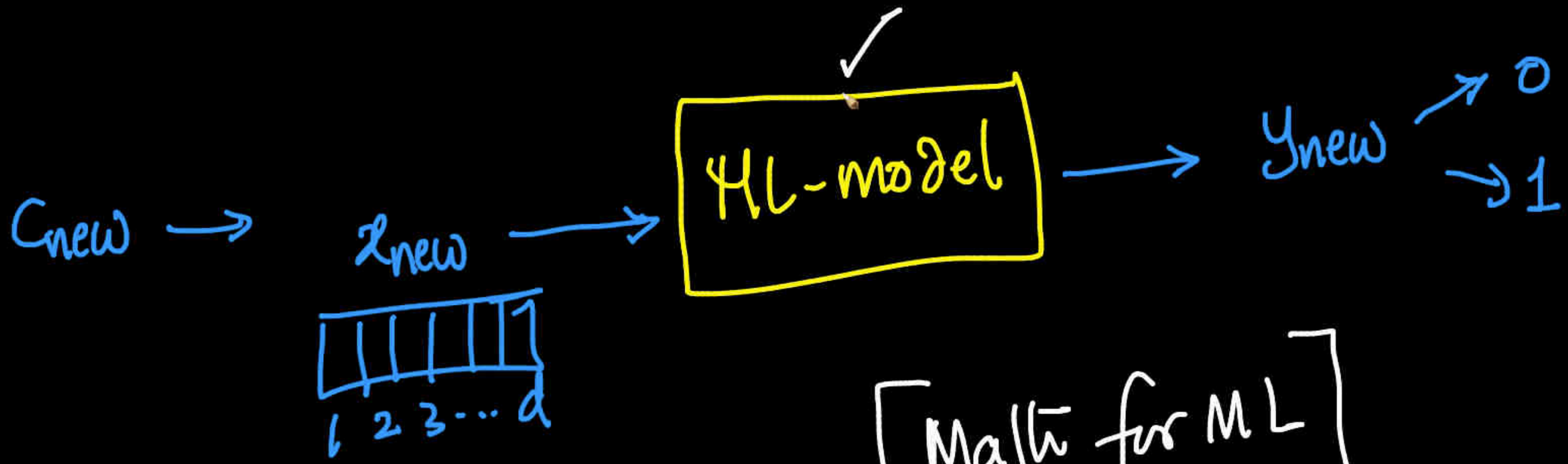
historical data

Customer → Is Loan Approved

features: Income; Credit score; Loan Amount; Age; Gender; Occupation; Education

RISK


f1 f2 - - - - - fd
Is Approved
D
C1
C2
i
Cn
xi
yi
historical Data
Notation



[Math for ML]

↓

Classical ML

+ Code + Text RAM Disk

```
LP001267,Female,Yes,2,Graduate,No,1378,1881,167,360,1,Urban,N
LP001273,Male,Yes,0,Graduate,No,6000,2250,265,360,,Semiurban,N
```

```
data = pd.read_csv('./train.csv')
data.shape
```

```
(614, 13)
```

[8]
```
data.columns
```

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

```
data.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loa
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	LP001006	Male			Not		2583	2358.0	120.0	

```
LP001267,Female,Yes,2,Graduate,No,1378,1881,167,360,1,Urban,N
LP001273,Male,Yes,0,Graduate,No,6000,2250,265,360,,Semiurban,N
```

```
data = pd.read_csv('./train.csv')
data.shape
```

```
(614, 13)
```

```
[8] data.columns
```

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

```
data.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loa
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	LP001006	Male			Not		2583	2358.0	120.0	

```
LP001267,Female,Yes,2,Graduate,No,1378,1881,167,360,1,Urban,N
LP001273,Male,Yes,0,Graduate,No,6000,2250,265,360,,Semiurban,N
```

```
data = pd.read_csv('./train.csv')
data.shape
```

```
(614, 13)
```

```
[8] data.columns
```

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

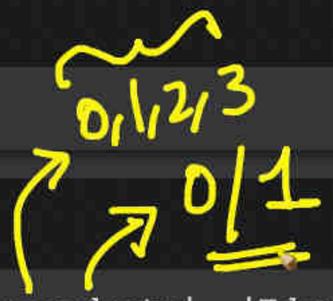
```
data.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loa
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	LP001006	Male			Not		2583	2358.0	120.0	

```
LP001267,Female,Yes,2,Graduate,No,1378,1881,167,360,1,Urban,N
LP001273,Male,Yes,0,Graduate,No,6000,2250,265,360,,Semiurban,N
```

```
data = pd.read_csv('./train.csv')
data.shape
```

```
(614, 13)
```



[8]
```
data.columns
```

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

[9]
```
data.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loa
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	

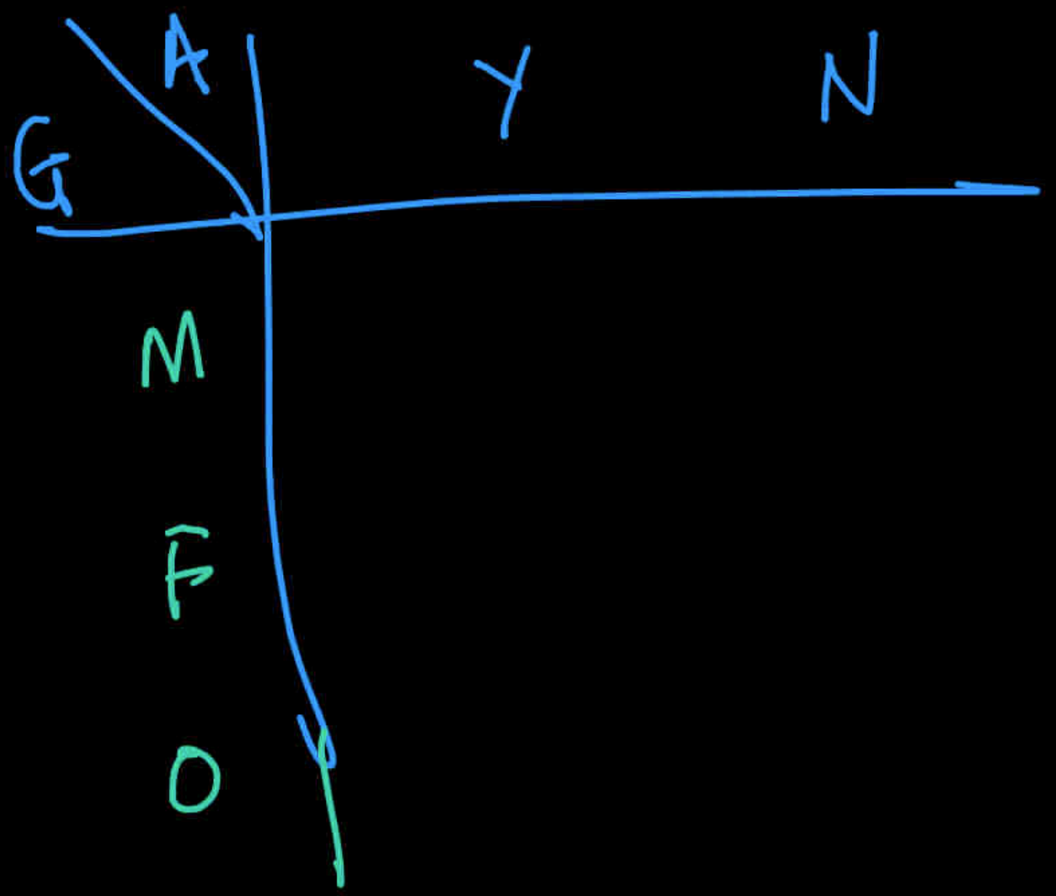
stats
indep. columns → ML features

Loan_Approved (cat) → Y / N

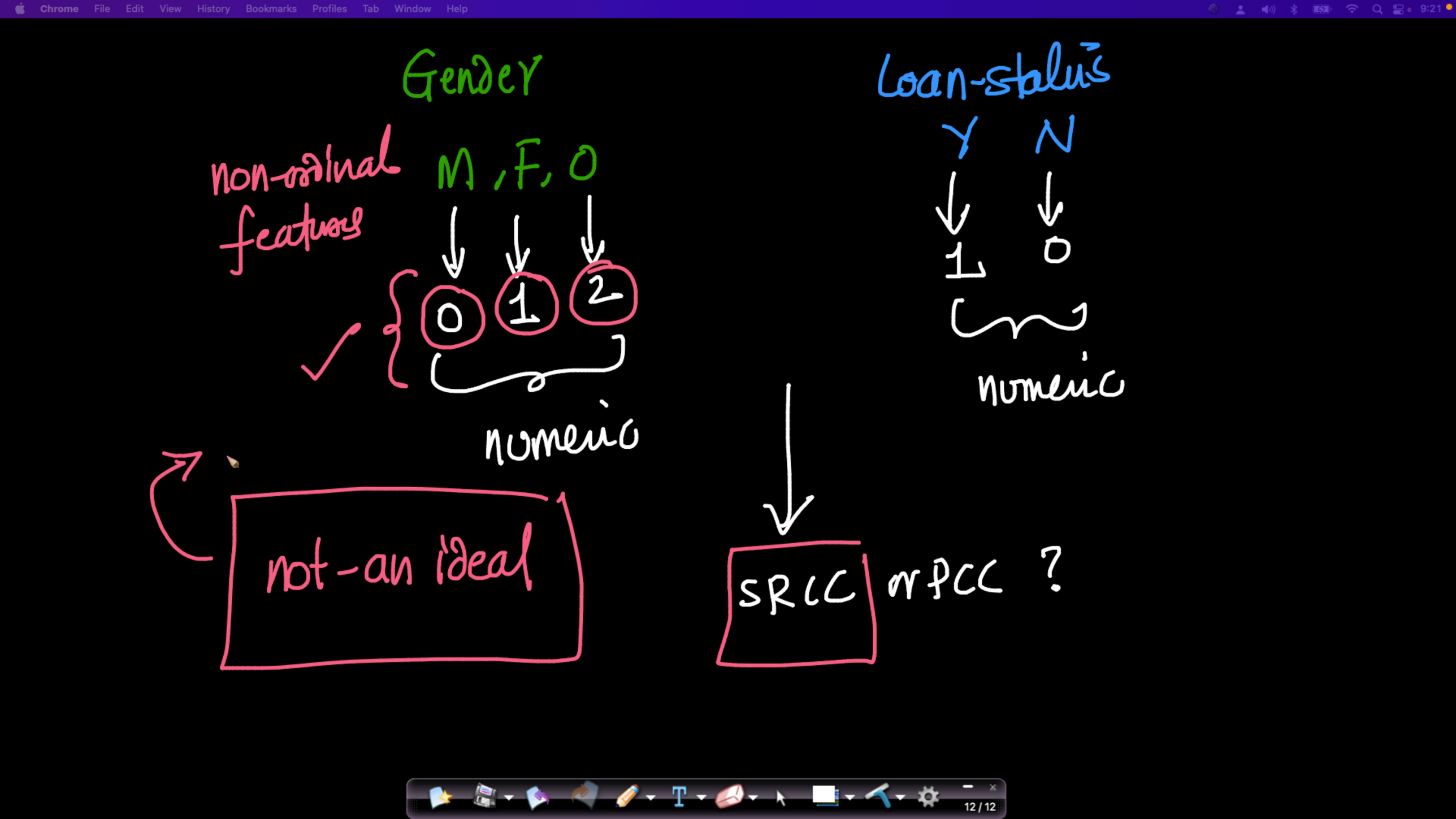
Gender (cat) → M / F / O

→ correlation

→ chi-square test



G indep of Appraisal or not?


Gender
non-ordinal features
M, F, O
0
1
2
numeric
not-an ideal
Loan-status
Y
N
1
0
numeric
SRCC
or PCC ?

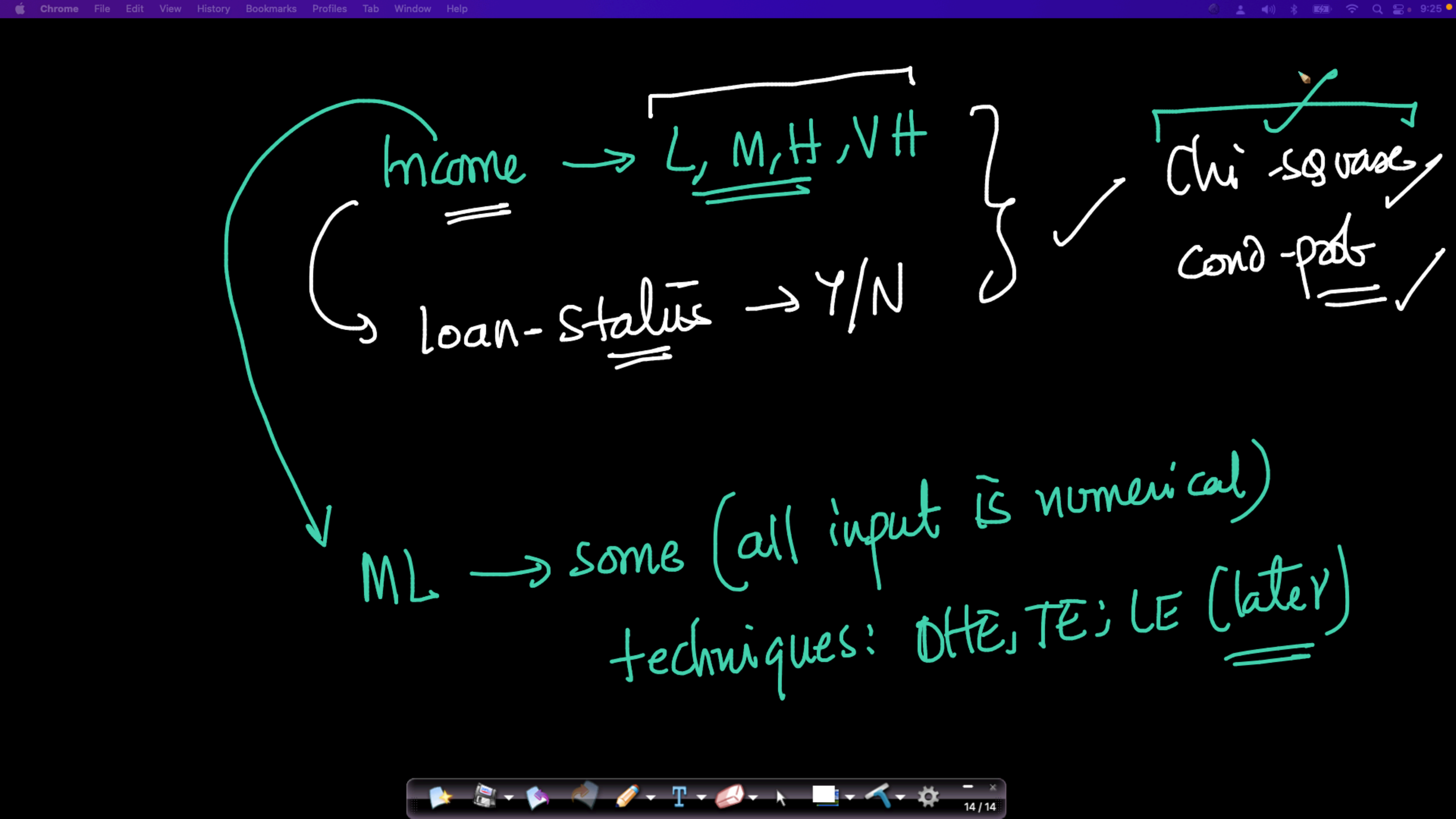
High-scool

Gender → M → Y, age of loans ase approved
F
O

$$\begin{cases} P(L=Y \mid G=M) \\ P(L=Y \mid G=F) \\ P(L=Y \mid G=O) \end{cases}$$

Cond. probs

Plot → . --


Income → L, M, H, VH
loan-status → Y/N
Chi-square
cond-prob
ML → some (all input is numerical)
techniques: OHE, TE; LE (later)

```
[10] data.dtypes
     #object => typically categorical/IDs
     #Int64, Float64
```

```
Loan_ID              object
Gender               object
Married              object
Dependents           object
Education            object
Self_Employed        object
ApplicantIncome       int64
CoapplicantIncome   float64
LoanAmount          float64
Loan_Amount_Term    float64
Credit_History      float64
Property_Area        object
Loan_Status          object
dtype: object
```

64bits

Real valued 64bits

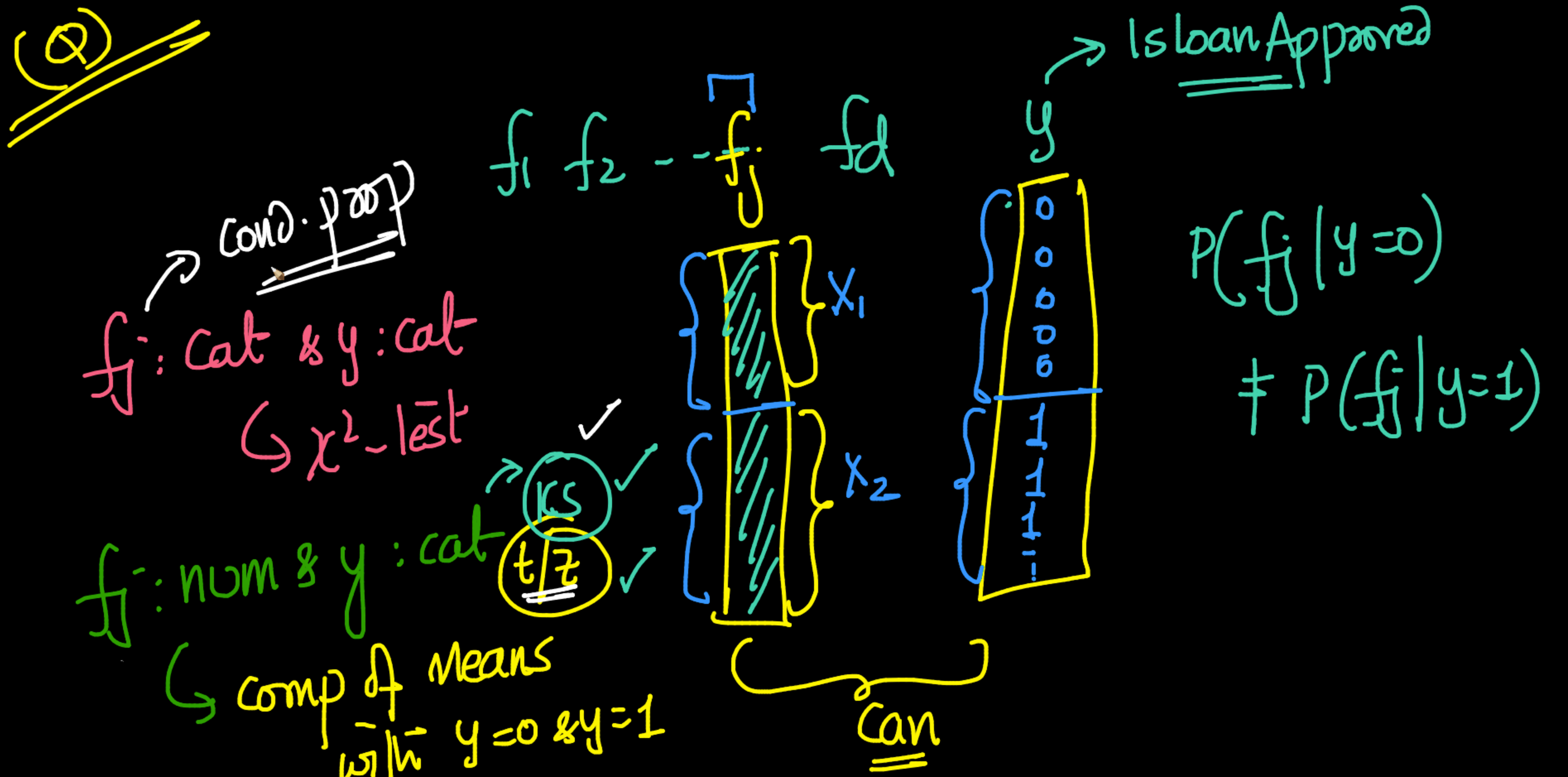
```
[11] # drop loanID column
     data = data.drop('Loan_ID',axis = 1)
```

Basic Data Exploration

```
[10] data.dtypes
     #object => typically categorical/IDs
     #Int64, Float64
```

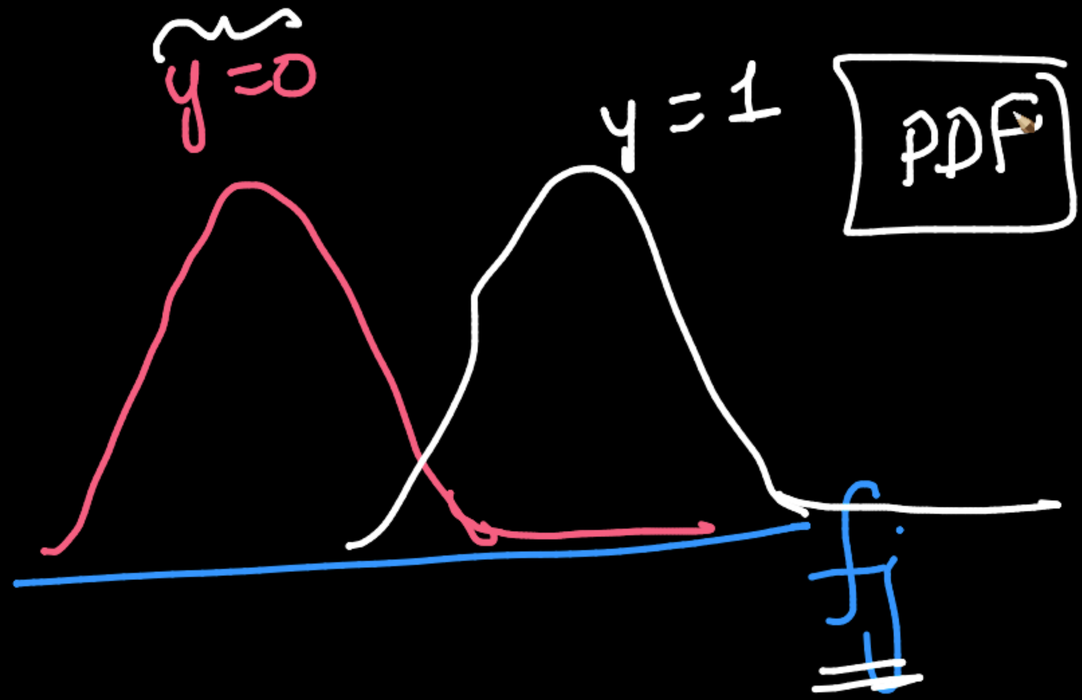
```
Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
Loan_Status           object
dtype: object
```

```
[11] # drop loanID column
     data = data.drop('Loan_ID',axis = 1)
```

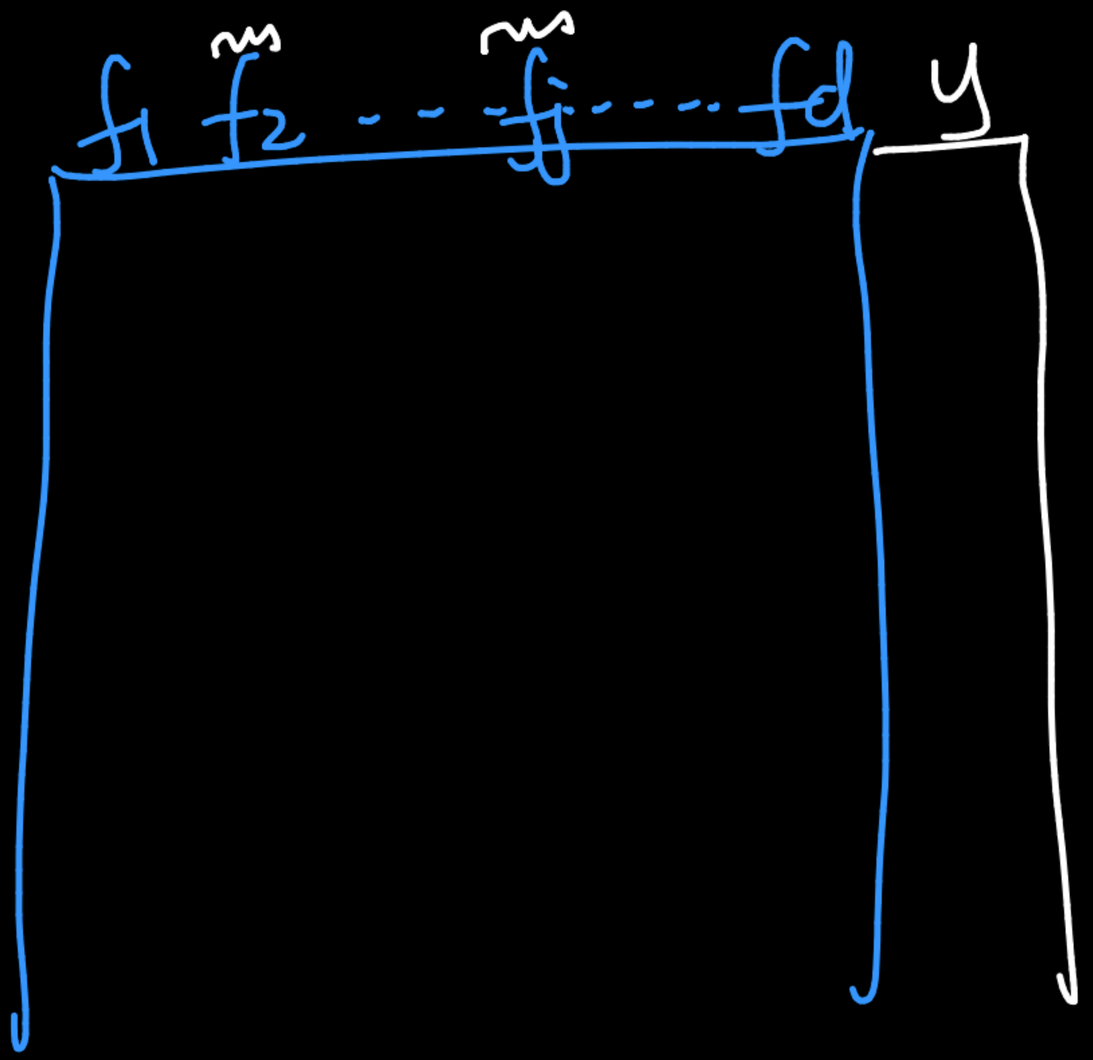

Q
cond. prob
f_j : cat & y : cat
↳ χ^2 - test
f_j : num & y : cat
↳ comp of means
with y=0 & y=1
f_1 f_2 - - f_j f_d
X_1
X_2
y
Is loan Approved
0 0 0 0 0
1 1 1 1 1
yes
t/f
can
$P(f_j | y=0)$
$\neq P(f_j | y=1)$

KS-Test:

CDFs



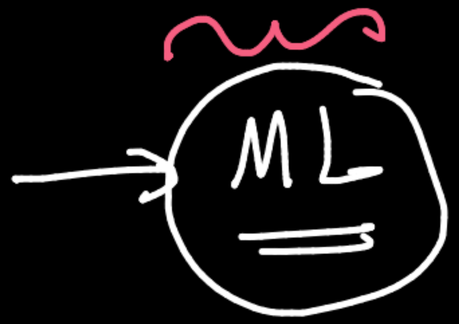
ANOVA! → Pop. per gp are Gaussian
↳ var across gps is same



f_i : numerical

y : numerical

(SRCC) : Correlation Coeff



feature Importance

[Classical ML]

Q

f: Income numeric

y: Loan-status Categorical → Y / N

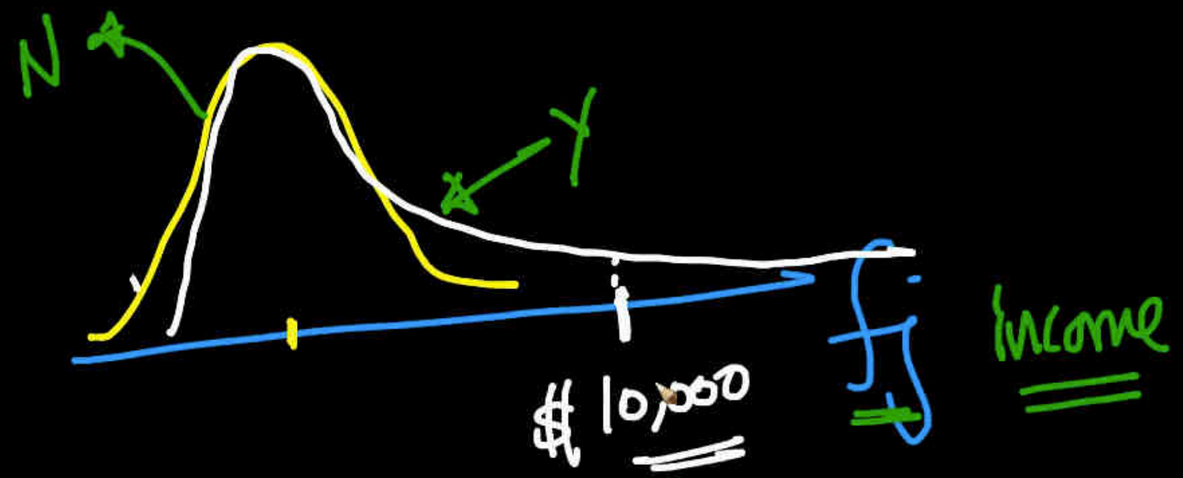
X1 → f → Y → n_1

X2 → f → N → n_2

Compare means

- t-test → n_1 & n_2 are small
- z-test → n_1 & $n_2 > 30$
- ks-test →

↳ compare risk : more powerful


N
Y
$10,000
Income

n_1 & n_2 are < 30

σ_1 & σ_2 are known

} → Z-test

n_1 & $n_2 > 30$

σ_1 & σ_2 are not known

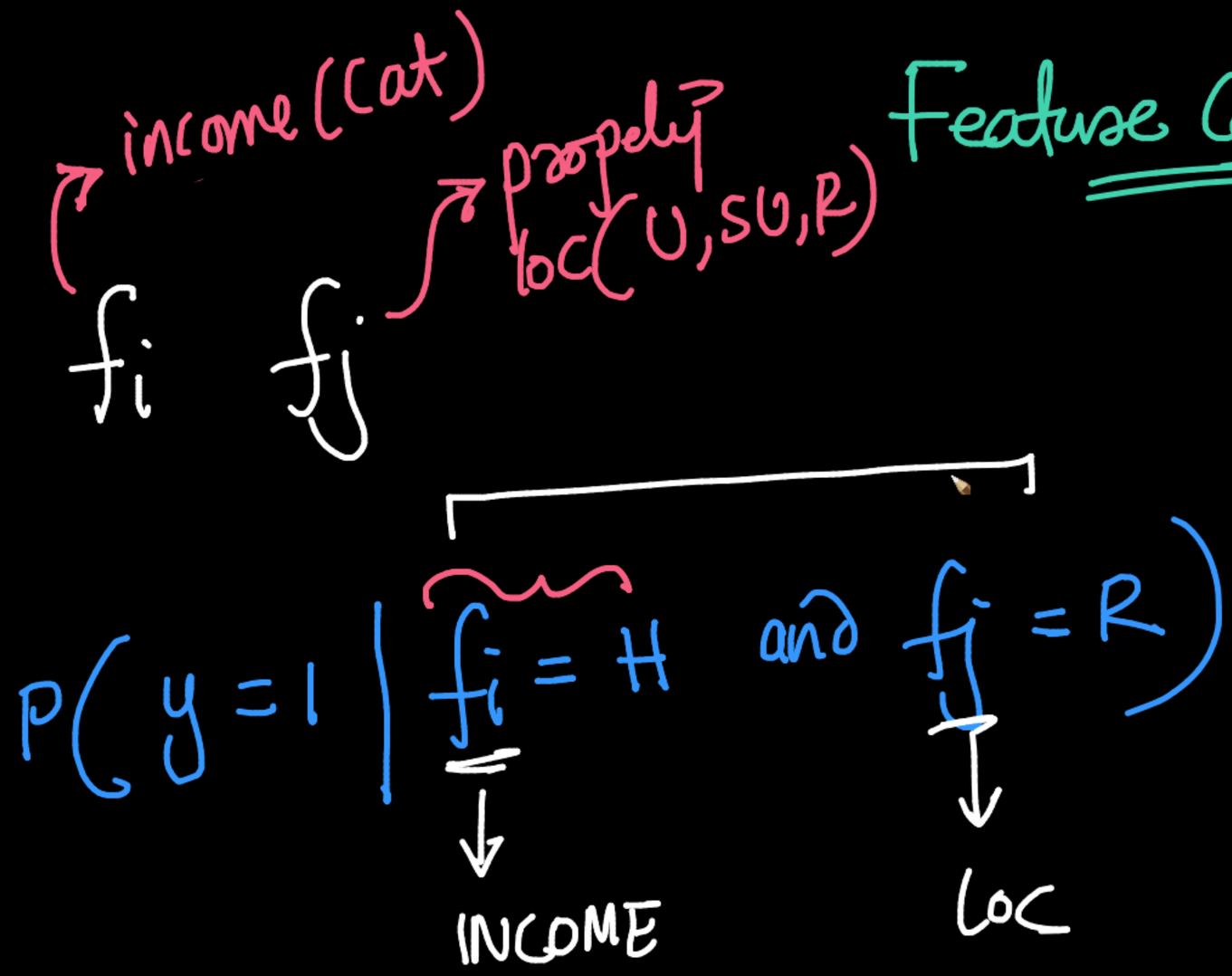
} → z-test

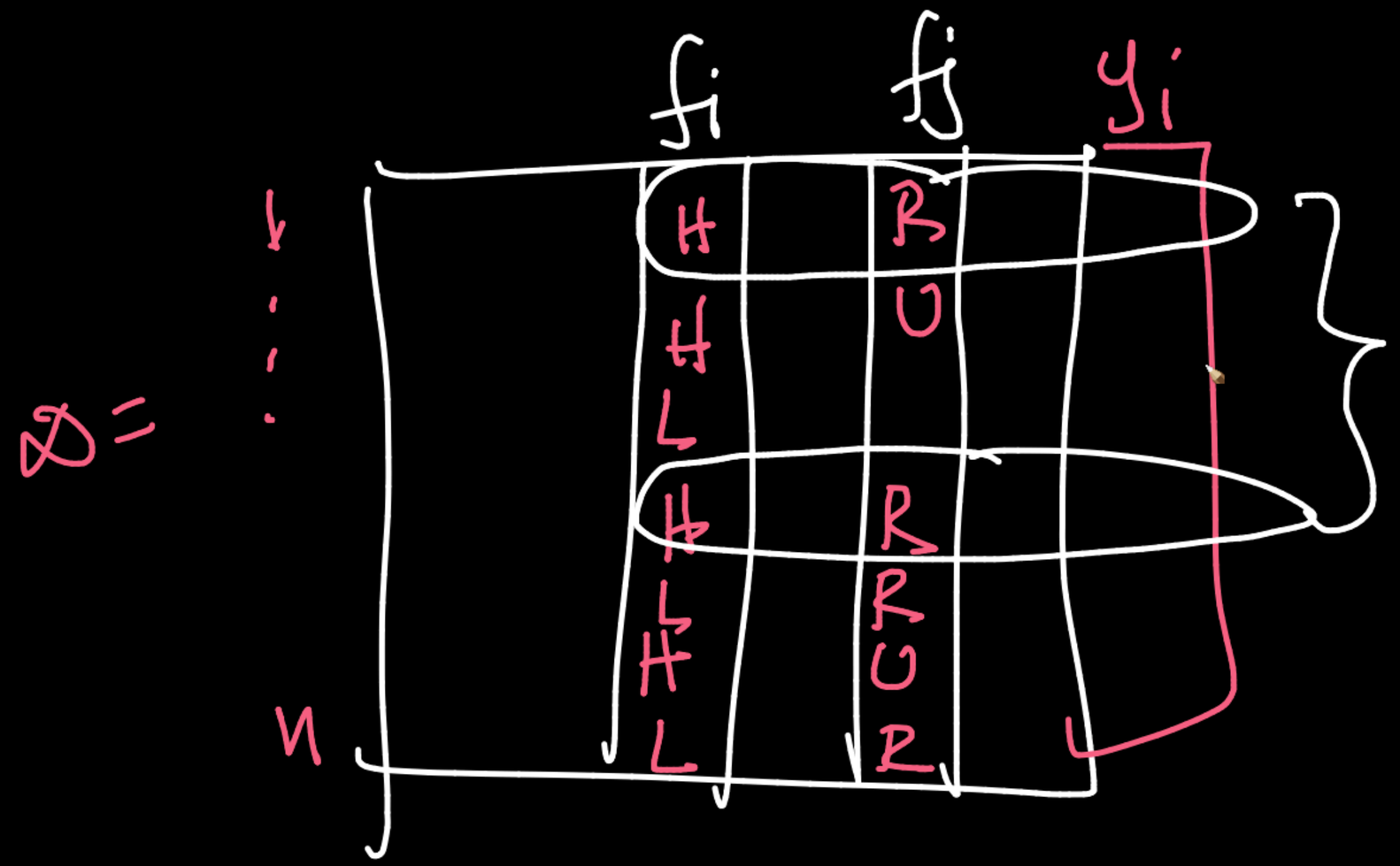
σ_1 & σ_2 estimated

σ_1 & σ_2 are not known
n_1 & n_2 are small
} → t-test

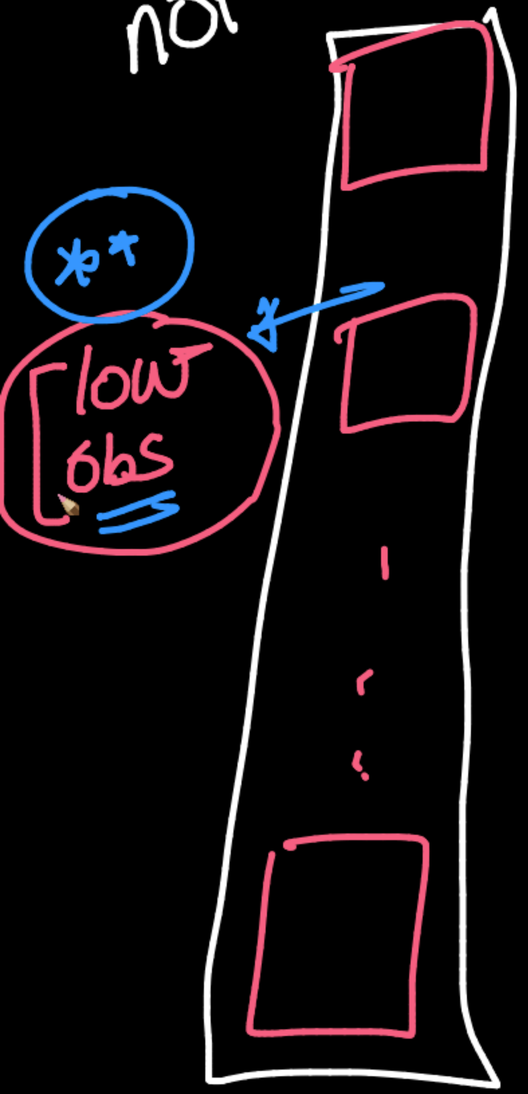
σ_1 & σ_2 are not known
n_1 & n_2 are large
} → t-test or z-test

Feature Combinations




fi
fj
yi
H
R
H
U
L
H
R
H
R
H
U
L
R

not-same



$= P(y=1 \mid$ INC=H and LOC=R) → one-line

$= P(y=1 \mid$ INC=VH and LOC=SU) $= \frac{1}{5}$

$f_1\ f_2\ ----\ f_d$

1 feat → d

2 feat → d_{C_2}

3 feat → d_{C_3}

⋮

d-feat → d_{C_d}

2^d

$d = 10$

$2^d = 1024$

ML-algo: → GBDT & RF

↓

automatically discover good combinations of features

Q

Task

$H_0: \mu_1 = \mu_2$

$H_a: \mu_1 \neq \mu_2$

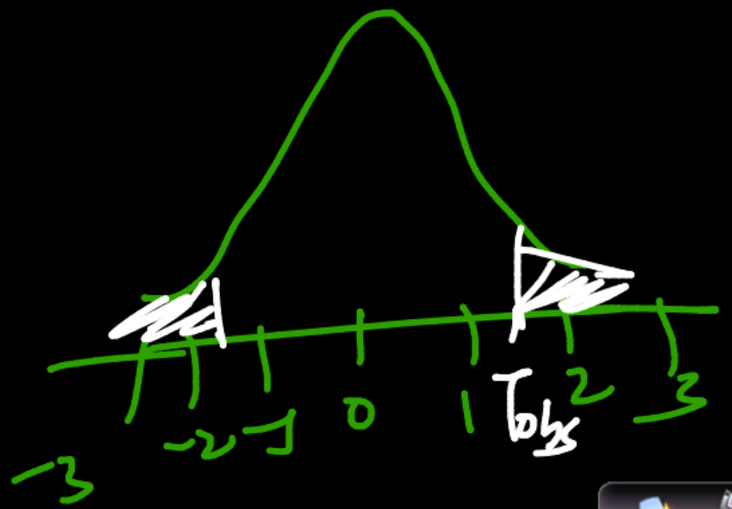
$n_1 = 85$

Z-Test ✓

$n_2 = 46$

$m_1 = 83$ $\quad s_1 = 8$

$m_2 = 88$ $\quad s_2 = 12$



$$T_z = \frac{m_1 - M_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \sim Z(0,1)$$

```
[12] data.describe()
     # only numeric features
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

Missing-data
nan
NaN

```
# catgeorical features
data.describe(include = ['object'])
```

	Gender	Married	Dependents	Education	Self_Employed	Property_Area	Loan_Status
count	601	611					614

[12]

25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

0s

```
# catgeorical features
data.describe(include = ['object'])
```

	Gender	Married	Dependents	Education	Self_Employed	Property_Area	Loan_Status
count	601	611	599	614	582	614	614
unique	2	2	4	2	2	3	2
top	Male	Yes	0	Graduate	No	Semiurban	Y
freq	489	398	345	480	500	233	422

0s

[16]

```
#missing values
data.isna().sum()
```

```
Gender          13
Married          3
Dependents      15
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help 10:05

EDA_FE.ipynb - Colaboratory

colab.research.google.com/drive/1_P-nKswaKnx5v77lOx3pCxvJve73MvnN#scrollTo=WsyIZ1nfHCG7

+ Code + Text RAM Disk

[12]

25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

0s

```
# catgeorical features
data.describe(include = ['object'])
```

	Gender	Married	Dependents	Education	Self_Employed	Property_Area	Loan_Status
count	601	611	599	614	582	614	614
unique	2	2	4	2	2	3	2
top	Male	Yes	0	Graduate	No	Semiurban	Y
freq	489	398	345	480	500	233	422

614

0s

```
[16] #missing values
     data.isna().sum()
```

```
Gender            13
Married            3
Dependents        15
```

32 / 32

[12]

25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

```
# catgeorical features
data.describe(include = ['object'])
```

	Gender	Married	Dependents	Education	Self_Employed	Property_Area	Loan_Status
count	601	611	599	614	582	614	614
unique	2	2	4	2	2	3	2
top	Male	Yes	0	Graduate	No	Semiurban	Y
freq	489	398	345	480	500	233	422

[16]
```
#missing values
data.isna().sum()
```

```
Gender          13
Married          3
Dependents      15
```

[12]

25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

```
# catgeorical features
data.describe(include = ['object'])
```

	Gender	Married	Dependents	Education	Self_Employed	Property_Area	Loan_Status
count	601	611	599	614	582	614	614
unique	2	2	4	2	2	3	2
top	Male	Yes	0	Graduate	No	Semiurban	Y
freq	489	398	345	480	500	233	422

```
[16] #missing values
     data.isna().sum()
```

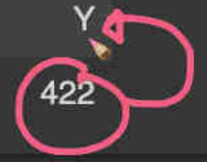
```
Gender          13
Married          3
Dependents      15
```

[12]

25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

```
# catgeorical features
data.describe(include = ['object'])
```

	Gender	Married	Dependents	Education	Self_Employed	Property_Area	Loan_Status
count	601	611	599	614	582	614	614
unique	2	2	4	2	2	3	2
top	Male	Yes	0	Graduate	No	Semiurban	Y
freq	489	398	345	480	500	233	422



[16]
```
#missing values
data.isna().sum()
```

```
Gender            13
Married            3
Dependents        15
```

NaN
2,326...

```
[12] data.describe()
     # only numeric features
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

```
# catgeorical features
data.describe(include = ['object'])
```

	Gender	Married	Dependents	Education	Self_Employed	Property_Area	Loan_Status

unique	2	2	4	2	2	3	2
top	Male	Yes	0	Graduate	No	Semiurban	Y
freq	489	398	345	480	500	233	422

```
[16] #missing values
     data.isna().sum()
```

```
Gender              13
Married              3
Dependents          15
Education            0
Self_Employed       32
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
dtype: int64
```

NaN

```
[17] # catgeorical and numerical columns
     cat_cols = data.dtypes =='object'
     cat_cols = list(cat_cols[cat_cols].index)
```

unique	2	2	4	2	2	3	2
top	Male	Yes	0	Graduate	No	Semiurban	Y
freq	489	398	345	480	500	233	422

```
[16] #missing values
     data.isna().sum()
```

```
Gender              13
Married              3
Dependents          15
Education            0
Self_Employed       32
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
dtype: int64
```

```
[17] # catgeorical and numerical columns
     cat_cols = data.dtypes =='object'
     cat_cols = list(cat_cols[cat_cols].index)
```

```
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

```
# catgeorical and numerical columns
cat_cols =(data.dtypes =='object')
cat_cols = list(cat_cols[cat_cols].index)

num_cols = data.dtypes !='object'
num_cols = list(num_cols[num_cols].index)
cat_cols.remove('Loan_Status')
```

```
cat_cols
```

```
['Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'Property_Area']
```

```
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

[16] 0s

```
[17] # catgeorical and numerical columns
cat_cols = data.dtypes =='object'
cat_cols = list(cat_cols[cat_cols].index)

num_cols = data.dtypes !='object'
num_cols = list(num_cols[num_cols].index)
cat_cols.remove('Loan_Status')
```

```
[18] cat_cols
```

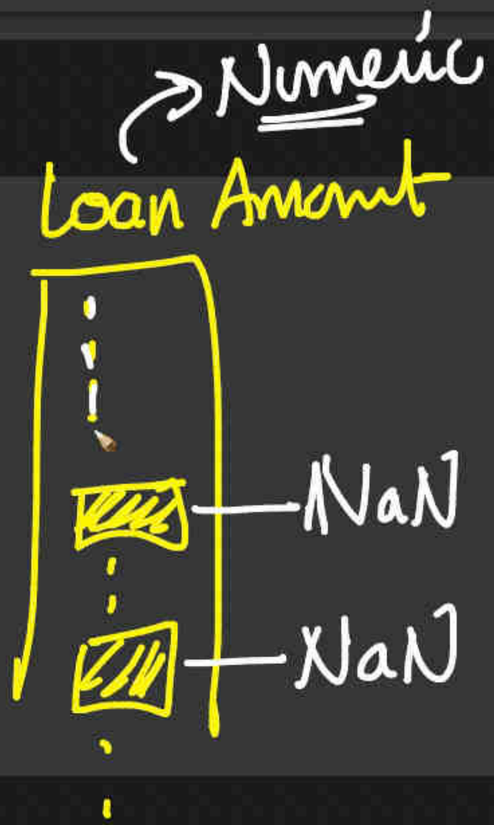
0s

```
['Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'Property_Area']
```

freq	489	398	345	480	500	233	422

[16] #missing values
data.isna().sum()

```
Gender              13
Married              3
Dependents          15
Education            0
Self_Employed       32
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
dtype: int64
```



[17] # catgeorical and numerical columns
cat_cols = data.dtypes =='object'
cat_cols = list(cat_cols[cat_cols].index)

num_cols = data.dtypes !='object'
num_cols = list(num_cols[num_cols].index)
cat_cols.remove('Loan St

```
[10] Loan_ID              object
     Gender               object
     Married              object
     Dependents           object
     Education            object
     Self_Employed        object
     ApplicantIncome       int64
     CoapplicantIncome   float64
     LoanAmount          float64
     Loan_Amount_Term    float64
     Credit_History      float64
     Property_Area        object
     Loan_Status          object
     dtype: object
```

```
data['Dependents'].value_counts()
```

```
0     345
1     102
2     101
3+     51
Name: Dependents, dtype: int64
```

```
[96] # drop loanID column
     data = data.drop('Loan_ID',axis = 1)
```

[10]
```
Loan_ID              object
Gender               object
Married              object
Dependents           object
Education            object
Self_Employed        object
ApplicantIncome       int64
CoapplicantIncome   float64
LoanAmount          float64
Loan_Amount_Term    float64
Credit_History      float64
Property_Area        object
Loan_Status          object
dtype: object
```

0s
```
data['Dependents'].value_counts()
```

```
0     345
1     102
2     101
3+     51
Name: Dependents, dtype: int64
```

[96] 0s
```
# drop loanID column
data = data.drop('Loan_ID',axis = 1)
```

[10]
```
Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
Loan_Status           object
dtype: object
```

0s
```
data['Dependents'].value_counts()
```

```
0     345
1     102
2     101
3      51
Name: Dependents, dtype: int64
```

[96]
```
# drop loanID column
data = data.drop('Loan_ID',axis = 1)
```

```
# drop loanID column
data = data.drop('Loan_ID',axis = 1)
```

object

Categorical — nominal

ordinal

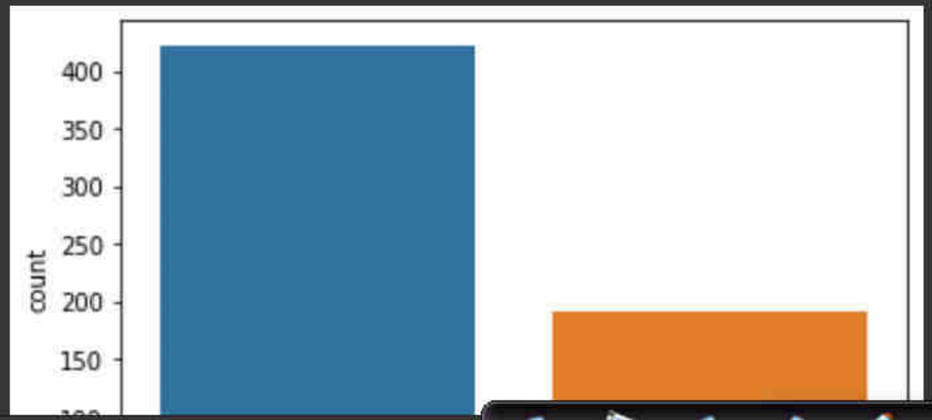
Basic Data Exploration

[ ] ↳ 9 cells hidden

Basic Data visualization: Univariate

1

```
#Q: How many loans the company has approved in the past?
sns.countplot(data=data, x='Loan_Status')
plt.show()
```



Chrome File Edit View History Bookmarks Profiles Tab Window Help 10:18

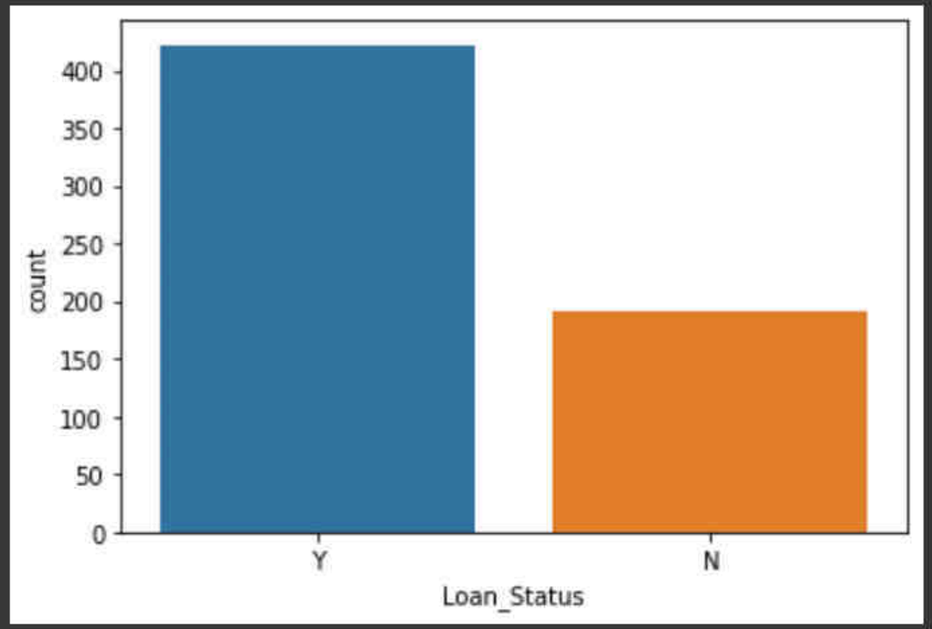
EDA_FE.ipynb - Colaboratory

colab.research.google.com/drive/1_P-nKswaKnx5v77lOx3pCxvJve73MvnN#scrollTo=GNmtX3um9oJR

+ Code + Text

RAM Disk

```
[22] #Q: How many loans the company has approved in the past?
     sns.countplot(data=data, x='Loan_Status')
     plt.show()
```



Questions [Sherlock-holmes]

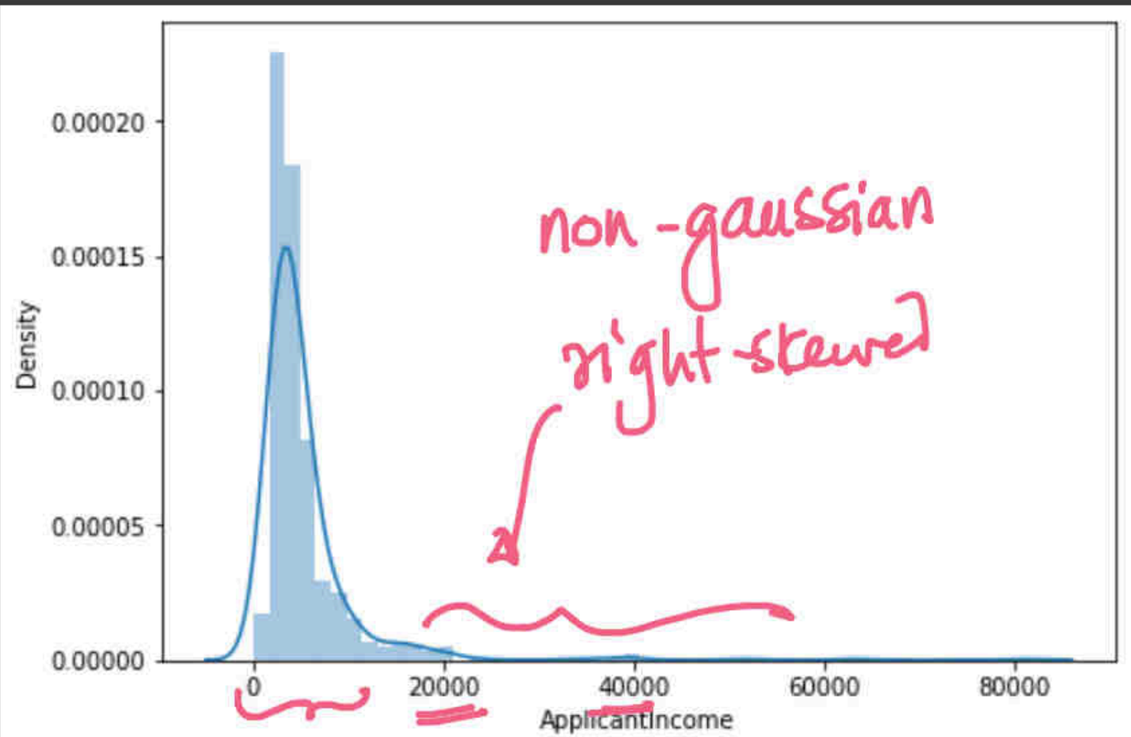
+ tools (Post-stats, ML, DL---) high-school

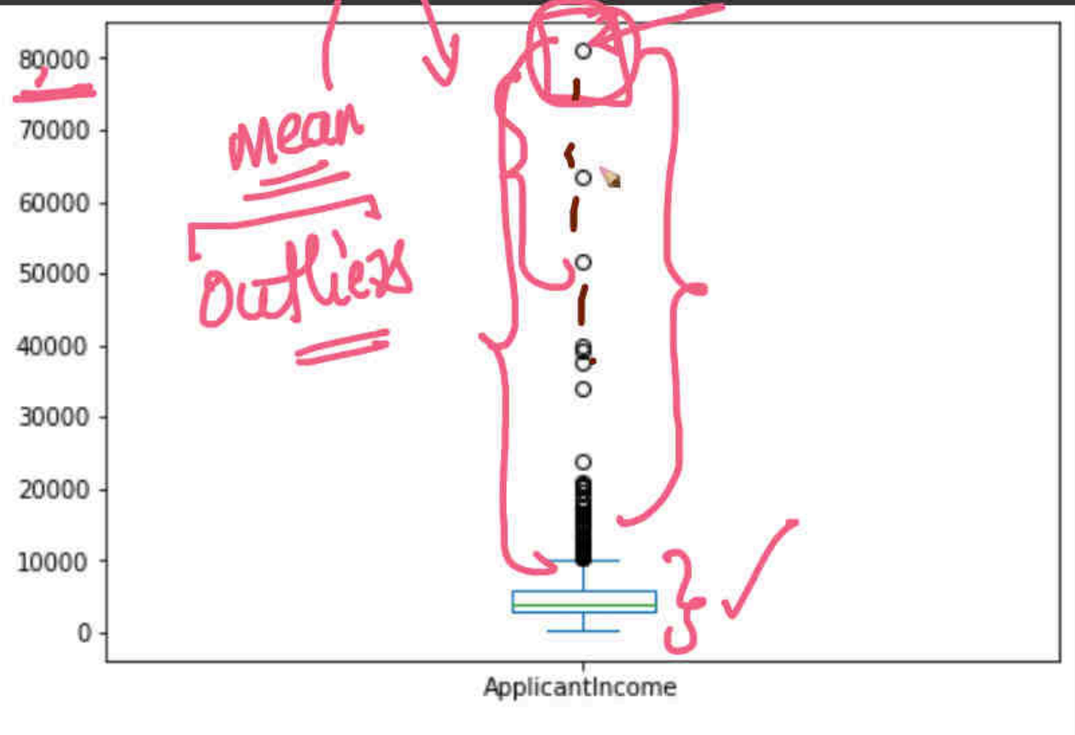
```
[23] target = 'Loan_Status'
     data[target].value_counts()

     # Imbalanced data
```

```
Y    422
N    192
```

46 / 47

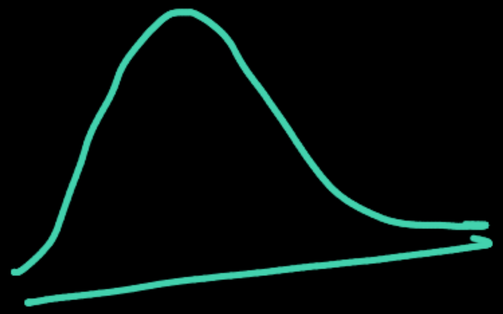
```
#Income of the applicant
plt.subplot(121)
sns.distplot(data["ApplicantIncome"])

plt.subplot(122)
data["ApplicantIncome"].plot.box(figsize=(16,5))
plt.show()
```



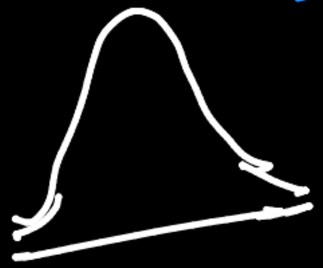


Q

Income → outliers ⇒ we want to avoid t/z test



↓ log / box cox



Can I somehow find a soln using t/z test

↳ remove outliers (IQR) ✓

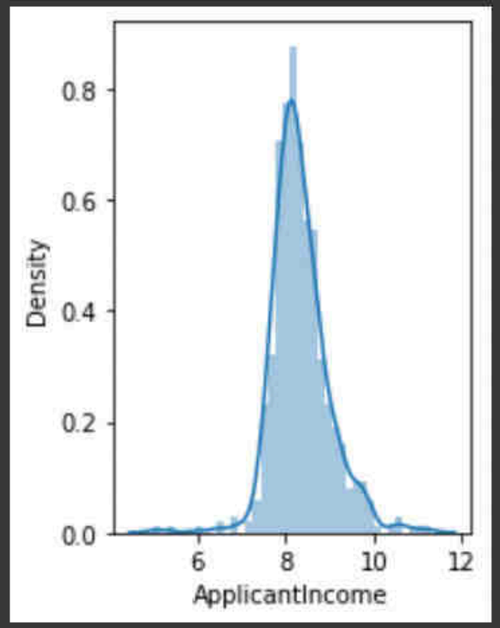
→ median X

→

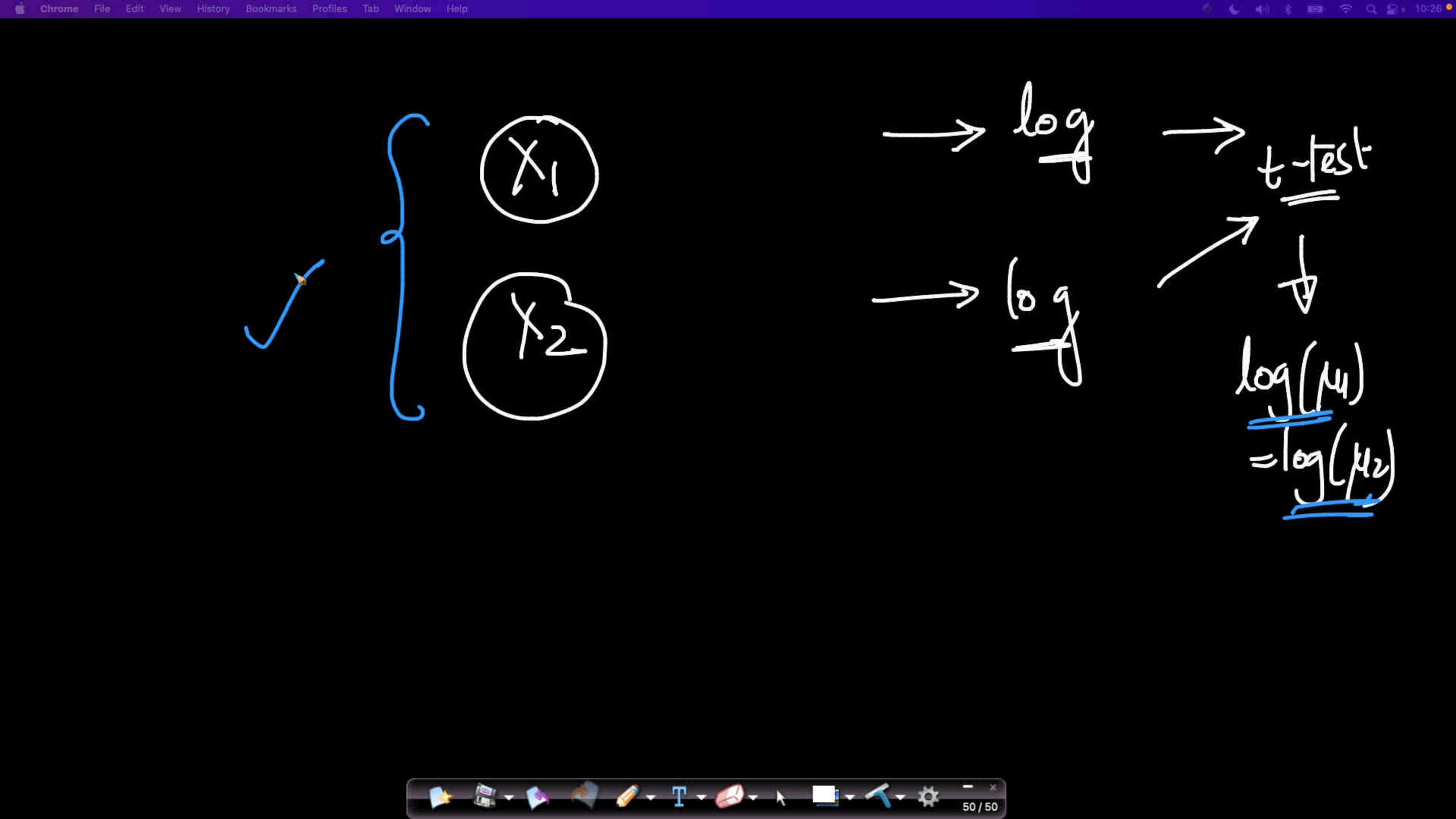
ApplicantIncome

```
plt.subplot(121)
sns.distplot(np.log(data["ApplicantIncome"]))

plt.show()
```

box-cox ✓

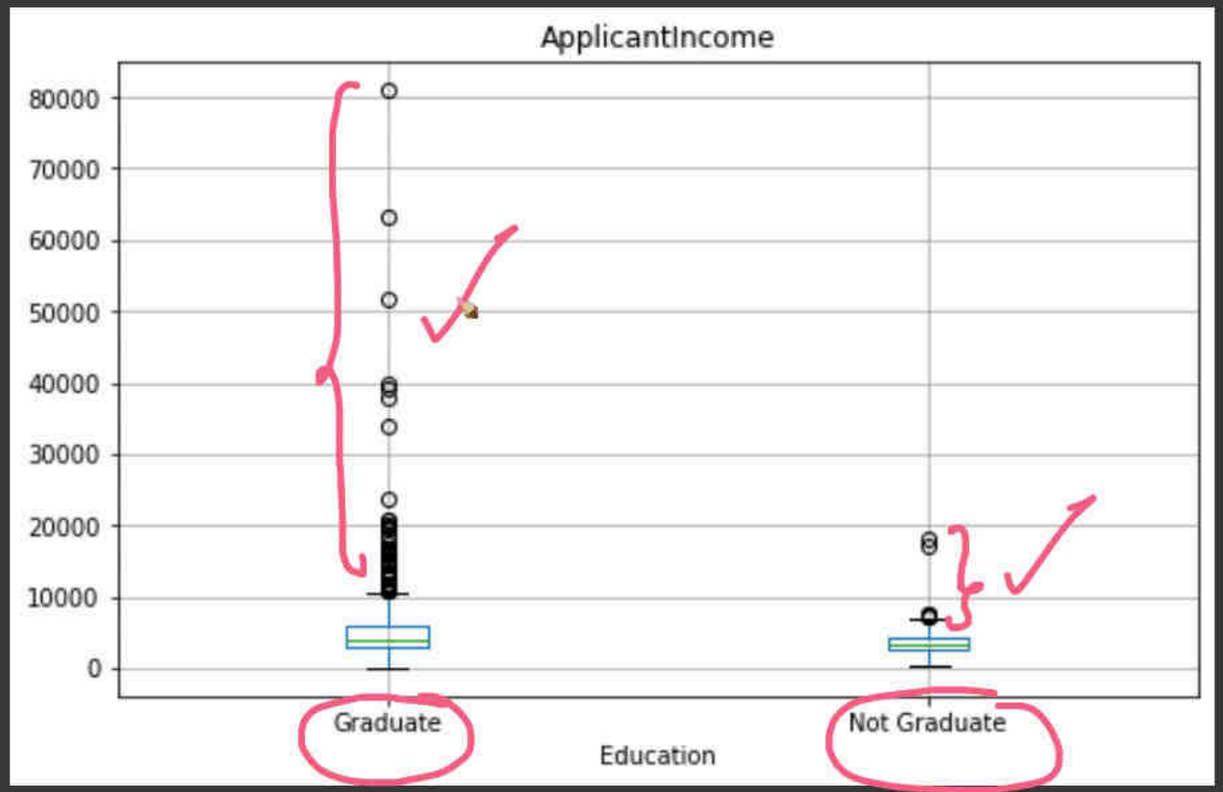



```
[ ]  #Slice this data by Education
```


X_1
X_2
log
log
t-test
$\log(\mu_1)$
$= \log(\mu_2)$
50 / 50

```
[ ] #slice this data by Education
```

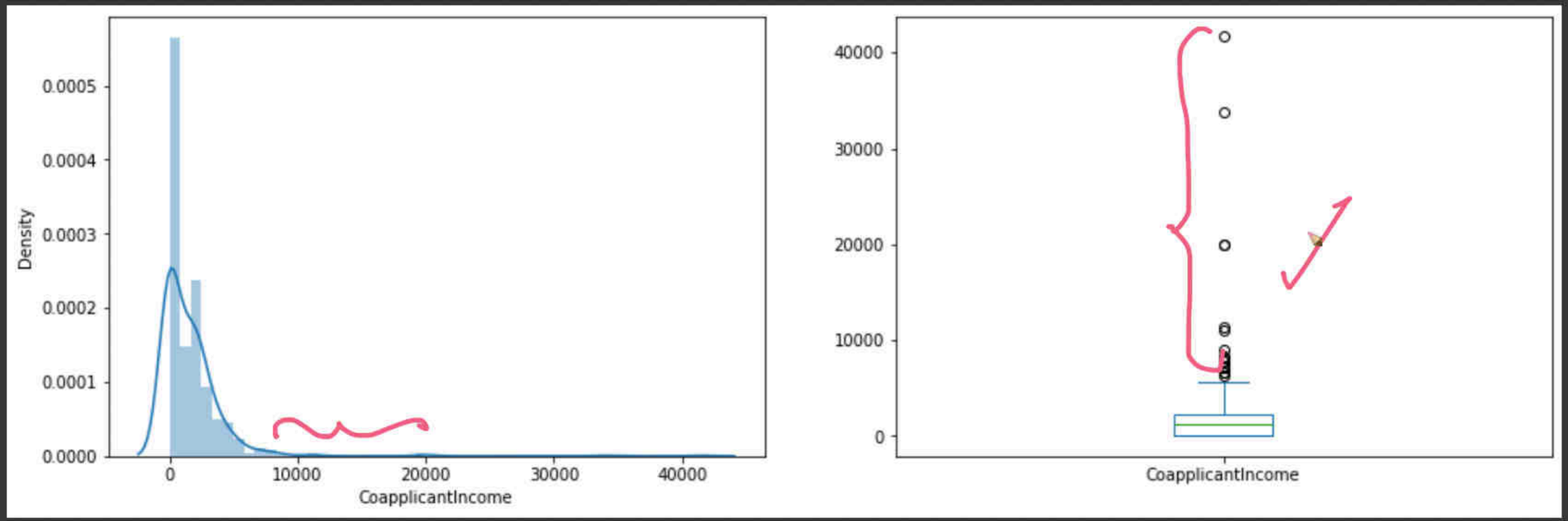
```
[25] data.boxplot(column='ApplicantIncome', by="Education", figsize=(8,5))
     plt.suptitle("")
     plt.show()
```



Inc

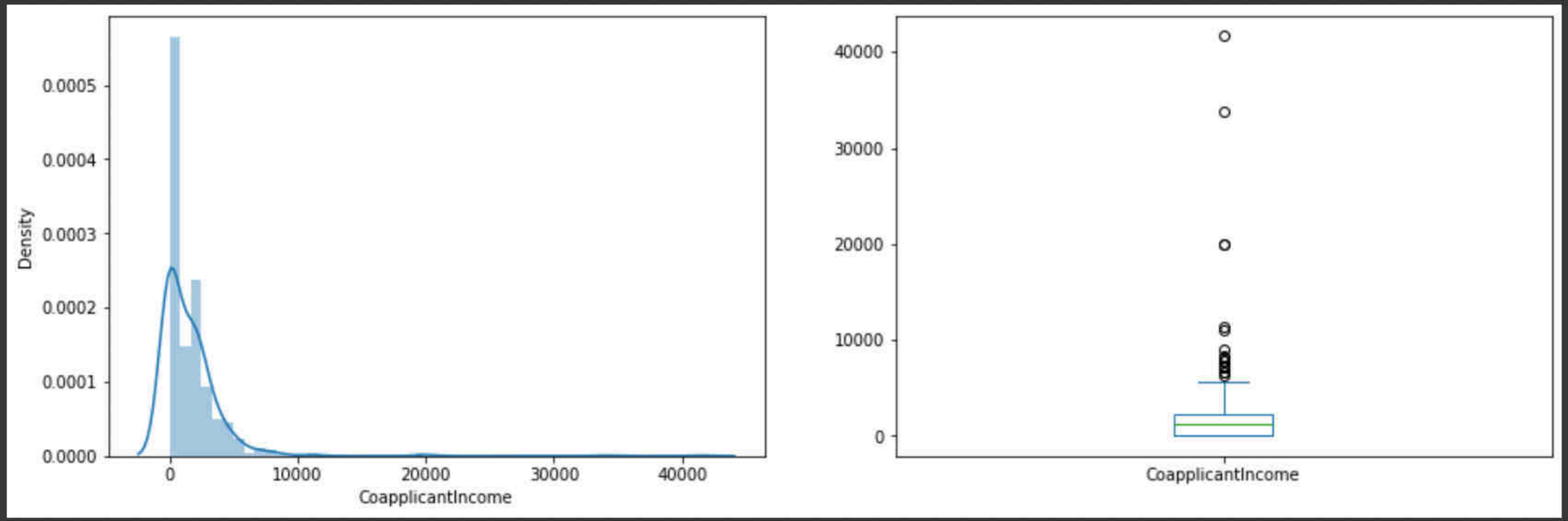
```
[26] #co-applicant income
```

```
#co-applicant income
plt.subplot(121)
sns.distplot(data["CoapplicantIncome"])

plt.subplot(122)
data["CoapplicantIncome"].plot.box(figsize=(16,5))
plt.show()
```



```
#co-applicant income
plt.subplot(121)
sns.distplot(data["CoapplicantIncome"])

plt.subplot(122)
data["CoapplicantIncome"].plot.box(figsize=(16,5))
plt.show()
```

DINK



Chrome File Edit View History Bookmarks Profiles Tab Window Help 10:31

EDA_FE.ipynb - Colaboratory

colab.research.google.com/drive/1_P-nKswaKnx5v77lOx3pCxvJve73MvnN#scrollTo=gwIudxlnKDvf

+ Code + Text RAM Disk

```
#Relation between "Loan_Status" and "Income"
```

```
[27] data.groupby("Loan_Status").mean()['ApplicantIncome']
```

```
Loan_Status
N    5446.078125
Y    5384.068720
Name: ApplicantIncome, dtype: float64
```

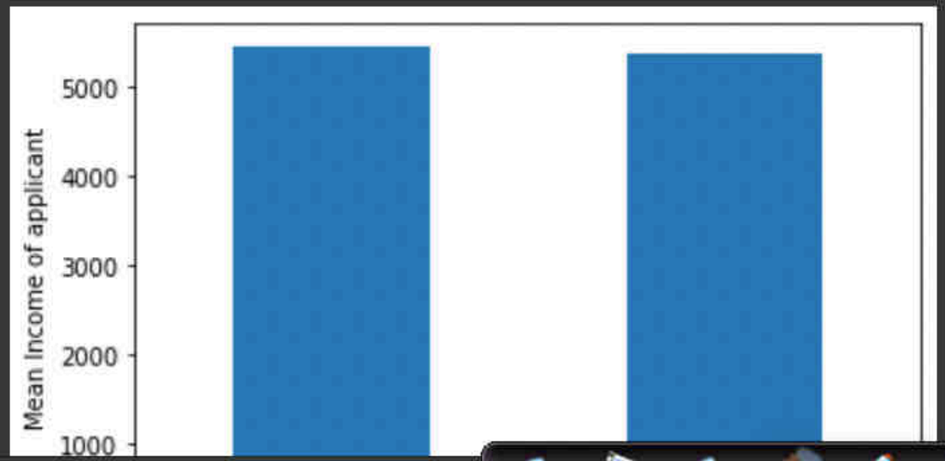
M1

Mean-inc where Status = Y

Mean-inc where Status = N

```
data.groupby("Loan_Status").mean()['ApplicantIncome'].plot.bar()
plt.ylabel("Mean Income of applicant")
plt.show()
```

M2



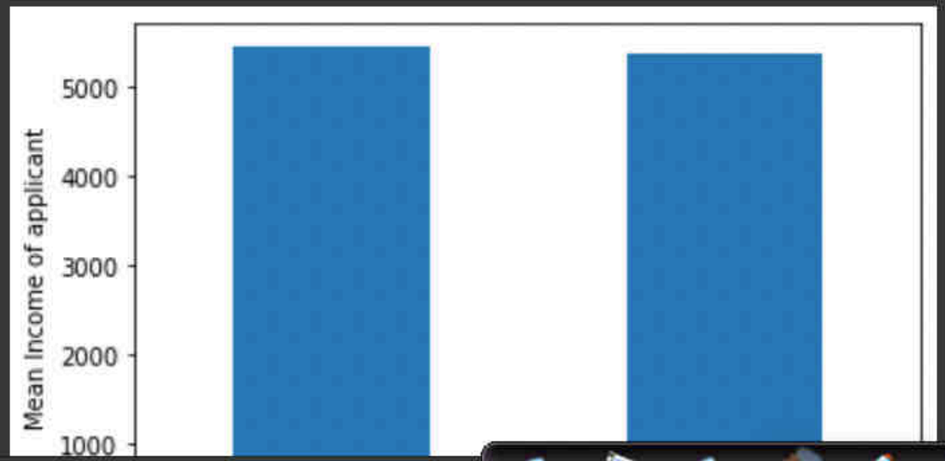
54 / 54

```
#Relation between "Loan_Status" and "Income"
```

```
[27] data.groupby("Loan_Status").mean()['ApplicantIncome']
```

```
Loan_Status
N    5446.078125
Y    5384.068720
Name: ApplicantIncome, dtype: float64
```

```
data.groupby("Loan_Status").mean()['ApplicantIncome'].plot.bar()
plt.ylabel("Mean Income of applicant")
plt.show()
```



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

EDA_FE.ipynb - Colaboratory

colab.research.google.com/drive/1_P-nKswaKnx5v77lOx3pCxvJve73MvnN#scrollTo=gwIudxlnKDvf

+ Code + Text

RAM
Disk

```
#Relation between "Loan_Status" and "Income"
```

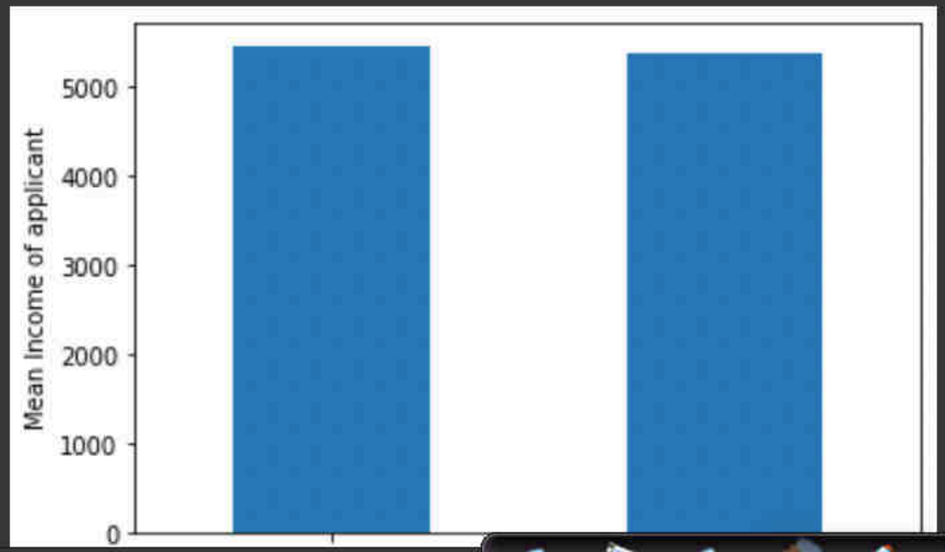
```
[27] data.groupby("Loan_Status").mean()['ApplicantIncome']
```

```
Loan_Status
N    5446.078125
Y    5384.068720
Name: ApplicantIncome, dtype: float64
```

n_1 & n_2 are large

{$M_1 - M_2$ is very small}

```
[28] data.groupby("Loan_Status").mean()['ApplicantIncome'].plot.bar()
     plt.ylabel("Mean Income of applicant")
     plt.show()
```



Ques: error in this analysis

outliers

56 / 56

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

EDA_FE.ipynb - Colaboratory

colab.research.google.com/drive/1_P-nKswaKnx5v77lOx3pCxvJve73MvnN#scrollTo=gwIudxlnKDvf

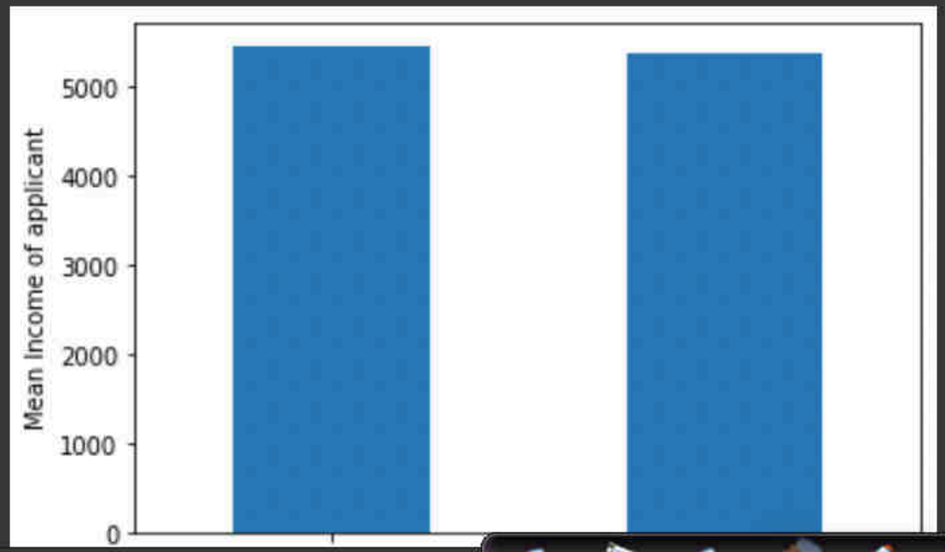
+ Code + Text RAM Disk

```
#Relation between "Loan_Status" and "Income"
```

```
[27] data.groupby("Loan_Status").mean()['ApplicantIncome']
```

```
Loan_Status
N    5446.078125
Y    5384.068720
Name: ApplicantIncome, dtype: float64
```

```
[28] data.groupby("Loan_Status").mean()['ApplicantIncome'].plot.bar()
     plt.ylabel("Mean Income of applicant")
     plt.show()
```



task! IQR remove outliers

Mean$_1$ & Mean$_2$

57 / 57

Chrome File Edit View History Bookmarks Profiles Tab Window Help 10:36

EDA_FE.ipynb - Colaboratory

colab.research.google.com/drive/1_P-nKswaKnx5v77lOx3pCxvJve73MvnN#scrollTo=gwIudxlnKDvf

+ Code + Text

RAM
Disk

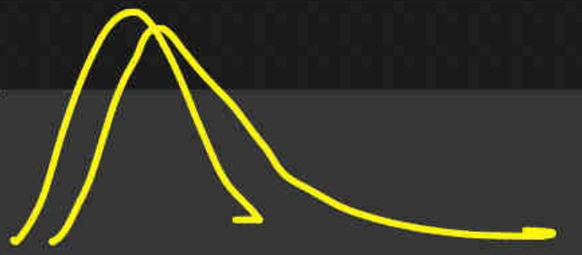
```
#Relation between "Loan_Status" and "Income"
```

```
[27] data.groupby("Loan_Status").mean()['ApplicantIncome']
```

```
Loan_Status
N    5446.078125
Y    5384.068720
Name: ApplicantIncome, dtype: float64
```

→ 95% C.I

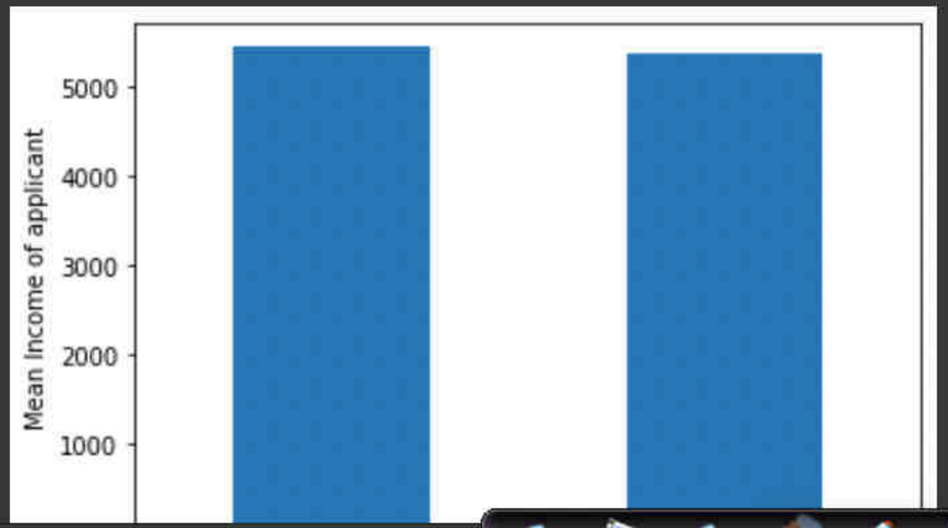
PDF:



log-transform (KS) t-test

```
data.groupby("Loan_Status").mean()['ApplicantIncome'].plot.bar()
plt.ylabel("Mean Income of applicant")
plt.show()
```

CDF: →



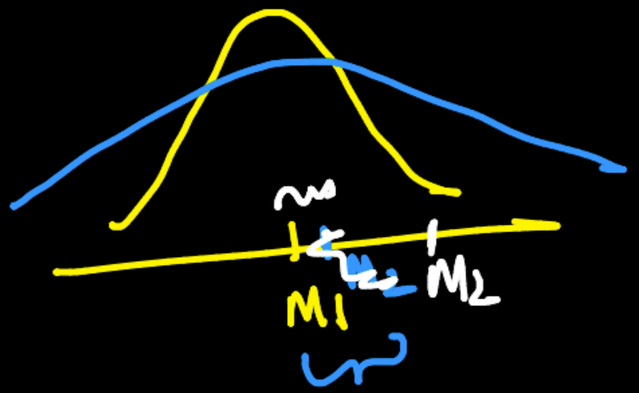
QQ-plot: →

Q_1, Q_2, Q_3 →

58 / 58

Q

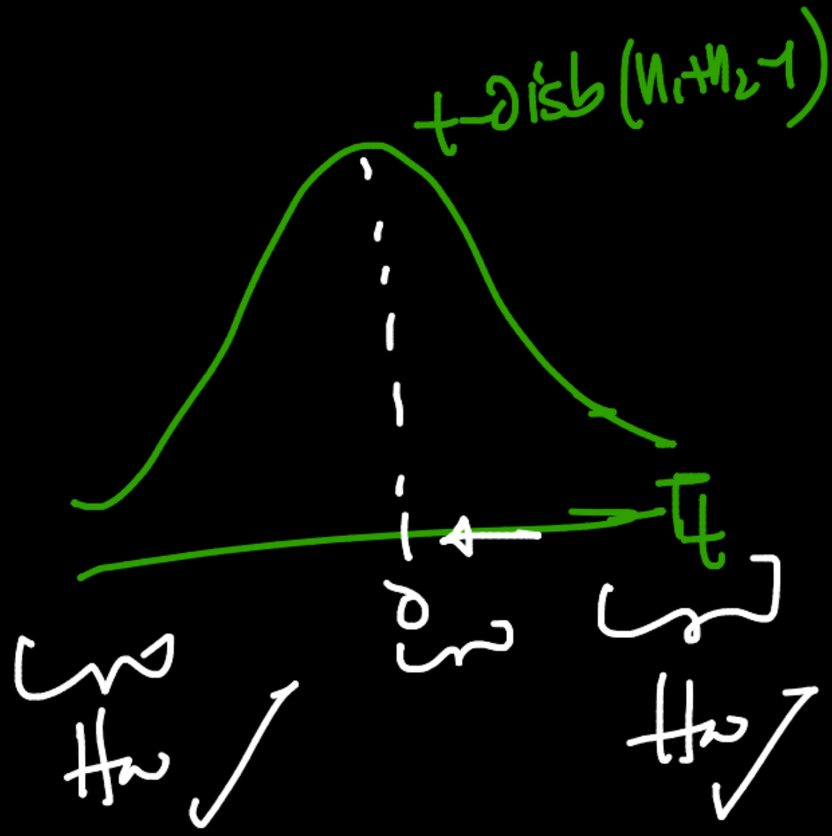
$|M_1 - M_2|$



Sig-level
Conf-level

+ve

$$T_t = \frac{M_1 - M_2}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}}$$



95% - C.I

→ t-test

ks-test

outlier - analysis

} Inc & loan

Cat

{ Race/ethnicity χ Cat loan approval

χ^2-test } ✓

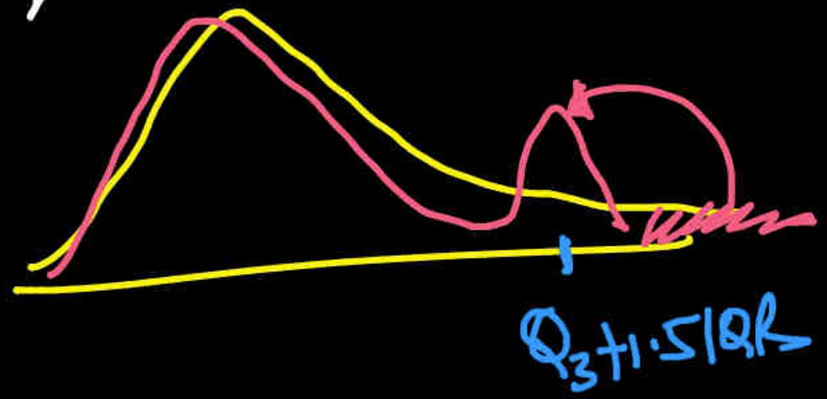
✓ [Harvard (Asian-Amer)

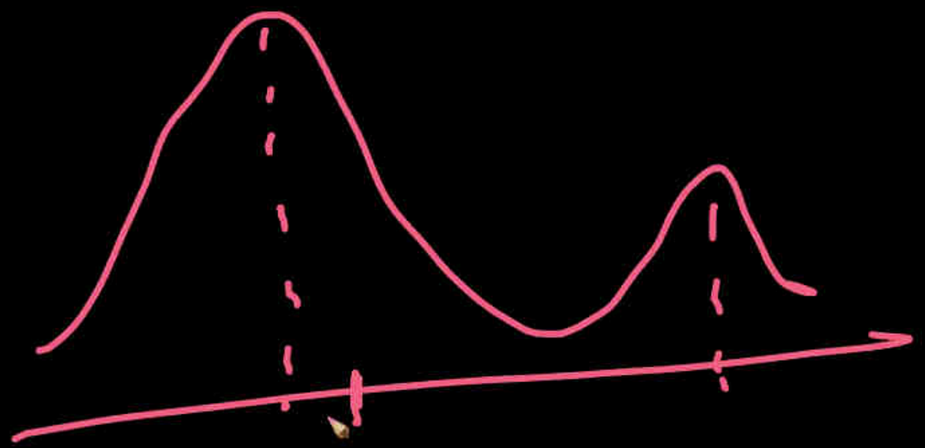
Q

Val > Q_3 + 1.5 IQR

↳ Q_3 + 1.5 IQR X

log/box-cox





multi-modal

Q

INC → loan approval

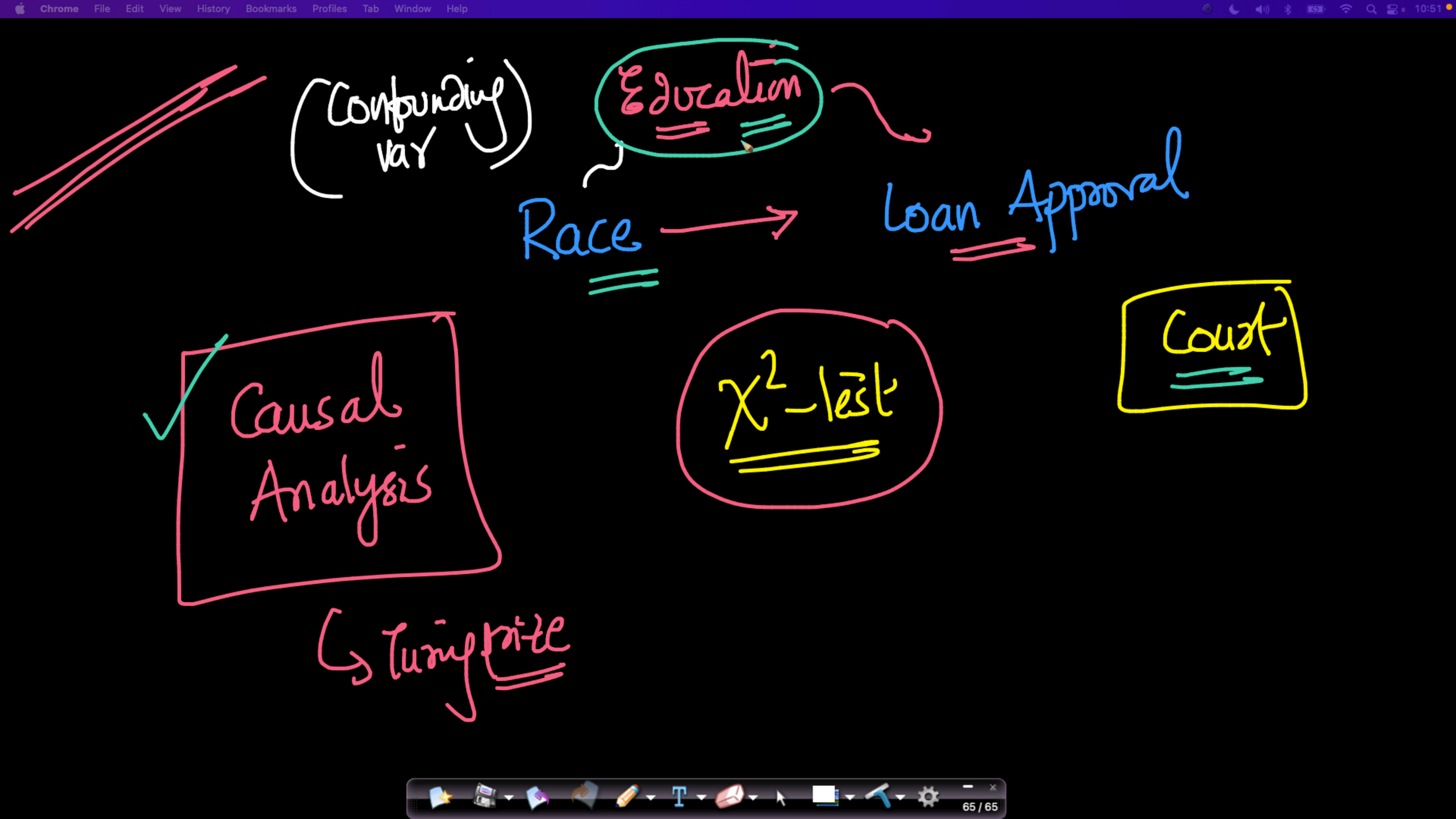
$H_0: \mu_1 \neq \mu_2$

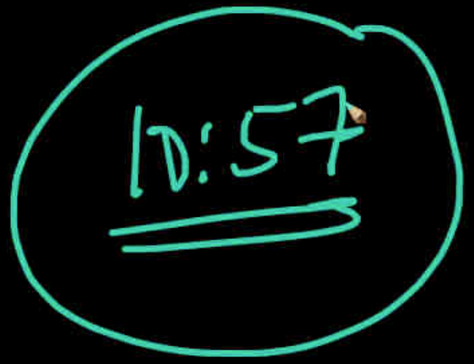
$H_a: \mu_1 = \mu_2$

T_t under H_0 → Do not know

↓

Cannot use Hypothesis


(Confounding var)
Education
Race
Loan Approval
Causal Analysis
χ^2-test
Count
Turnitin


10:57

Feature Engineering

Simple Feature Engineering

binning basically

```
# Feature binning: income
bins=[0,2500,4000,6000,81000]
group=['Low','Average','High', 'Very high']
data['Income_bin']= pd.cut(data['ApplicantIncome'],bins,labels=group)
```

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

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_
0	Male	No	0	Graduate	No	5849	0.0	NaN	
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	
4	Male	No	0	Graduate	No	6000	0.0	141.0	

Simple Feature Engineering

0 – 2499

2500 – 3999

```
# Feature binning: income
bins=[0,2500,4000,6000,81000]
group=['Low','Average','High', 'Very high']
data['Income_bin']= pd.cut(data['ApplicantIncome'],bins,labels=group)
```

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

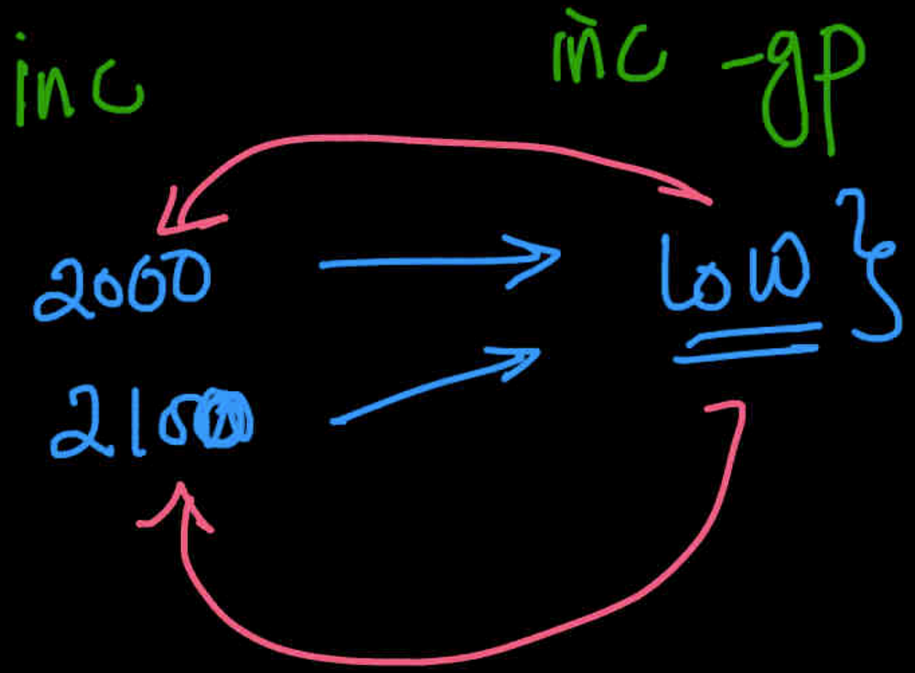
	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_
0	Male	No	0	Graduate	No	5849	0.0	NaN	
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	
4	Male	No	0	Graduate	No	6000	0.0	141.0	

Simple Feature Engineering

```
# Feature binning: income
bins=[0,2500,4000,6000,81000]
group=['Low','Average','High', 'Very high']
data['Income_bin']= pd.cut(data['ApplicantIncome'],bins,labels=group)
```

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

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_
0	Male	No	0	Graduate	No	5849	0.0	NaN	
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	
4	Male	No	0	Graduate	No	6000	0.0	141.0	



non-invertable

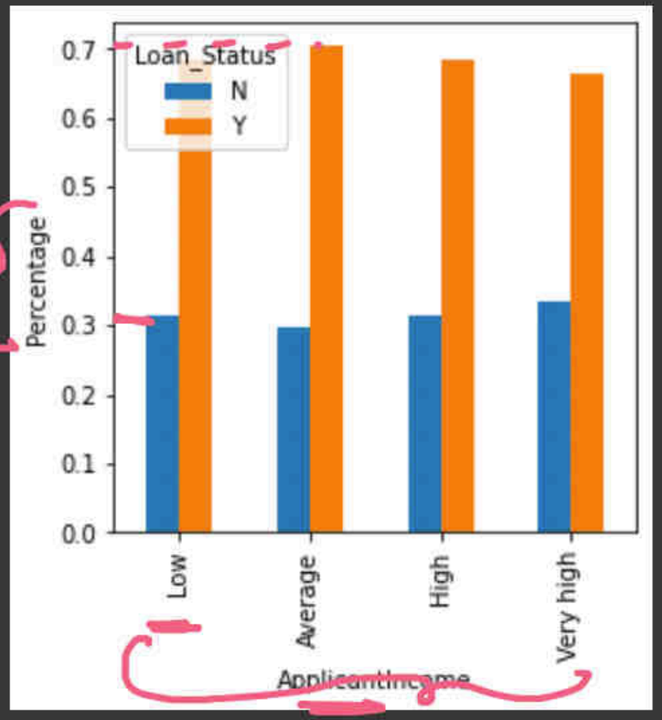
```
[31] #observed
     pd.crosstab(data["Income_bin"],data["Loan_Status"])
```

Income_bin	N	Y
Low	34	74
Average	67	159
High	45	98
Very high	46	91

χ^2-test

```
[32] Income_bin = pd.crosstab(data["Income_bin"],data["Loan_Status"])

     Income_bin.div(Income_bin.sum(axis=1),axis=0).plot(kind="bar",figsize=(4,4))
     plt.xlabel("ApplicantIncome")
     plt.ylabel("Percentage")
     plt.show()

     #It can be inferred that Applicant income does not affect the chances of loan approval. Which seems wrong as per ou
```

```
[32] Income_bin.div(Income_bin.sum(axis=1),axis=0).plot(kind="bar",figsize=(4,4))
     plt.xlabel("ApplicantIncome")
     plt.ylabel("Percentage")
     plt.show()

     #It can be inferred that Applicant income does not affect the chances of loan approval. Which seems wrong as per ou
```

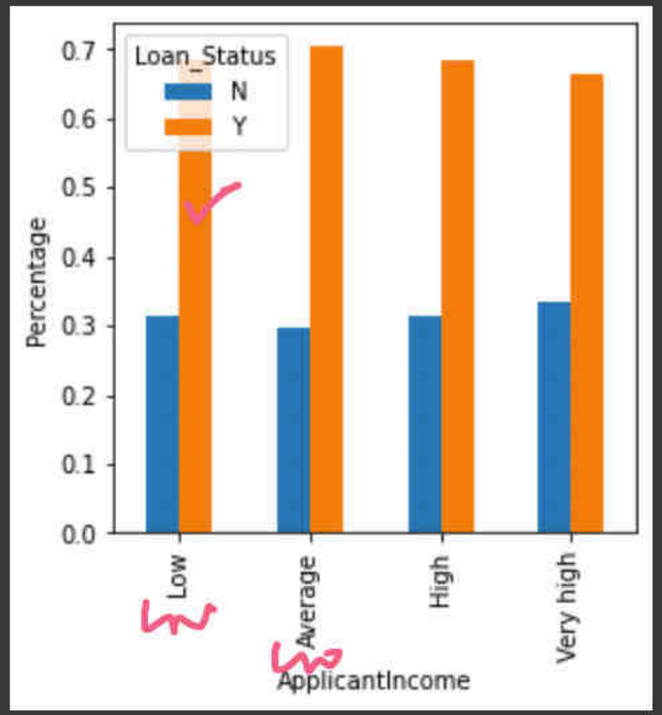


Insightful

INC → INC-bias

```
[33] #co-applicant income
```

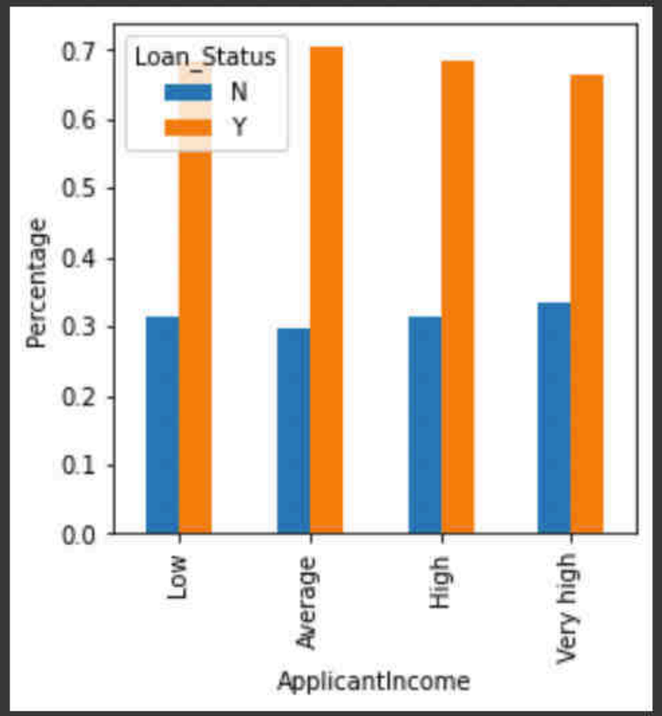
```
[32] Income_bin.div(Income_bin.sum(axis=1),axis=0).plot(kind="bar",figsize=(4,4))
     plt.xlabel("ApplicantIncome")
     plt.ylabel("Percentage")
     plt.show()

     #It can be inferred that Applicant income does not affect the chances of loan approval. Which seems wrong as per ou
```



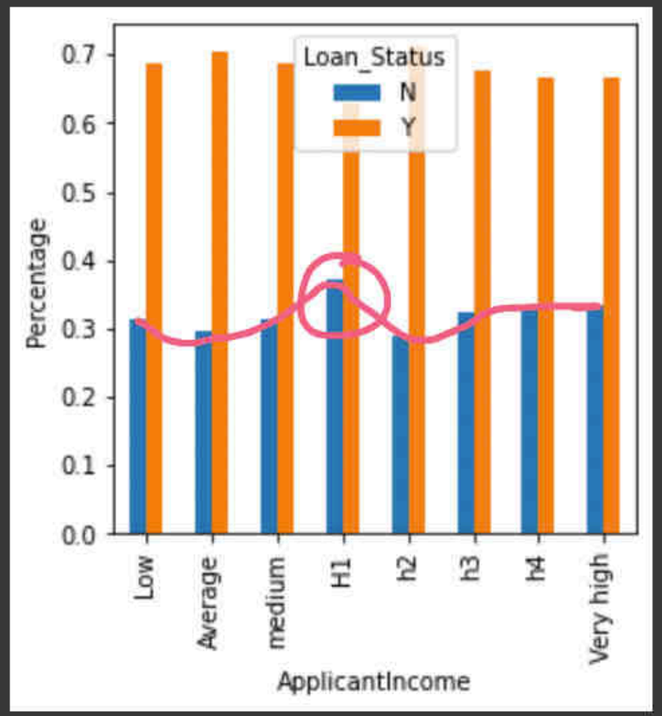
{ INC-bins ✗ loan-status

INC ? loan-status

```
[32] Income_bin.div(Income_bin.sum(axis=1),axis=0).plot(kind="bar",figsize=(4,4))
     plt.xlabel("ApplicantIncome")
     plt.ylabel("Percentage")
     plt.show()

     #It can be inferred that Applicant income does not affect the chances of loan approval. Which seems wrong as per ou
```



```
Income_bin.div(Income_bin.sum(axis=1),axis=0).plot(kind="bar",figsize=(4,4))
plt.xlabel("ApplicantIncome")
plt.ylabel("Percentage")
plt.show()

#It can be inferred that Applicant income does not affect the chances of loan approval. Which seems wrong as per ou
```



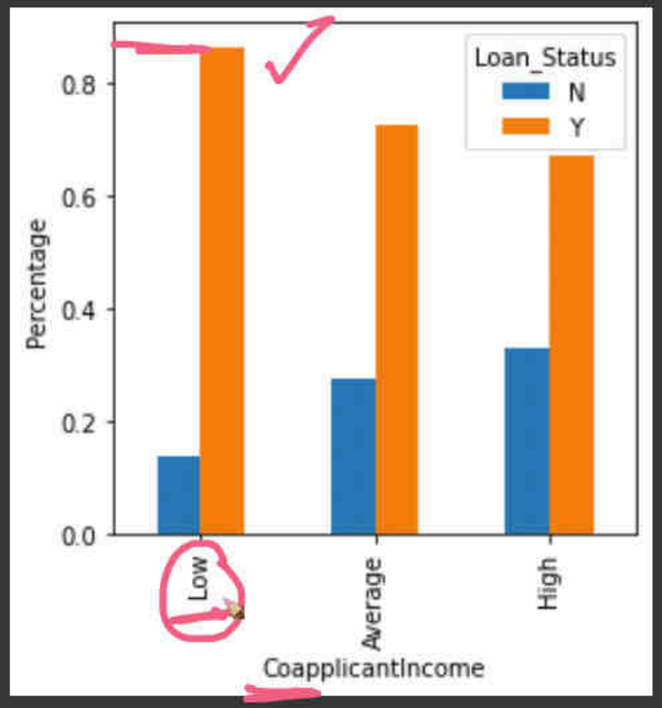
0·3 - 0·33

[104]

Income_bin		
Low	34	74
Average	67	159
medium	45	98
H1	20	34
h2	9	22
h3	13	27
h4	3	6
Very high	1	2

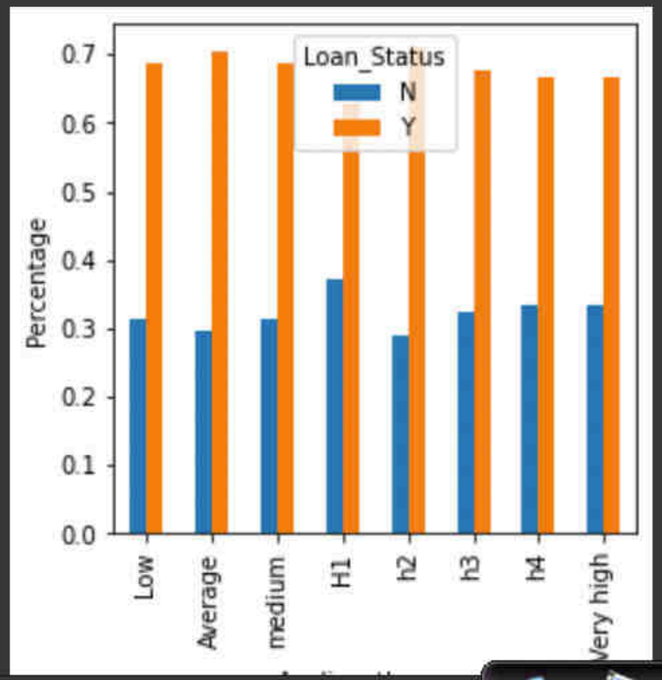
```
Income_bin = pd.crosstab(data["Income_bin"],data["Loan_Status"])

Income_bin.div(Income_bin.sum(axis=1),axis=0).plot(kind="bar",figsize=(4,4))
plt.xlabel("ApplicantIncome")
plt.ylabel("Percentage")
plt.show()

#It can be inferred that Applicant income does not affect the chances of loan approval. Which seems wrong as per ou
```

```
CoapplicantIncome_Bin.div(CoapplicantIncome_Bin.sum(axis = 1),axis=0).plot(kind='bar',figsize=(4,4))
plt.xlabel("CoapplicantIncome")
plt.ylabel("Percentage")
plt.show()

## What's the problem here? Why co-applicant having low income is getting maximum loan approved?
```

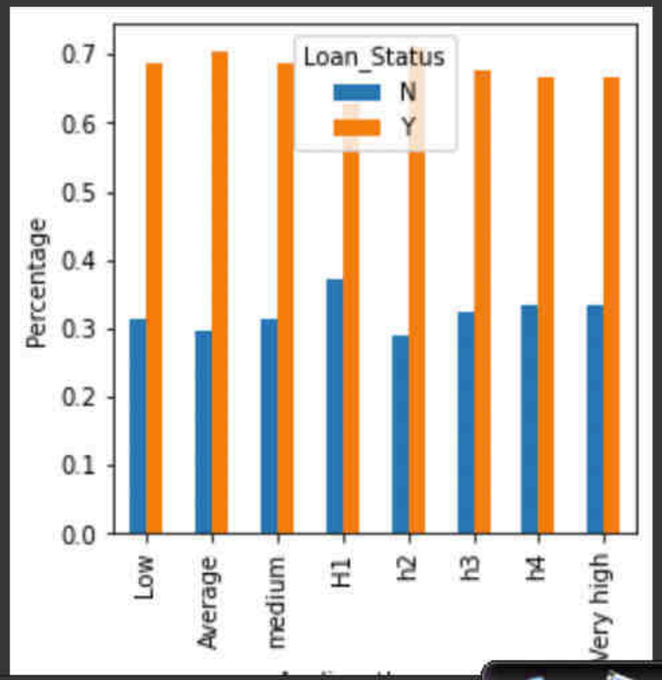



```
[36] data['CoapplicantIncome'].value_counts().head()
```

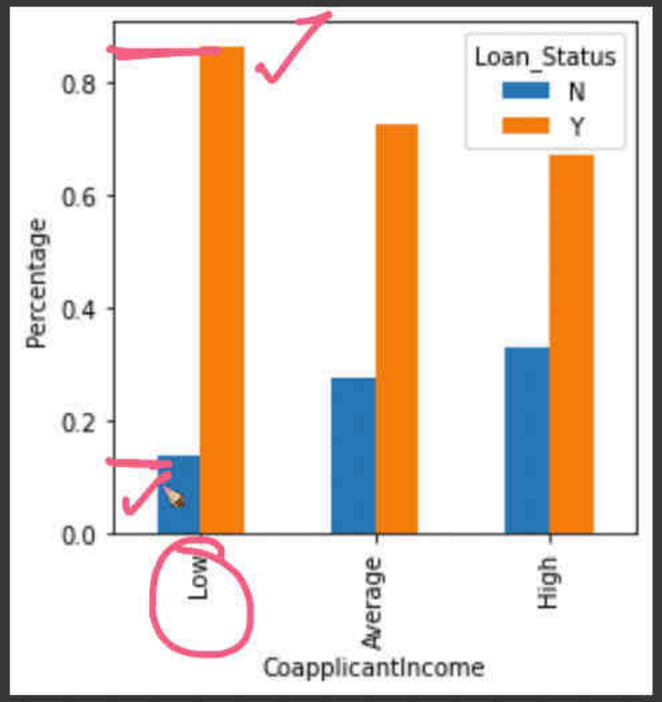
```
Income_bin = pd.crosstab(data["Income_bin"],data["Loan_Status"])

Income_bin.div(Income_bin.sum(axis=1),axis=0).plot(kind="bar",figsize=(4,4))
plt.xlabel("ApplicantIncome")
plt.ylabel("Percentage")
plt.show()

#It can be inferred that Applicant income does not affect the chances of loan approval. Which seems wrong as per ou
```



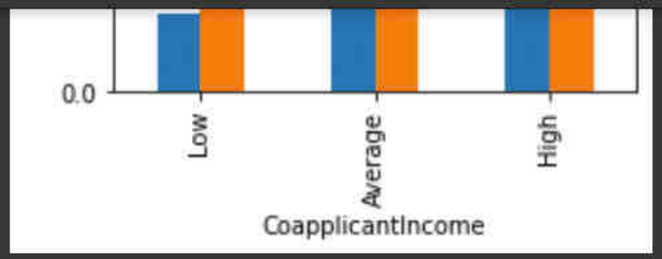
```
Income_bin = pd.crosstab(data["Income_bin"],data["Loan_Status"])

Income_bin.div(Income_bin.sum(axis=1),axis=0).plot(kind="bar",figsize=(4,4))
plt.xlabel("ApplicantIncome")
plt.ylabel("Percentage")
plt.show()

#It can be inferred that Applicant income does not affect the chances of loan approval. Which seems wrong as per ou
```



```
CoapplicantIncome_Bin = pd.crosstab(data["CoapplicantIncome_bin"],data["Loan_Status"])
CoapplicantIncome_Bin.div(CoapplicantIncome_Bin.sum(axis = 1),axis=0).plot(kind='bar',figsize=(4,4))
plt.xlabel("CoapplicantIncome")
plt.ylabel("Percentage")
plt.show()

## What's the problem here? Why co-applicant having low income is getting maximum loan approved?
```



[108]




```
[36] data['CoapplicantIncome'].value_counts().head()
```

```
0.0       273
2500.0      5
2083.0      5
1666.0      5
2250.0      3
Name: CoapplicantIncome, dtype: int64
```

```
[37] # New feature:  total household income
     data["TotalIncome"] = data["ApplicantIncome"] + data["CoapplicantIncome"]
```

```
[38] bins = [0,2500,4000,6000,81000]
     group = ['Low','Average','High','Very High']
     data["TotalIncome_bin"] = pd.cut(data["TotalIncome"],bins,labels=group)
```

```
[39] pd.crosstab(data["TotalIncome_bin"], data["Loan_Status"])
```

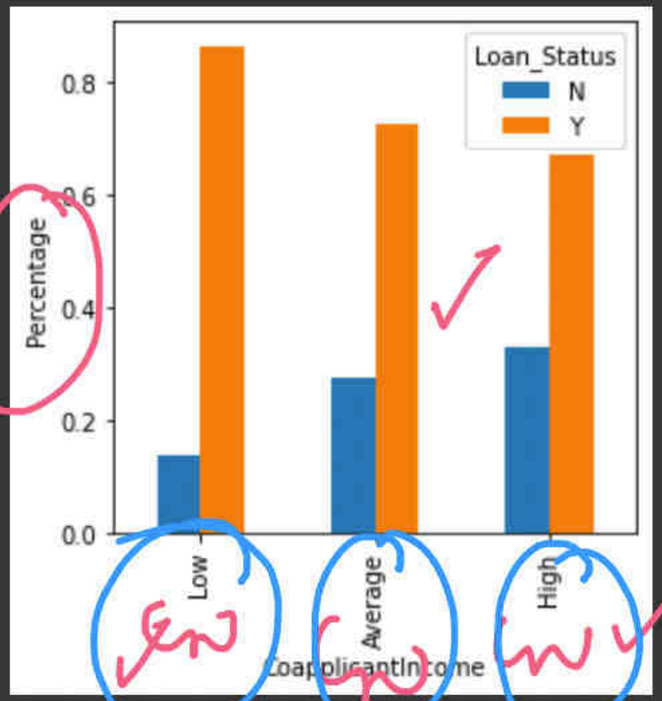
Chrome File Edit View History Bookmarks Profiles Tab Window Help 11:19

EDA_FE.ipynb - Colaboratory | pandas.DataFrame.div — pand

colab.research.google.com/drive/1_P-nKswaKnx5v77lOx3pCxvJve73MvnN#scrollTo=EcFsZRhLK1DI

+ Code + Text RAM Disk

[108]



percentages / ratios / probs

num / den

[36] data['CoapplicantIncome'].value_counts().head()

```
0.0       273
2500.0      5
2083.0      5
1666.0      5
2250.0      3
Name: CoapplicantIncome, dtype: int64
```

83 / 83

```
# New feature:  total household income
data["TotalIncome"] = data["ApplicantIncome"] + data["CoapplicantIncome"]
```

```
[38] bins = [0,2500,4000,6000,81000]
     group = ['Low','Average','High','Very High']
     data["TotalIncome_bin"] = pd.cut(data["TotalIncome"],bins,labels=group)
```

```
[39] pd.crosstab(data["TotalIncome_bin"], data["Loan_Status"])
```

Loan_Status	N	Y
TotalIncome_bin		
Low	14	10
Average	32	87
High	65	159
Very High	81	166

```
[41] TotalIncome = pd.crosstab(data["TotalIncom  axis: Any  a["Loan_Status"])
     TotalIncome.div(TotalIncome.sum(axis = 1),axis=0).plot(kind='bar', figsize=(7,5))
```

Ideas

90-95% will fail

```
# New feature:  total household income
data["TotalIncome"] = data["ApplicantIncome"] + data["CoapplicantIncome"]
```

```
[38] bins = [0,2500,4000,6000,81000]
     group = ['Low','Average','High','Very High']
     data["TotalIncome_bin"] = pd.cut(data["TotalIncome"],bins,labels=group)
```

```
[39] pd.crosstab(data["TotalIncome_bin"], data["Loan_Status"])
```

TotalIncome_bin	N	Y
Low	14	10
Average	32	87
High	65	159
Very High	81	166

```
[41] TotalIncome = pd.crosstab(data["TotalIncome_bin"],data["Loan_Status"])
     TotalIncome.div(TotalIncome.sum(axis = 1),axis=0).plot(kind='bar', figsize=(7,5))
```

```
# New feature:  total household income
data["TotalIncome"] = data["ApplicantIncome"] + data["CoapplicantIncome"]
```

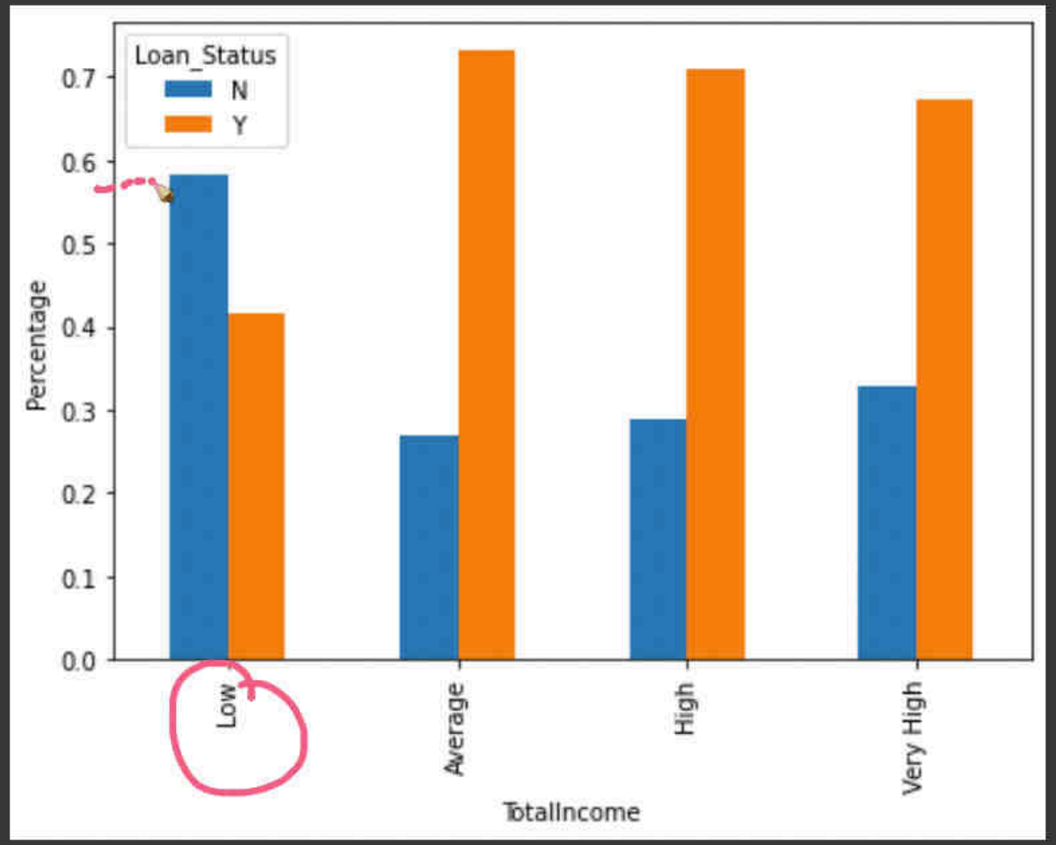
```
[38] bins = [0,2500,4000,6000,81000]
     group = ['Low','Average','High','Very High']
     data["TotalIncome_bin"] = pd.cut(data["TotalIncome"],bins,labels=group)
```

```
[39] pd.crosstab(data["TotalIncome_bin"], data["Loan_Status"])
```

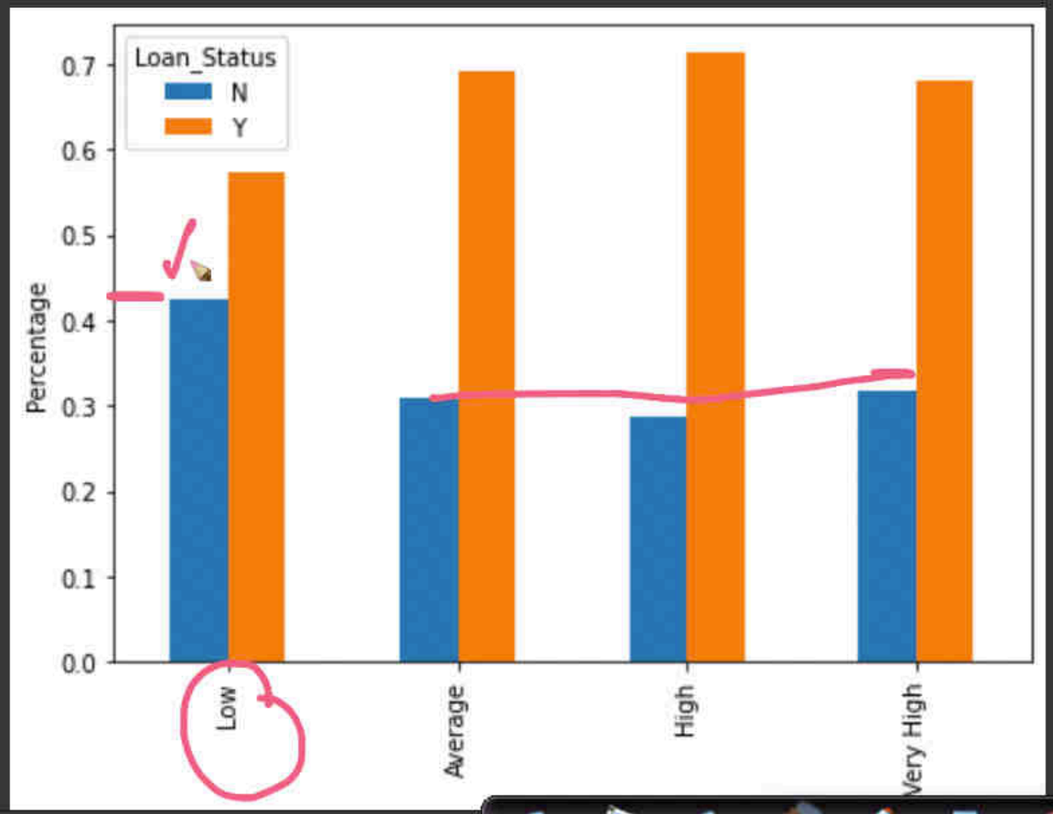
Loan_Status	N	Y
TotalIncome_bin		
Low	14	10
Average	32	87
High	65	159
Very High	81	166

```
[41] TotalIncome = pd.crosstab(data["TotalIncome_bin"],data["Loan_Status"])
     TotalIncome.div(TotalIncome.sum(axis = 1),axis=0).plot(kind='bar', figsize=(7,5))
```

```
[41]  plt.show()

      # Observation: We can see that Proportion of loans getting approved for
      # applicants having low Total_Income is very less as compared to that of applicants
      # with Average, High and Very High Income.
```



```
plt.xlabel("TotalIncome")
plt.ylabel("Percentage")
plt.show()

# Observation: We can see that Proportion of loans getting approved for
# applicants having low Total_Income is very less as compared to that of applicants
# with Average, High and Very High Income.
```



```
[112] bins = [0,3000,5000,8000,81000]
      group = ['Low','Average','High','Very High']
      data["TotalIncome_bin"] = pd.cut(data["TotalIncome"],bins,labels=group)
```

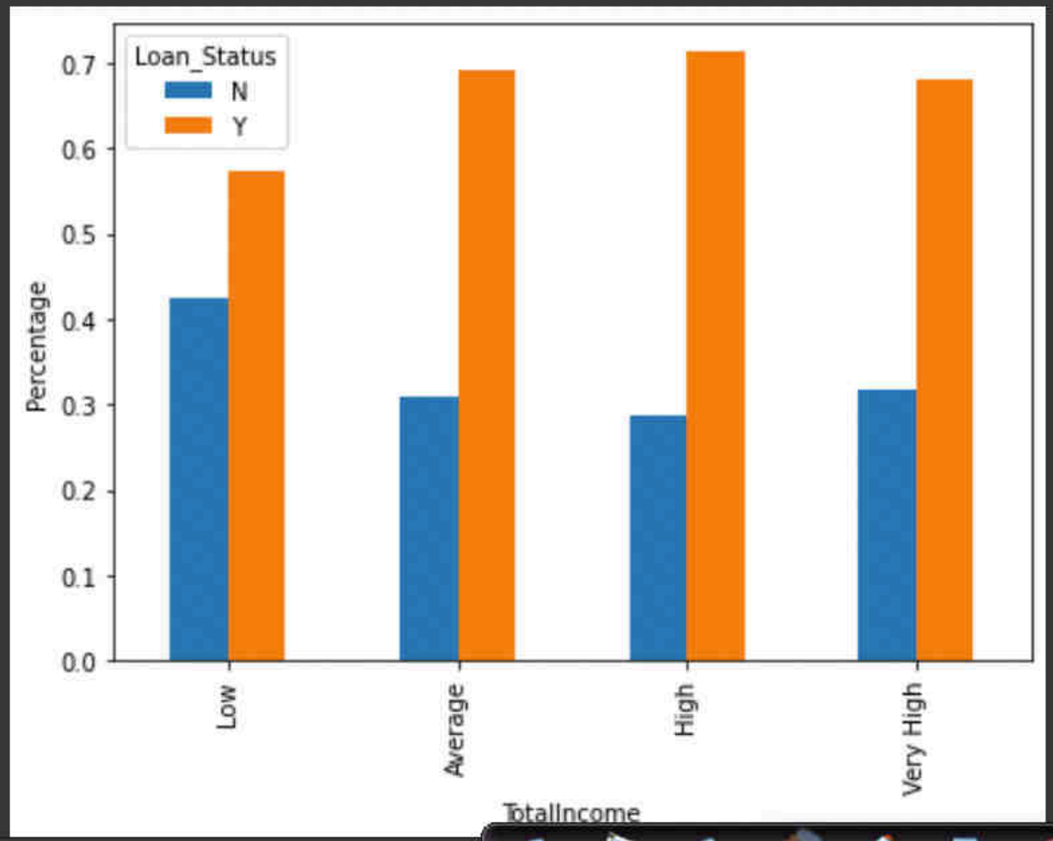
```
[113] pd.crosstab(data["TotalIncome_bin"], data["Loan_Status"])
```

Loan_Status	N	Y
TotalIncome_bin		
Low	20	27
Average	69	154
High	61	151
Very High	42	90

$\frac{20}{47}$; $\frac{27}{47}$ } ✓

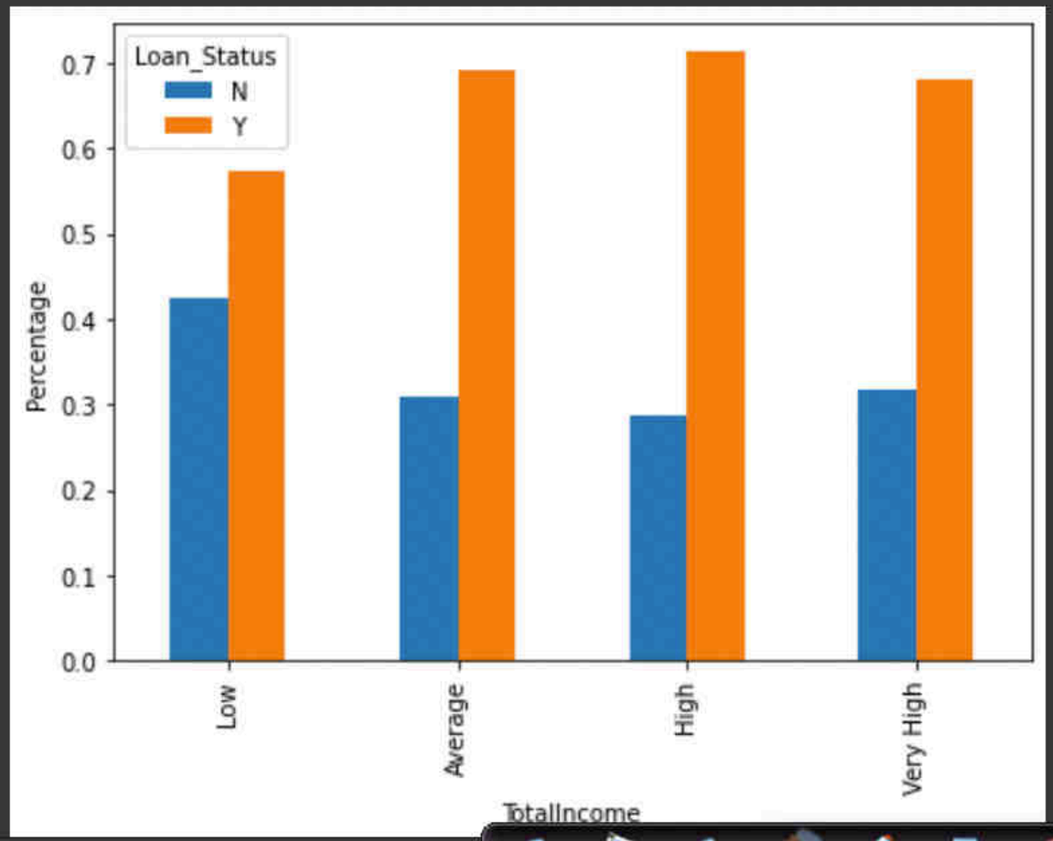
```
TotalIncome = pd.crosstab(data["TotalIncome_bin"],data["Loan_Status"])
TotalIncome.div(TotalIncome.sum(axis = 1),axis=0).plot(kind='bar', figsize=(7,5))
plt.xlabel("TotalIncome")
plt.ylabel("Percentage")
plt.show()
```

```
plt.xlabel("TotalIncome")
plt.ylabel("Percentage")
plt.show()

# Observation: We can see that Proportion of loans getting approved for
# applicants having low Total_Income is very less as compared to that of applicants
# with Average, High and Very High Income.
```



$$f_{new} = f_{inc} + f_{coapp \cdot inc}$$

```
plt.xlabel("TotalIncome")
plt.ylabel("Percentage")
plt.show()

# Observation: We can see that Proportion of loans getting approved for
# applicants having low Total_Income is very less as compared to that of applicants
# with Average, High and Very High Income.
```



Task: Y or N

Q&A

d = #features : 1000

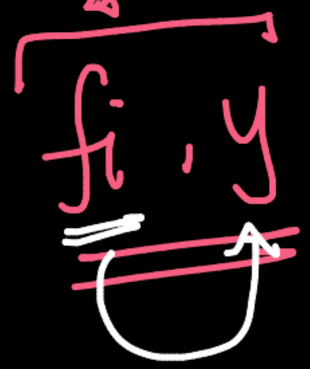
Stats: → > 10's of features

→ DA → ML-models

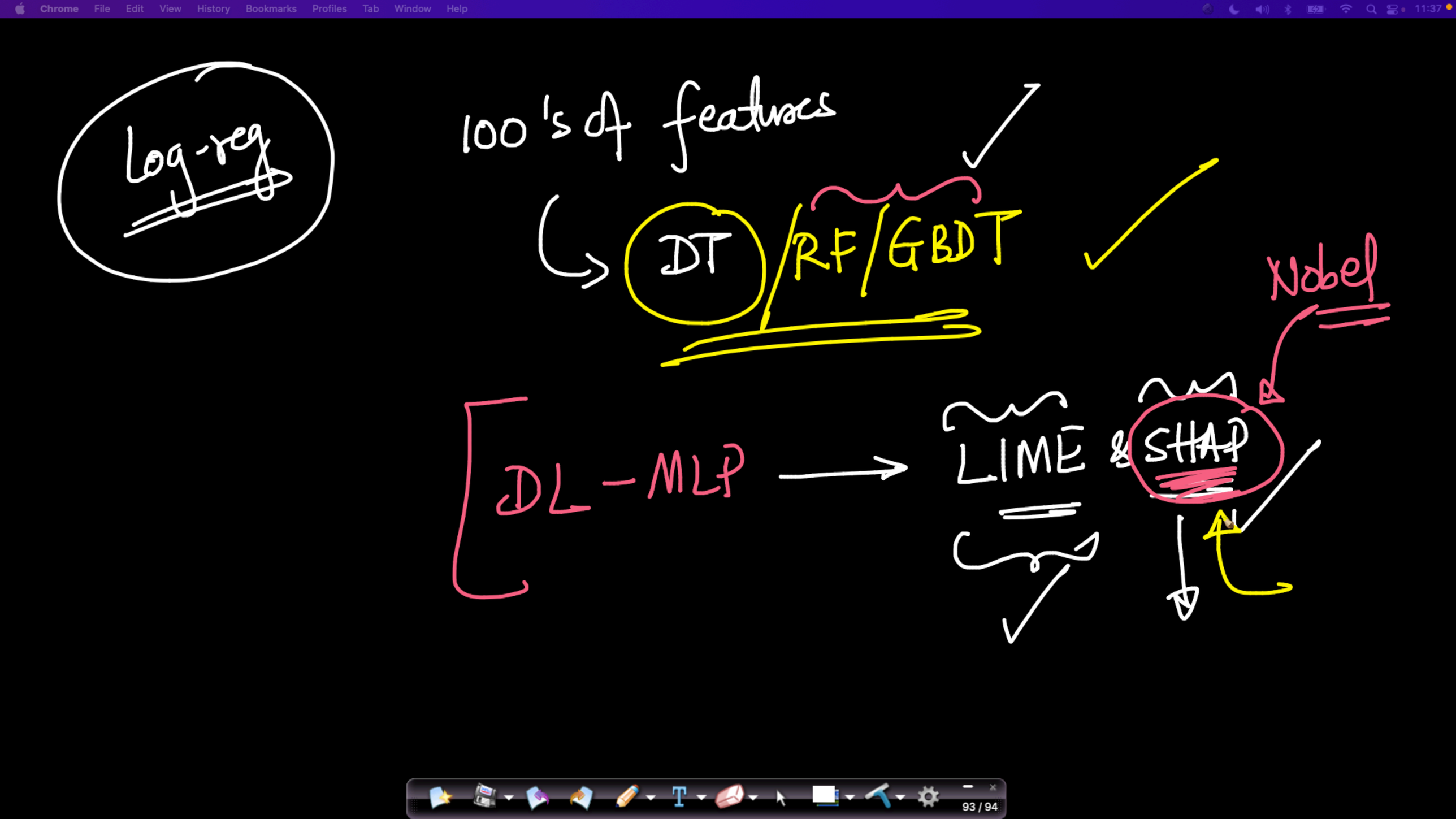
(1000)

cat → numerical

→ SRCC → Univariate analysis



NOT


Log-reg
100's of features
DT
RF/GBDT
DL-MLP
LIME & SHAP
Nobel

Cars24

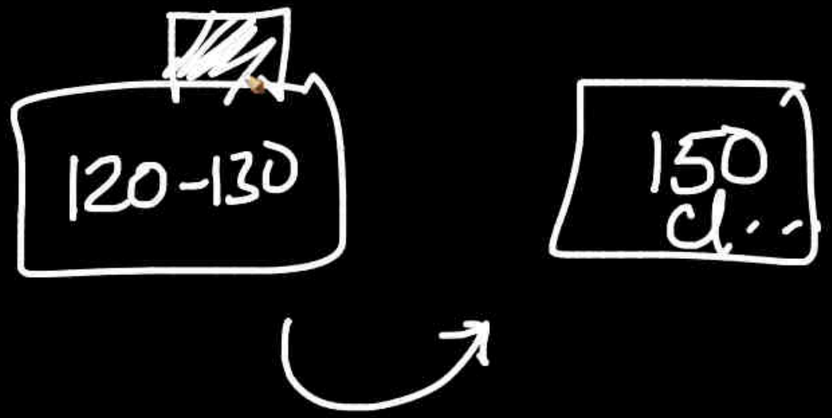
f1 f2 ---- fd y

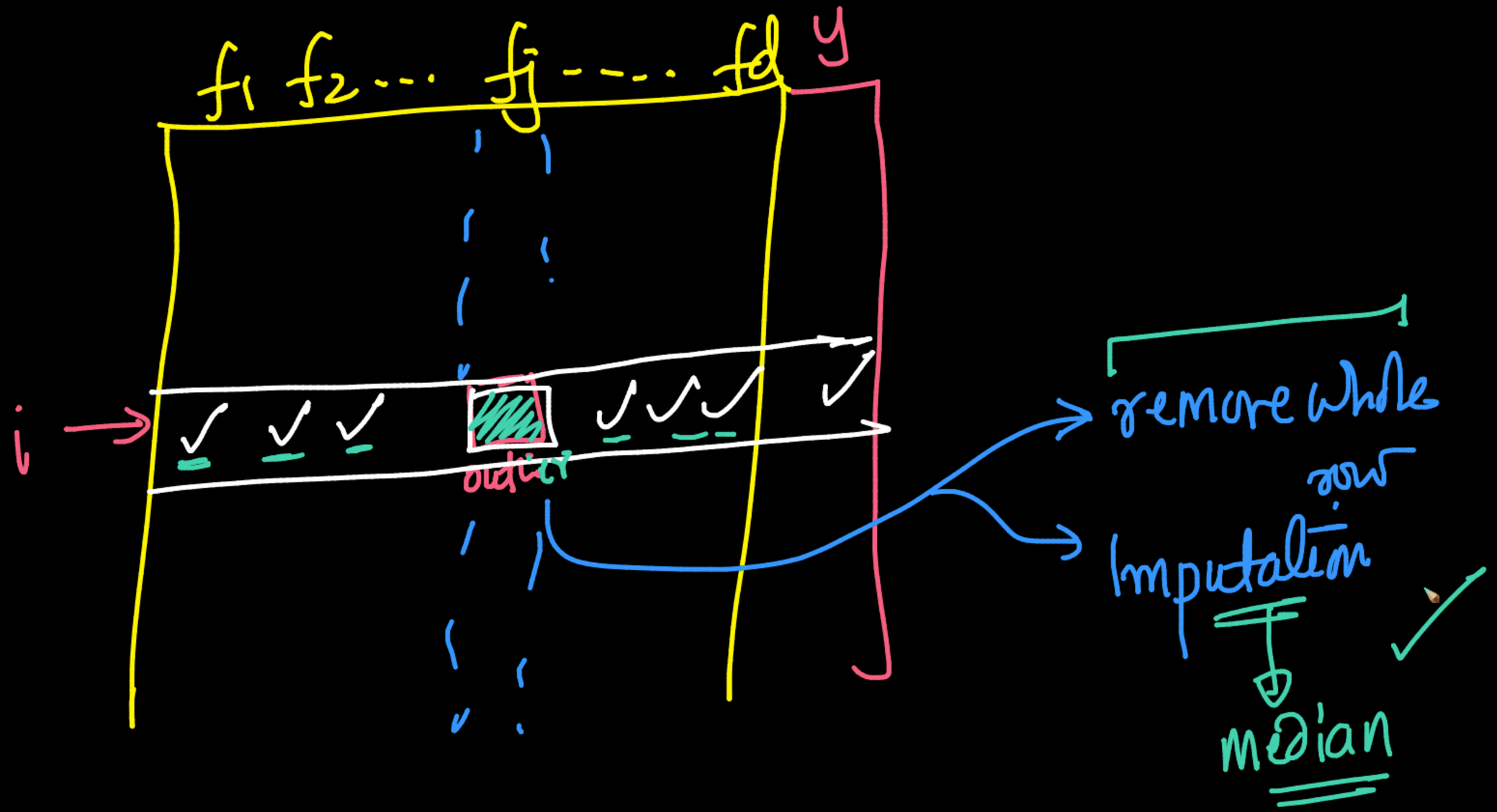
numeric

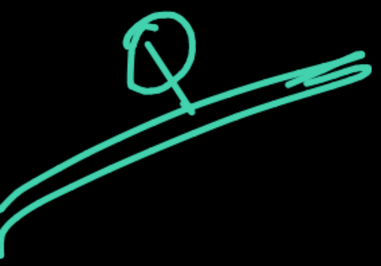
resale value

IQR: hack

✓ domain knowledge (preferred)


120-130
150


f1 f2 ... fj ---- fd
y
i
outlier
remove whole row
Imputation
median



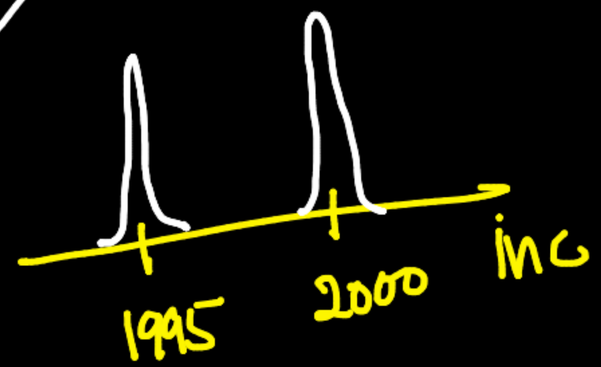
mean_Inc_M = 2000 $

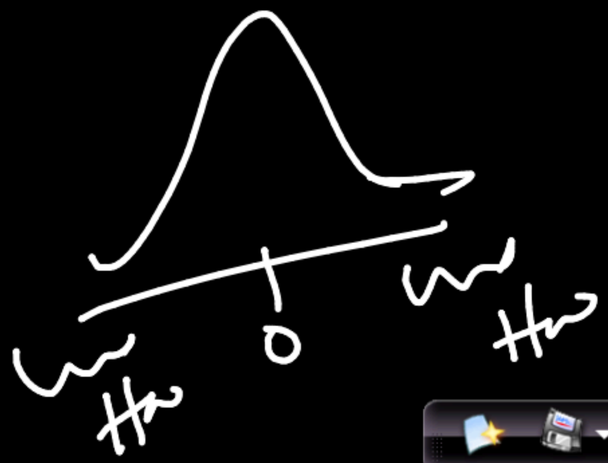
mean_Inc_f = 1995 $

no outliers

$\hat{\sigma}_1$ & $\hat{\sigma}_2$ are V.V small

n_1 & n_2 are large ✓





$\uparrow T_t =$

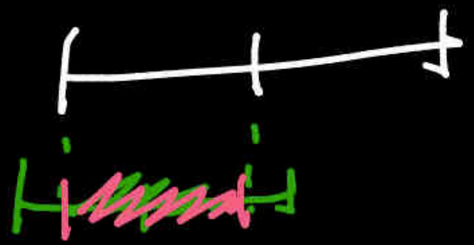
$$\frac{m_1 - M_2}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}}$$

Q

p-val vs α

accept H_0

C.I of Male-inc & female-inc



pandas.DataFrame.corr

DataFrame.corr(*method='pearson', min_periods=1*)

[source]

Compute pairwise correlation of columns, excluding NA/null values.

Parameters: **method** : *{'pearson', 'kendall', 'spearman'} or callable*

Method of correlation:

- pearson : standard correlation coefficient
- kendall : Kendall Tau correlation coefficient
- spearman : Spearman rank correlation
- **callable: callable with input two 1d ndarrays**

  and returning a float. Note that the returned matrix from corr will have 1 along the diagonals and will be symmetric regardless of the callable's behavior.

min_periods : *int, optional*

Minimum number of observations required per pair of columns to have a valid result. Currently only available for Pearson and Spearman correlation.

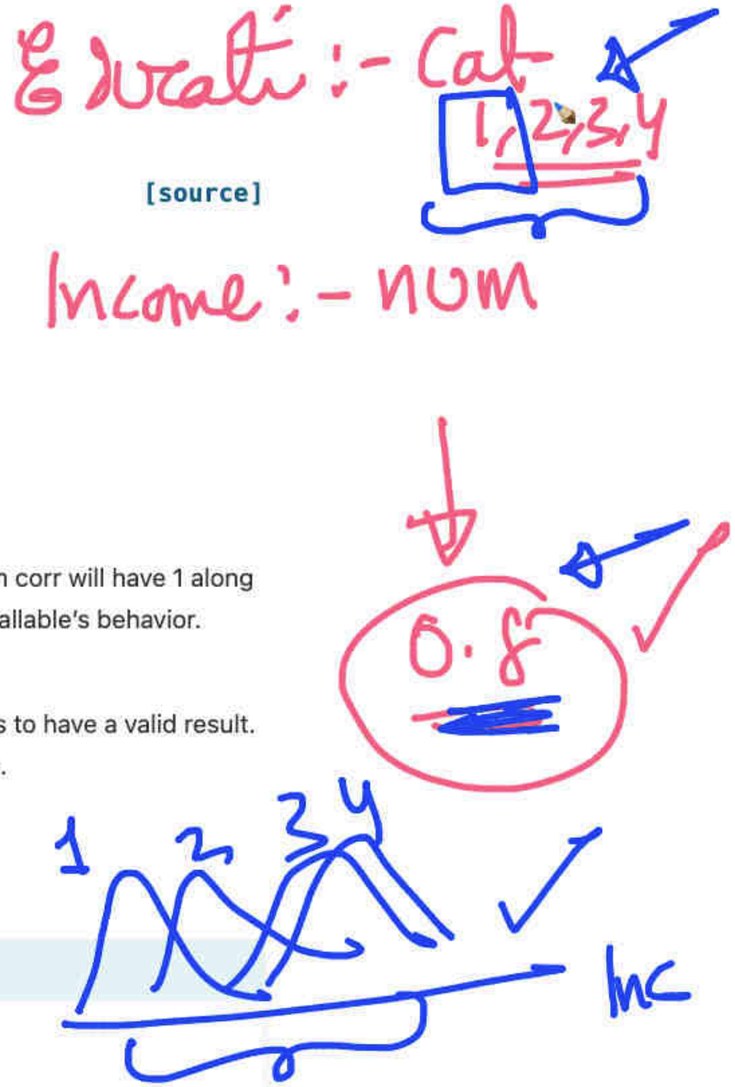
Returns: **DataFrame**

Correlation matrix.

See also

DataFrame.corrwith

Compute pairwise correlation with another DataFrame or Series.



pandas.DataFrame.corr

DataFrame.corr(*method='pearson', min_periods=1*)

Compute pairwise correlation of columns, excluding NA/null values.

Parameters: **method** : *{'pearson', 'kendall', 'spearman'} or callable*

Method of correlation:

- pearson : standard correlation coefficient
- kendall : Kendall Tau correlation coefficient
- spearman : Spearman rank correlation
- **callable: callable with input two 1d ndarrays**

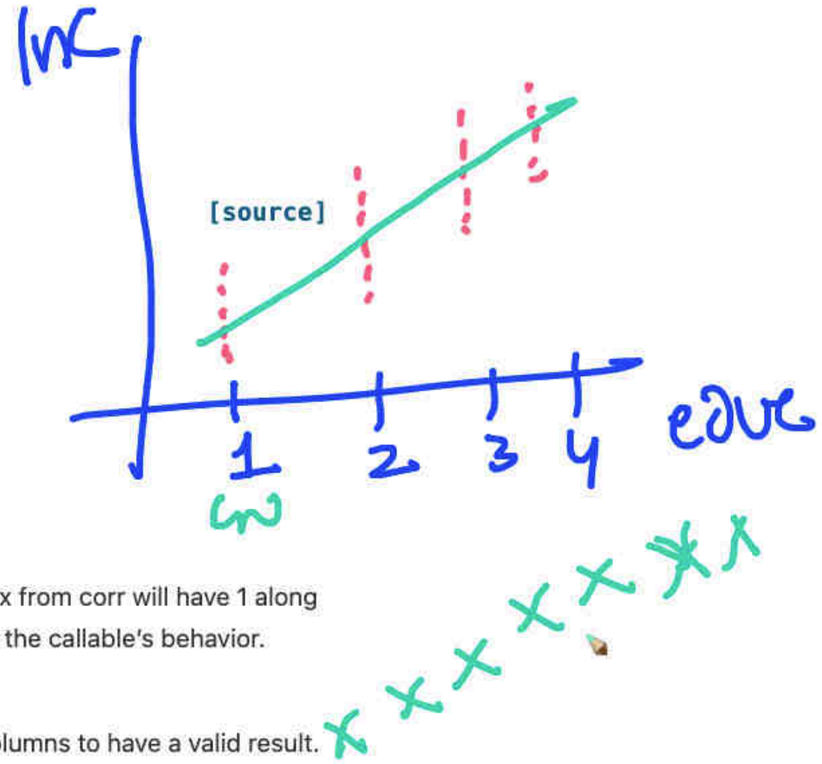
and returning a float. Note that the returned matrix from corr will have 1 along the diagonals and will be symmetric regardless of the callable's behavior.

min_periods : *int, optional*

Minimum number of observations required per pair of columns to have a valid result. Currently only available for Pearson and Spearman correlation.

Returns: **DataFrame**

Correlation matrix.



See also

DataFrame.corrwith

Compute pairwise correlation with another DataFrame or Series.

Education → 1, 2, 3, 4 → Mean (2.5)

Hs B M Phd

Inc → numeric → Mean

Task 7

Q Educati corr with Incomes

Corr.coeff → 0.8

[Cat → Numeric]

χ^2-test ...

Nominal → Gender → M, F, O (most-freq)

Target encoding

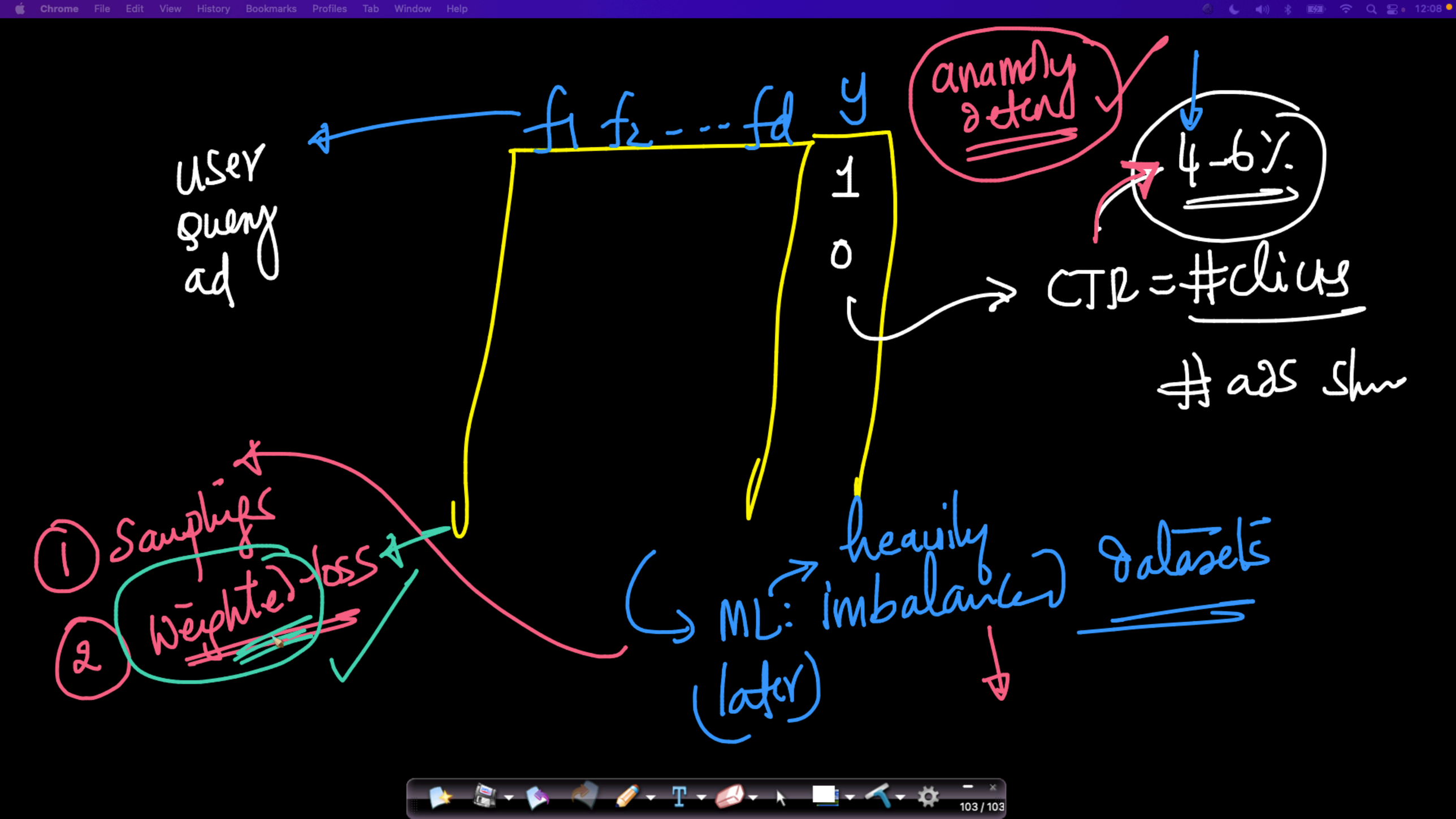
OHE

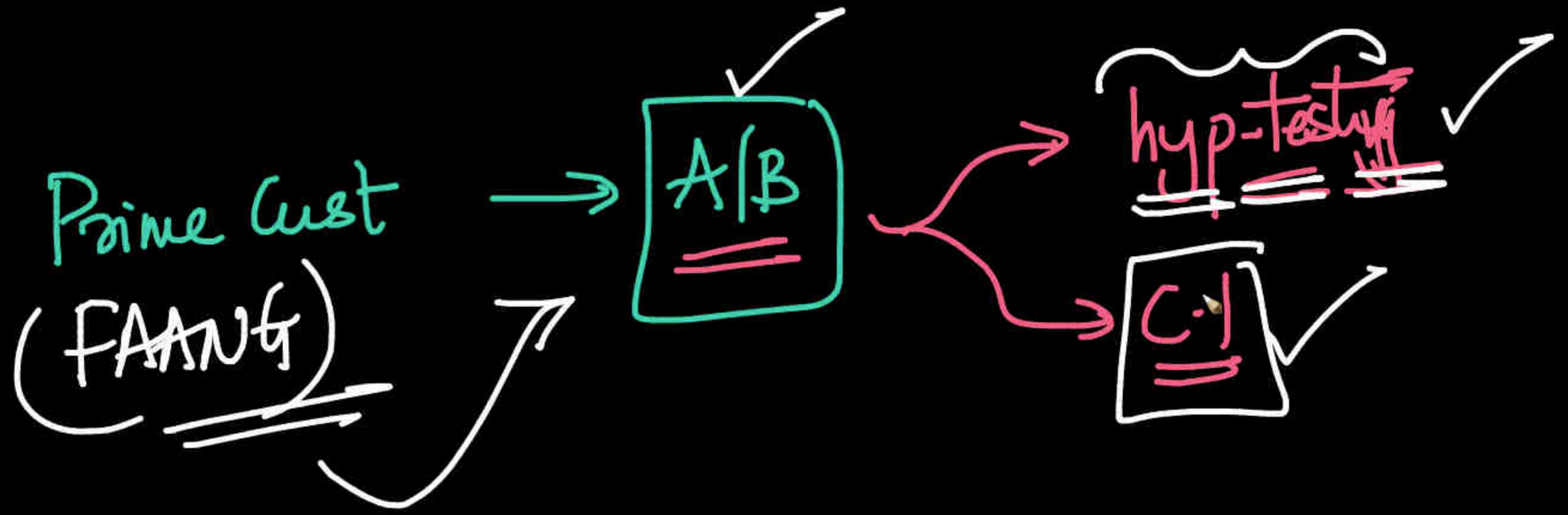
next Class

ML numeric

PCC / SRCC

↳ Yes

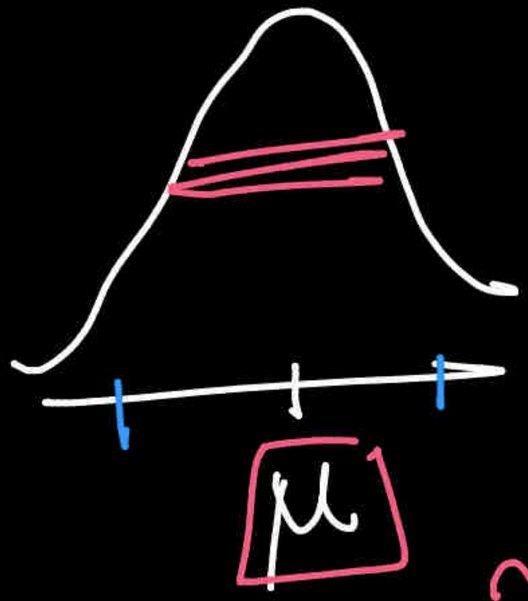

User query ad
f1 f2 ... fd
y
1
0
anamoly detect
4-6%
CTR = #clicks
# ads shown
1 Samplings
2 Weighted loss
ML: imbalanced datasets
heavily
(later)


Prime Cust
(FAANG)
A/B
hyp-testing
C.I

CLT:

$$\left\{\begin{array}{l} S_1 - M_1 \\ S_2 - M_2 \\ \vdots \quad \vdots \\ S_n - M_n \end{array}\right\}$$

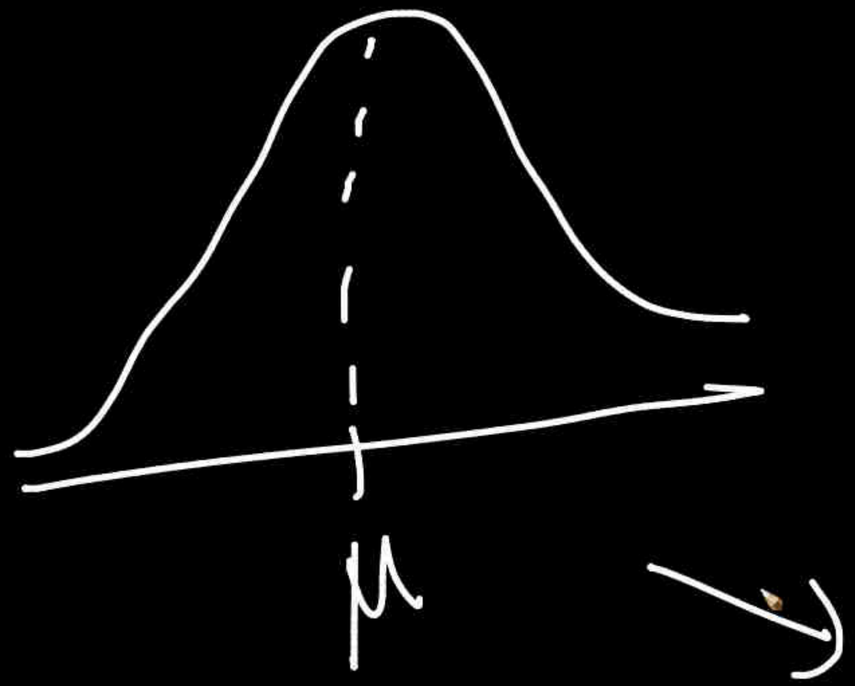
Std-dev. of the disb. of sample means



Std. error

$$\text{Std} = \frac{\sigma}{\sqrt{n}} \quad \text{as } n \to \infty$$

disb. of sample means (M_i)



$$\text{mean} = \mu$$

$$\text{std-dev} = \frac{\sigma}{\sqrt{n}}$$

$M_1 \quad M_2 \sim \cdots \quad M_n$

$M_i \sim N(\text{mean}, \text{std})$

$\dfrac{M_i - \text{mean}(M_i)}{\text{std}(M_i)} = M_i'$

$M_i' \sim Z(0,1)$

Gaussian dist (μ, σ)

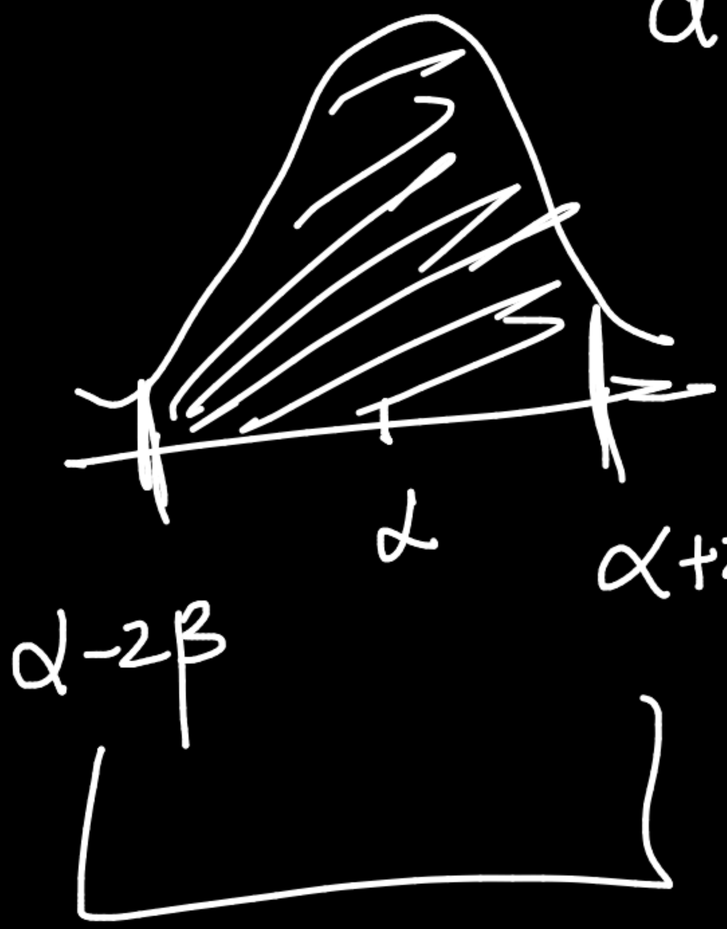
↓

Standardize

↓

$Z(0,1) = N(0,1)$

disb of sample means (M's)

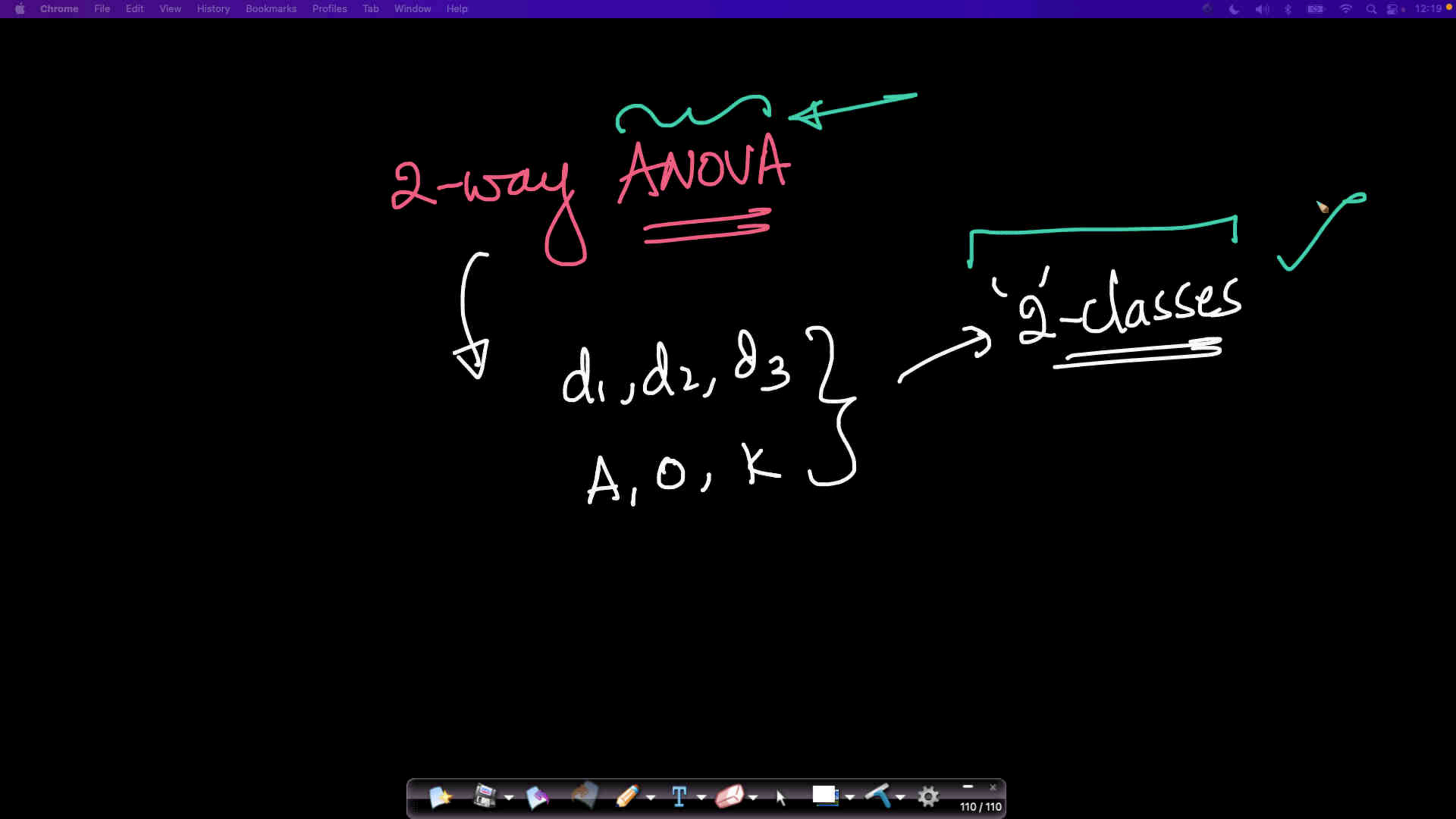


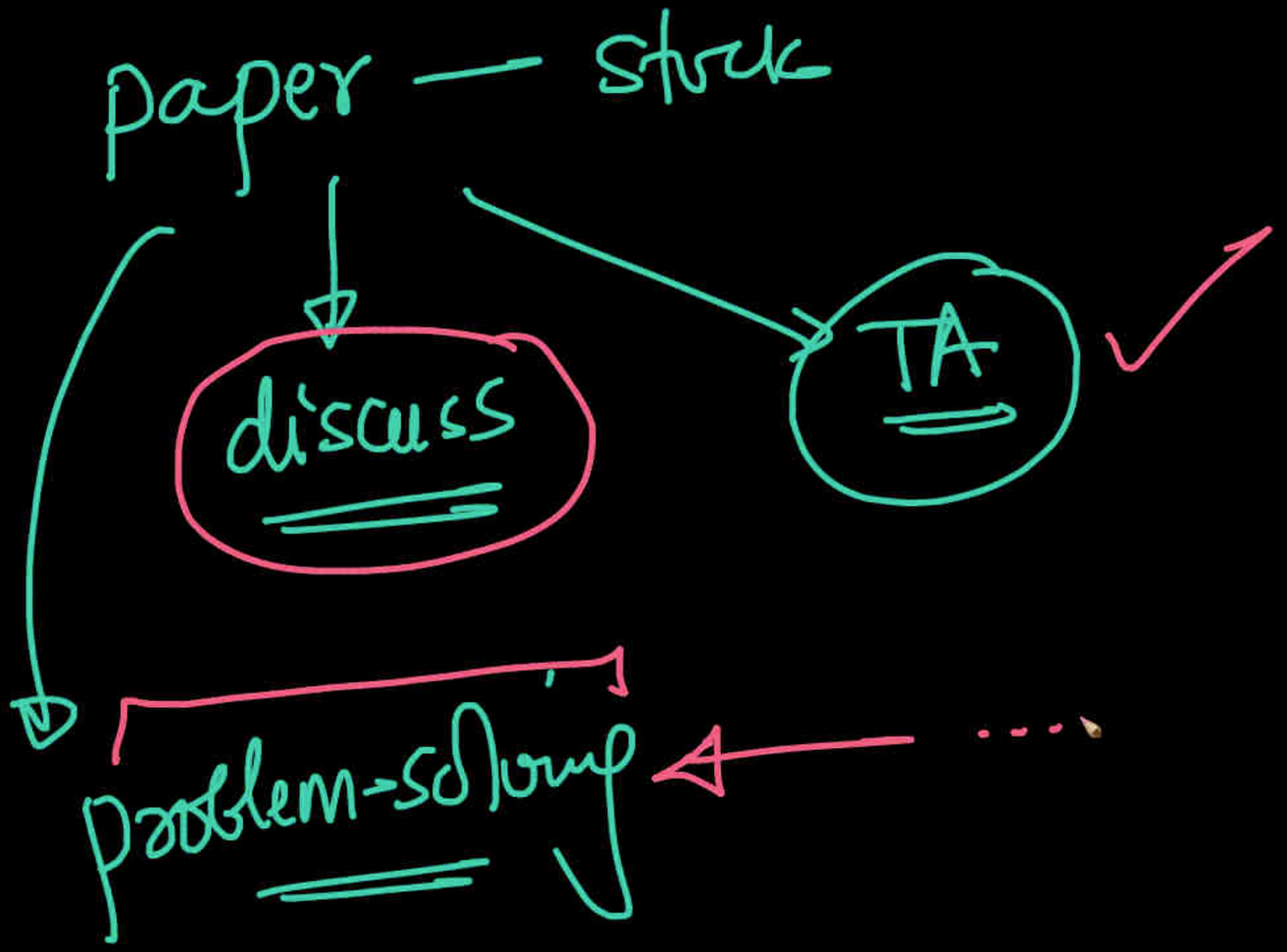
mean (M's) = α

std (M's) = β

$\alpha \approx \mu \rightarrow$ pop

$\beta \approx \frac{\sigma}{\sqrt{n}}$


2-way ANOVA
d1, d2, d3
A, O, K
2-classes


paper — stuck
discuss
TA
problem-solving

numpy.std

numpy.std(*a, axis=None, dtype=None, out=None, ddof=0, keepdims=<no value>, *, where=<no value>*) [source]

Compute the standard deviation along the specified axis.

Returns the standard deviation, a measure of the spread of a distribution, of the array elements. The standard deviation is computed for the flattened array by default, otherwise over the specified axis.

Parameters: **a** : *array_like*

Calculate the standard deviation of these values.

axis : *None or int or tuple of ints, optional*

Axis or axes along which the standard deviation is computed. The default is to compute the standard deviation of the flattened array.

New in version 1.7.0.

If this is a tuple of ints, a standard deviation is performed over multiple axes, instead of a single axis or all the axes as before.

dtype : *dtype, optional*

Type to use in computing the standard deviation. For arrays of integer type the default is float64, for arrays of float types it is the same as the array type.

ve the same shape as the expected output but the type (of the calculated values) will be cast if necessary.