A simple

ML project

Step-by-step Demonstration

Phyu Mon Latt

Pronouns: She/Her

PhD Candidate, MPH, M.B.,B.S

Monash University, School of Translational Medicine
Melbourne Sexual Health Centre, Alfred Health
phyu.latt@monash.edu; platt@mshc.org.au

Agenda

Dataset

Self-created dataset

Google Colab

Why Google Colab for today's demonstration

Pre-processing

What is data pre-processing?

Why important?

ML

ML using Pycaret Package

Dataset

- Self-created dataset (Fake)
- **10** dependent variables and **1** outcome variable
- **Dependent** variables: 👇🏿 👇🏿
- **Outcome** variable: **Infection Diagnosis**

Predictor Variables | **Outcome**

q01_age	q02_gender	q03_evermsm	q04_cob	q05_symptoms	q06_sex_men_12m	q07_male_partners	q08_condom_use_with_men	q09_female_partners	q10_contact_inf	final_dx
34	Female	No	Australia	0	0	2	Not applicable	0	Yes	0
36	Male	No	Australia	0	0	0	Not applicable	2	No	0
50	Male	No	Australia	0	0	0	Not applicable	3	No	0
44	Female	No	Australia	0	0	1	Never	2	No	0
44	Male	Yes	Others	1	1	2	Not applicable	0	Yes	1
27	Male	No	Australia	0	0	0	Not applicable	2	No	0
19	Male	Yes	Australia	1	1	13	Prefer not to answer	0	Yes	1
50	Female	No	Others	1	0	1	Sometimes/Usually	3	No	0
40	Female	No	Australia	0	0	1	Not applicable	0	No	0
27	Male	Yes	Australia	1	1	6	Always	0	Yes	0
18	Male	No	Australia	0	0	0	Not applicable	3	No	0

Agenda

Dataset

Self-created dataset

Google Colab

Why Google Colab for today's demonstration

Pre-processing

What is data pre-processing?

Why important?

ML

ML using Pycaret Package

Google Colab

Several IDEs for Python

Google Colab

Why Google Colab

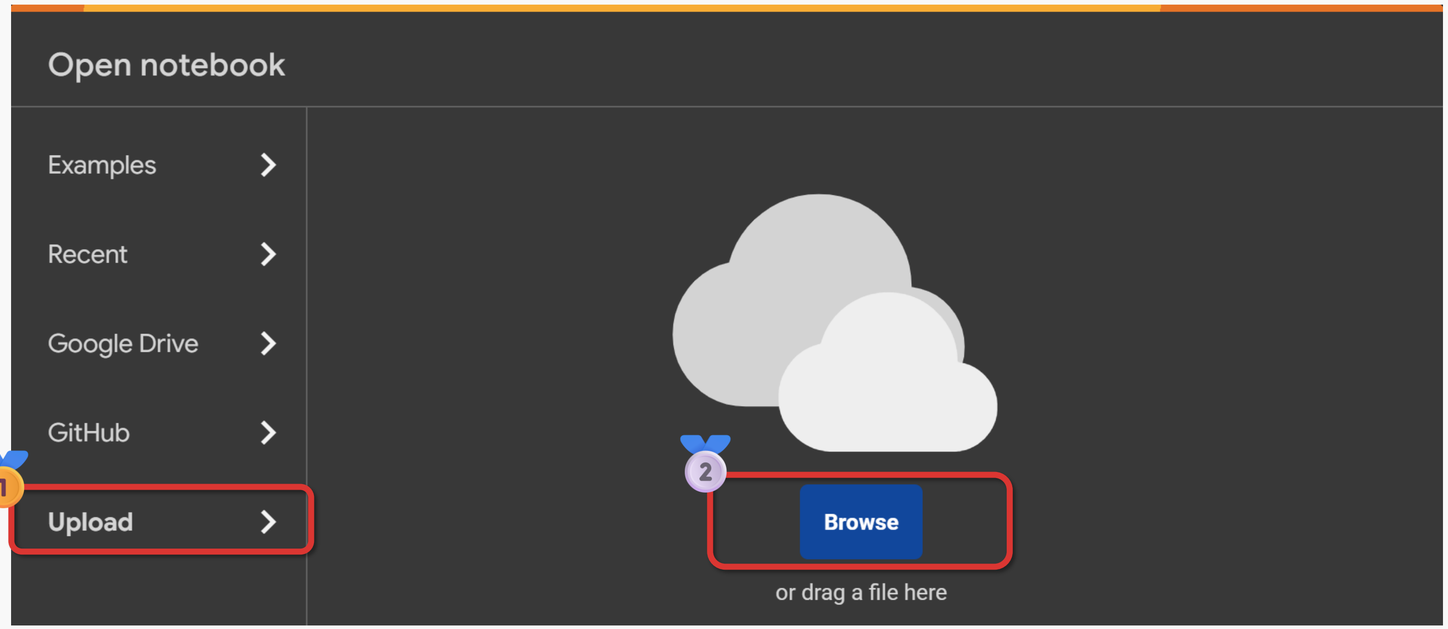
For today

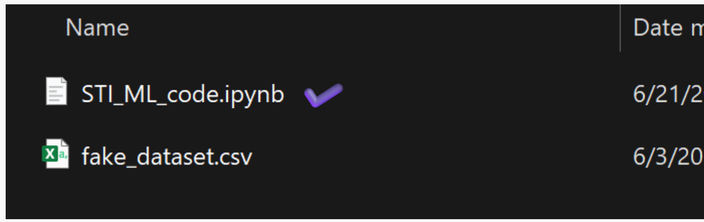
- **Online** and cloud-based
- Free and **easy-to-use**
- **Pre-installed** libraries and Tools
- Integration with **Google Drive**

Google Colab

Google Colab

- **Step 1**: Go to https://colab.research.google.com/
- **Step 2**: Upload **the notebook only** [Filename: STI_ML_code.ipynb]





Google Colab

Google Colab

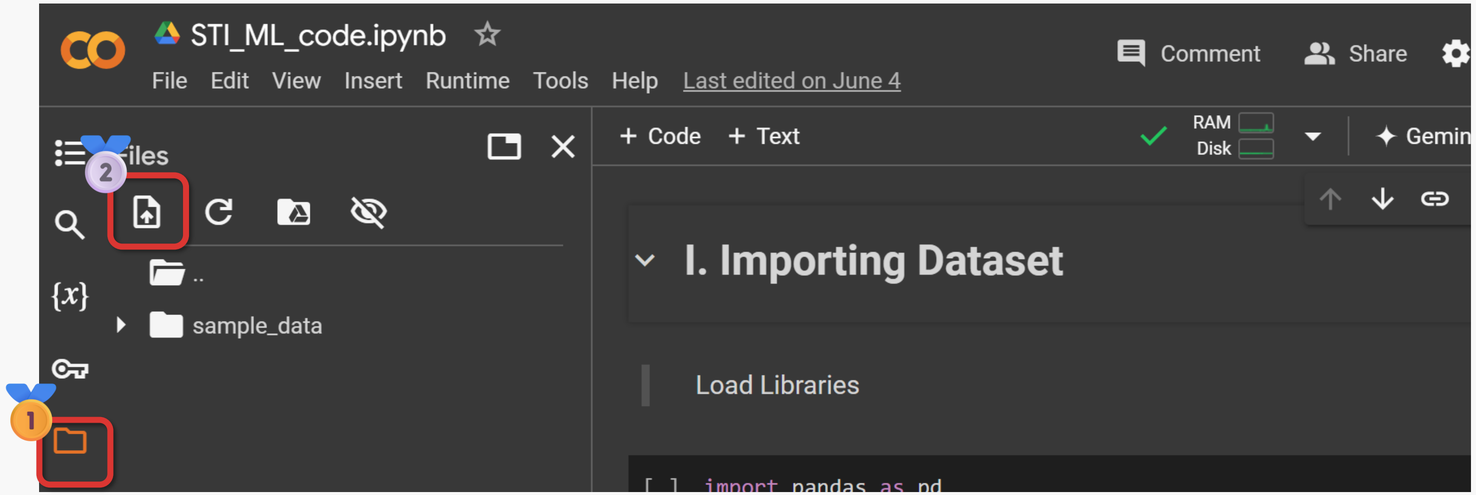
➢ **Step 3**: Connect to the **Google Cloud**.

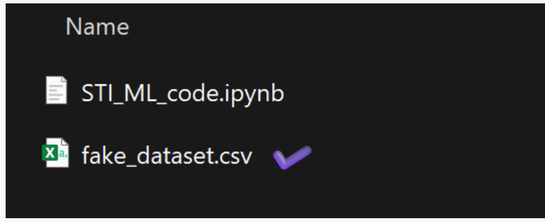
Google Colab

Google Colab

➢ **Step 4**: Upload the **dataset**

Note: Google Colab requires the dataset to be uploaded in a new session.





Agenda

Dataset

Self-created dataset

Google Colab

Why Google Colab for today's demonstration

Pre-processing

What is data pre-processing?

Why important?

ML

ML using Pycaret Package

Pre-processing

I. Import the Libraries

➢ Load the Python libraries.

Installation might be needed. (pip install xxx)

Pre-processing

II. Load the dataset

- Read the dataset using the command:

```
df = pd.read_csv("fake_gc_dataset.csv")
```

Load & Explore dataset 🤯

```
df = pd.read_csv("fake_gc_dataset.csv")
```

- At a glance:

```
df.head()
```

	q01_age	q02_gender	q03_evermsm	q04_cob	q05_symptoms	q06_sex_men_12m	q07_male_partners	q08_condom_use_with_men	q09_female_partners	q10_contact_inf	final_dx
0	34	Female	No	Australia	0	0	2	Not applicable	0	Yes	0
1	36	Male	No	Australia	0	0	0	Not applicable	2	No	0
2	50	Male	No	Australia	0	0	0	Not applicable	3	No	0
3	44	Female	No	Australia	0	0	1	Never	2	No	0
4	44	Male	Yes	Others	1	1	2	Not applicable	0	Yes	1

```
df.columns
```

```
Index(['q01_age', 'q02_gender', 'q03_evermsm', 'q04_cob', 'q05_symptoms',
       'q06_sex_men_12m', 'q07_male_partners', 'q08_condom_use_with_men',
       'q09_female_partners', 'q10_contact_inf', 'final_dx'],
      dtype='object')
```

Pre-processing

III. Data Exploration

- How many **rows/observations**?
- How many **columns/vars**?
- How many **missing values** are there in each var?
- What are the **datatypes** of variables?
- For **categorical** variables, what is the **frequency**?
- For **numerical** variables, what are the **mean, median, SD, IQR**, etc?

Pre-processing

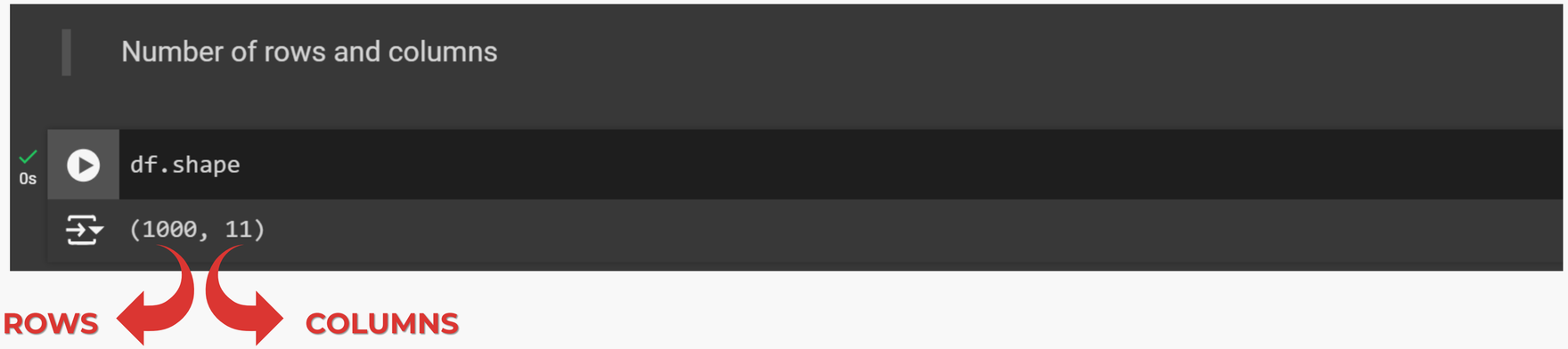
III. Data Exploration

- How many **rows/observations**?
- How many **columns/vars**?
- How many **missing values** are there in each var?
- What are the **datatypes** of variables?
- For **categorical** variables, what is the **frequency**?
- For **numerical** variables, what are the **mean, median, SD, IQR**, etc?

Pre-processing

How many **rows/observations**?
How many **columns/vars**?

```
df.shape
```



Pre-processing

I. Data Exploration

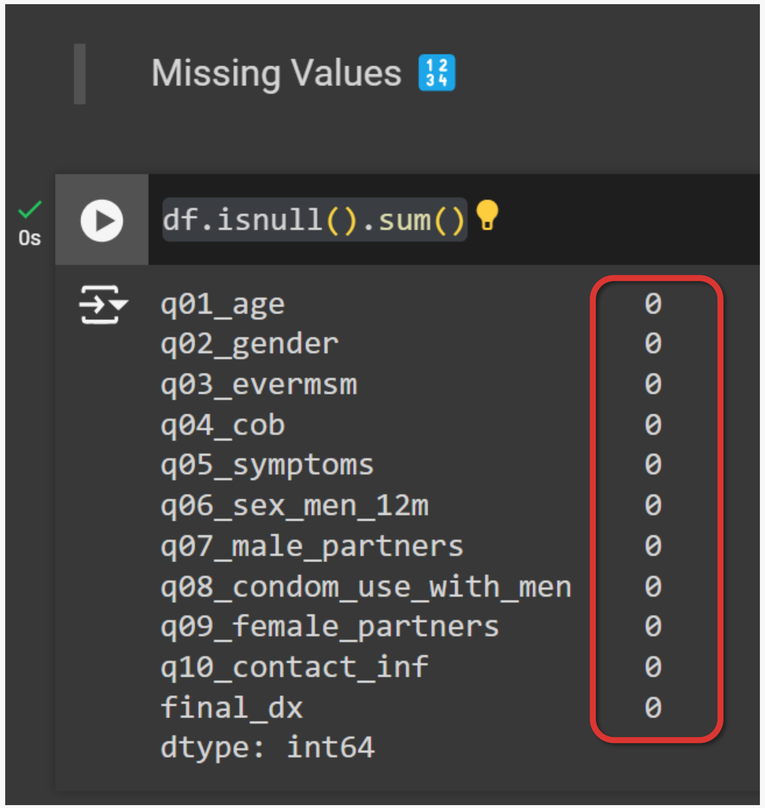
- How many **rows/observations**?
- How many **columns/vars**?
- How many **missing values** are there in each var?
- What are the **datatypes** of variables?
- For **categorical** variables, what is the **frequency**?
- For **numerical** variables, what are the **mean, median, SD, IQR**, etc?

Pre-processing

How many **missing values** are there in each var?

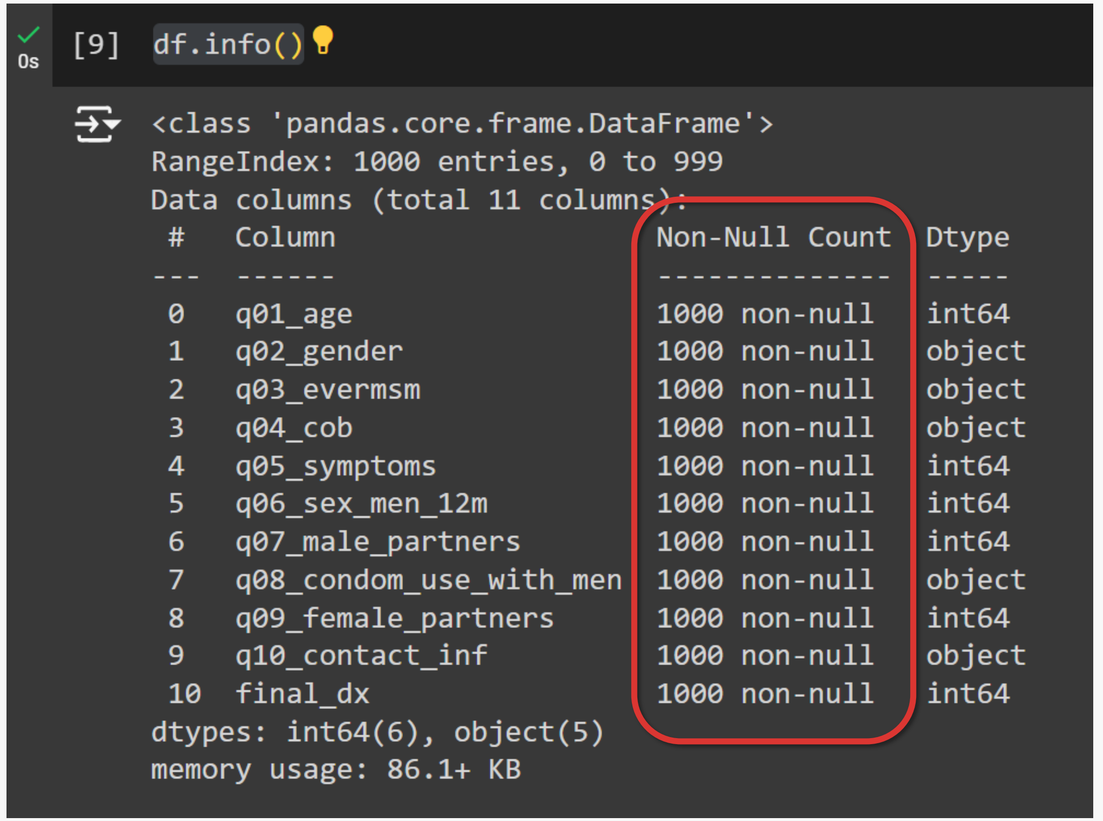
`df.isnull().sum()`

Counts missing values



`df.info()`

Counts non-missing values



Pre-processing

I. Data Exploration

- How many **rows/observations**?
- How many **columns/vars**?
- How many **missing values** are there in each var?
- What are the **datatypes** of variables?
- For **categorical** variables, what is the **frequency**?
- For **numerical** variables, what are the **mean, median, SD, IQR**, etc?

Pre-processing

What are the **datatypes** of variables?

`df.nunique()`

Check the datatype

- Categorical
- Numerical ... etc

```
[10] df.nunique()
```

q01_age	47
q02_gender	2
q03_evermsm	2
q04_cob	2
q05_symptoms	2
q06_sex_men_12m	2
q07_male_partners	15
q08_condom_use_with_men	5
q09_female_partners	5
q10_contact_inf	2
final_dx	2

dtype: int64

`df.dtypes`

```
df.dtypes
```

q01_age	int64
q02_gender	object
q03_evermsm	object
q04_cob	object
q05_symptoms	int64
q06_sex_men_12m	int64
q07_male_partners	int64
q08_condom_use_with_men	object
q09_female_partners	int64
q10_contact_inf	object
final_dx	int64

dtype: object

5 Integer vars

Pre-processing

Categorical or Numerical??

- **q01_age** — E.g. 22, 45, etc
- **q02_gender** — E.g. Male, Female
- **q03_evermsm** — E.g. Yes, No
- **q04_cob** — E.g. Australia, ...
- **q05_symptoms** — E.g. 0, 1, ...
- **q06_sex_men_12m** — E.g. Yes, No
- **q07_male_partners** — E.g. 1, 10
- **q08_condom_use_with_men** — E.g. never, ..
- **q09_female_partners** — E.g. 1, 10
- **q10_contact_inf** — E.g., yes, no.

Pre-processing

CATEGORICAL

- q02_gender
- q03_evermsm
- q04_cob
- q05_symptoms
- q06_sex_men_12m
- q08_condom_use_with_men
- q10_contact_inf

NUMERICAL

- q01_age
- q07_male_partners
- q09_female_partners

Pre-processing

Correct Datatype

```
0s
df['q01_age'] = df['q01_age'].astype('int64')
df['q02_gender'] = df['q02_gender'].astype('category')
df['q03_evermsm'] = df['q03_evermsm'].astype('category')
df['q04_cob'] = df['q04_cob'].astype('category')
df['q05_symptoms'] = df['q05_symptoms'].astype('category')
df['q06_sex_men_12m'] = df['q06_sex_men_12m'].astype('category')
df['q07_male_partners'] = df['q07_male_partners'].astype('int64')
df['q08_condom_use_with_men'] = df['q08_condom_use_with_men'].astype('category')
df['q09_female_partners'] = df['q09_female_partners'].astype('int64')
df['q10_contact_inf'] = df['q10_contact_inf'].astype('category')

print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   q01_age                 1000 non-null   int64
 1   q02_gender              1000 non-null   category
 2   q03_evermsm             1000 non-null   category
 3   q04_cob                 1000 non-null   category
 4   q05_symptoms            1000 non-null   category
 5   q06_sex_men_12m         1000 non-null   category
 6   q07_male_partners       1000 non-null   int64
 7   q08_condom_use_with_men  1000 non-null   category
 8   q09_female_partners     1000 non-null   int64
 9   q10_contact_inf         1000 non-null   category
 10  final_dx                1000 non-null   int64
dtypes: category(7), int64(4)
memory usage: 39.1 KB
None
```

Pre-processing

I. Data Exploration

- How many **rows/observations**?
- How many **columns/vars**?
- How many **missing values** are there in each var?
- What are the **datatypes** of variables?
- For **categorical** variables, what is the **frequency**?
- For **numerical** variables, what are the **mean, median, SD, IQR**, etc?

Pre-processing

For **categorical** variables, what is the **frequency**?

For **numerical** variables, what are the **mean, median, SD, IQR**, etc?

```
df['q01_age'].describe()
```

***Numerical** variables:*

Mean, median, SD, IQR

```
[13] df['q01_age'].describe()

count    1000.000000
mean       40.657000
std        13.594048
min        18.000000
25%        29.000000
50%        40.000000
75%        52.000000
max        64.000000
Name: q01_age, dtype: float64
```

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

***Categorical** variables:*

Frequency

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

final_dx
0    799
1    201
Name: count, dtype: int64
```

Agenda

Dataset

Self-created dataset

Google Colab

Why Google Colab for today's demonstration

Pre-processing

What is data pre-processing?

Why important?

ML

ML using Pycaret Package

ML: Pycaret

What is Pycaret?

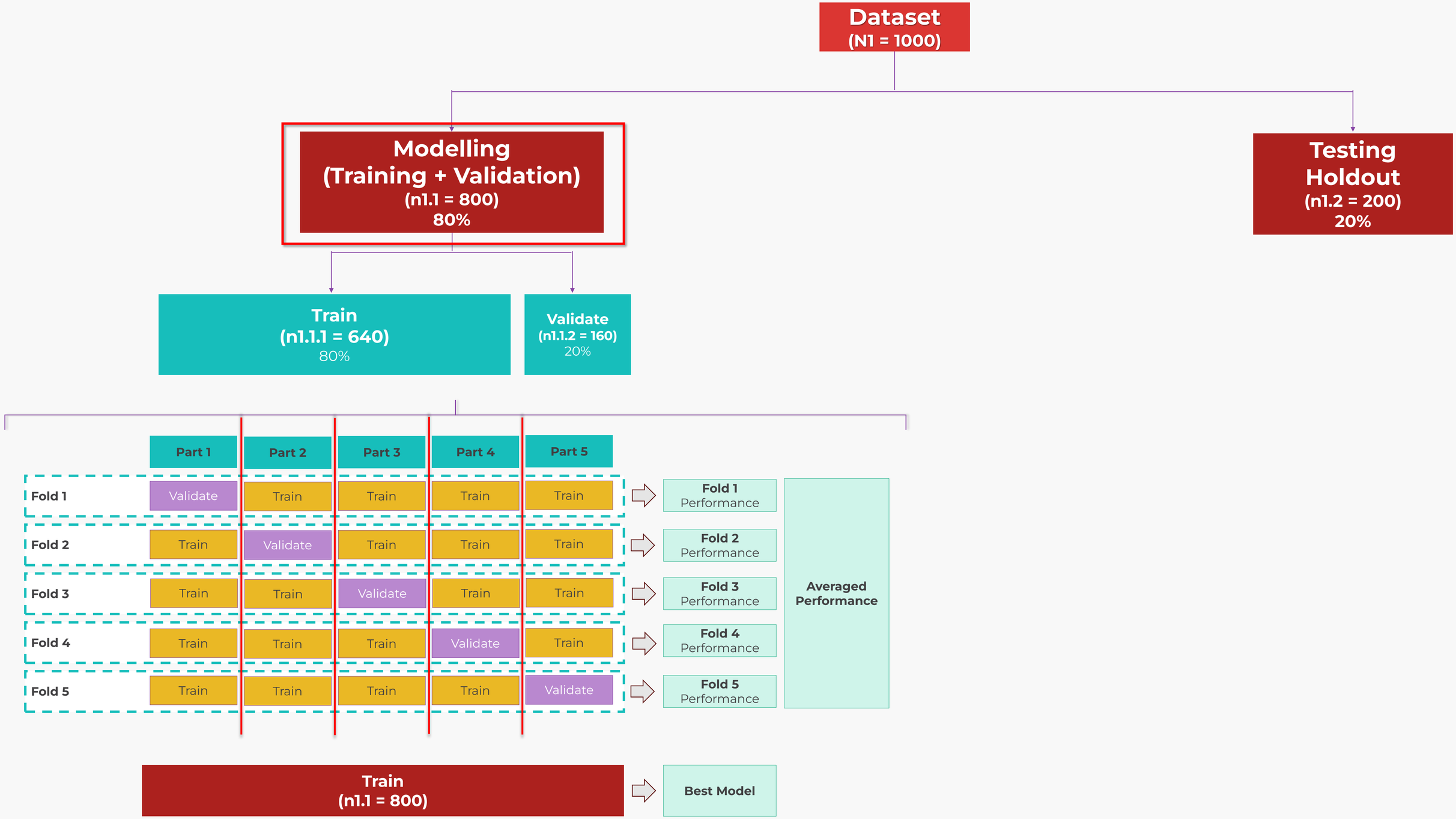
- open-source
- low-code machine learning library in Python
- automates ML workflows
- user-friendly
- https://pycaret.gitbook.io/docs

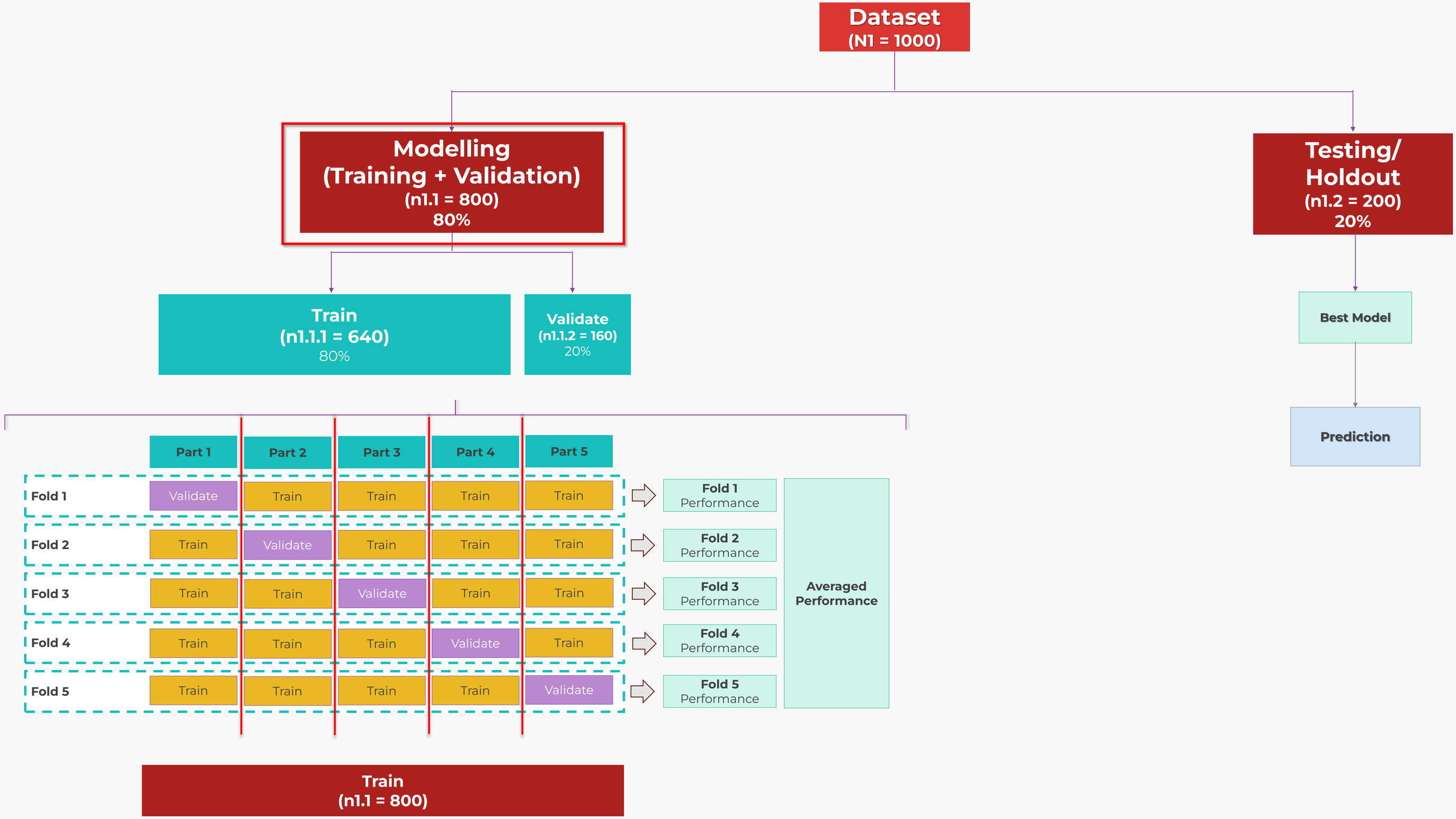
ML: Pycaret

I. Install Pycaret

```
!pip install --pre pycaret[full]
```

```
!pip install --pre pycaret[full]
...
  Downloading scikit_optimize-0.10.2-py2.py3-none-any.whl (107 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 107.8/107.8 kB 15.6 MB/s eta 0:00:00
Collecting mlflow>=2.0.0 (from pycaret[full])
  Downloading mlflow-2.14.0rc0-py3-none-any.whl (25.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 25.8/25.8 MB 44.8 MB/s eta 0:00:00
Collecting gradio>=3.50.2 (from pycaret[full])
  Downloading gradio-4.36.1-py3-none-any.whl (12.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.3/12.3 MB 23.4 MB/s eta 0:00:00
Collecting boto3>=1.24.56 (from pycaret[full])
  Downloading boto3-1.34.123-py3-none-any.whl (139 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 139.3/139.3 kB 19.2 MB/s eta 0:00:00
Collecting fastapi (from pycaret[full])
  Downloading fastapi-0.111.0-py3-none-any.whl (91 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 92.0/92.0 kB 13.8 MB/s eta 0:00:00
Collecting uvicorn>=0.17.6 (from pycaret[full])
  Downloading uvicorn-0.30.1-py3-none-any.whl (62 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.4/62.4 kB 8.9 MB/s eta 0:00:00
Collecting m2cgen>=0.9.0 (from pycaret[full])
  Downloading m2cgen-0.10.0-py3-none-any.whl (92 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 92.2/92.2 kB 13.8 MB/s eta 0:00:00
Collecting evidently~=0.4.16 (from pycaret[full])
  Downloading evidently-0.4.26-py3-none-any.whl (3.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.4/3.4 MB 66.9 MB/s eta 0:00:00
```

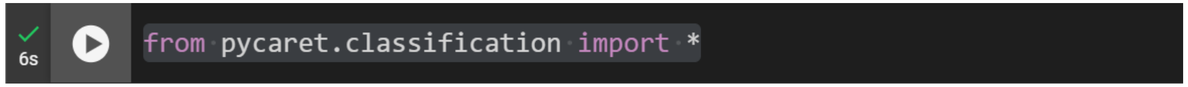

Dataset
(N1 = 1000)
Modelling
(Training + Validation)
(n1.1 = 800)
80%
Testing
Holdout
(n1.2 = 200)
20%
Train
(n1.1.1 = 640)
80%
Validate
(n1.1.2 = 160)
20%
Part 1
Part 2
Part 3
Part 4
Part 5
Fold 1
Validate
Train
Train
Train
Train
Fold 2
Train
Validate
Train
Train
Train
Fold 3
Train
Train
Validate
Train
Train
Fold 4
Train
Train
Train
Validate
Train
Fold 5
Train
Train
Train
Train
Validate
Fold 1 Performance
Fold 2 Performance
Fold 3 Performance
Fold 4 Performance
Fold 5 Performance
Averaged Performance
Train
(n1.1 = 800)
Best Model


Dataset
(N1 = 1000)
Modelling
(Training + Validation)
(n1.1 = 800)
80%
Testing/
Holdout
(n1.2 = 200)
20%
Train
(n1.1.1 = 640)
80%
Validate
(n1.1.2 = 160)
20%
Best Model
Prediction
Part 1
Part 2
Part 3
Part 4
Part 5
Fold 1
Validate
Train
Train
Train
Train
Fold 2
Train
Validate
Train
Train
Train
Fold 3
Train
Train
Validate
Train
Train
Fold 4
Train
Train
Train
Validate
Train
Fold 5
Train
Train
Train
Train
Validate
Fold 1 Performance
Fold 2 Performance
Fold 3 Performance
Fold 4 Performance
Fold 5 Performance
Averaged Performance
Train
(n1.1 = 800)

ML: Pycaret

II. Import classification module

```
from pycaret.classification import *
```



ML: Pycaret

III. Set up the environment

```
exp = setup(data = df, target = 'final_dx')
```

Compulsory:

1. dataset (**data**)
2. target variable (**target**)

	Description	Value
0	Session id	5217
1	Target	final_dx
2	Target type	Binary
3	Original data shape	(1000, 11)
4	Transformed data shape	(1000, 15)
5	Transformed train set shape	(700, 15)
6	Transformed test set shape	(300, 15)
7	Numeric features	3
8	Categorical features	7
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Maximum one-hot encoding	25
14	Encoding method	None
15	Fold Generator	StratifiedKFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	clf-default-name
21	USI	6299

https://pycaret.gitbook.io/docs/get-started/preprocessing

PyCaret 3.0

GET STARTED

Installation

Quickstart

Tutorials

Modules

Data Preprocessing

Data Preparation

Scale and Transform

Feature Engineering

Feature Selection

Other setup parameters

Functions

LEARN PYCARET

Blog

Videos

Cheat sheet

FAQs

Examples

IMPORTANT LINKS

Release Notes

Data Preprocessing

Data preprocessing and Transformations available in PyCaret

Data Preparation | Scale and Transf... | Feature Engineer... | Feature Select... | Other setup paramet...

Missing Values

Datasets for various reasons may have missing values or empty records, often encoded as blanks or `NaN`. Most of the machine learning algorithms are not capable of dealing with the missing values.

Data Types

Each feature in the dataset has an associated data type such as numeric, categorical, or Datetime. PyCaret automatically detects the data type of each feature.

One-Hot Encoding

Categorical features in the dataset contain the label values (ordinal or nominal) rather than continuous numbers. Most of the machine learning algorithms are not capable of handling categorical data without encoding.

Ordinal Encoding

When the categorical features in the dataset contain variables with intrinsic natural order such as *Low, Medium, and High*, these must be encoded differently than nominal variables (where there is no intrinsic order for e.g. Male or Female).

ML: Pycaret

III. Set up the environment

```
exp = setup(data = df, target = 'final_dx')
```

Compulsory:

1. dataset (**data**)
2. target variable (**target**)

Default values:

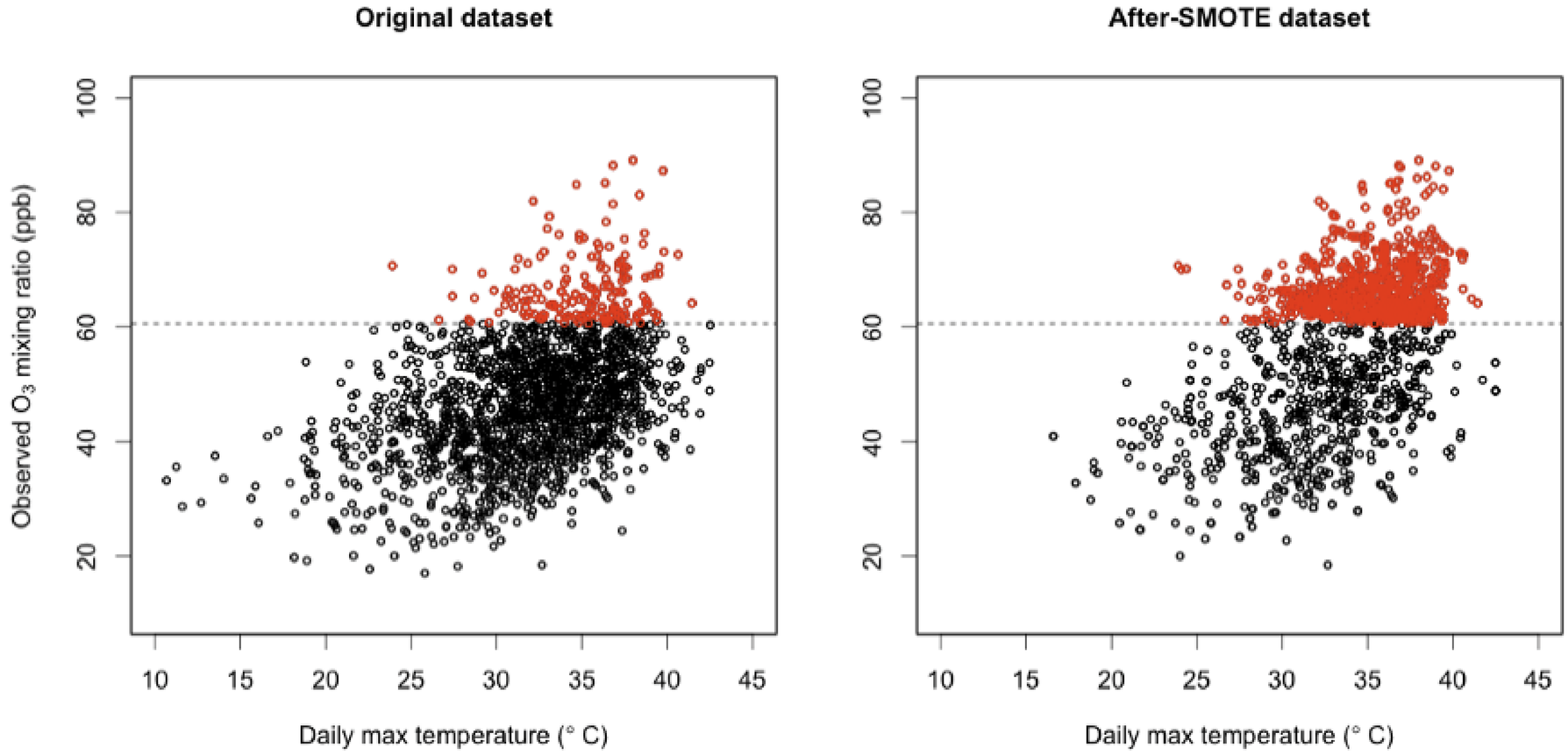
No	Description	Default	Meaning
1	session_id		Radom seed for reproducibility
2	normalize	False	To normalise numeric features
3	train_size	0.7	Training: Testing Split
4	categorical_features	None	To tell Pycaret: a list of categorical column names
5	numeric_features	None	a list of numerical column names
6	ignore_features	None	A list of columns to be excluded from modeling
7	remove_outliers	False	To remove outliers from the training data
8	fix_imbalance	False	The training dataset is resampled using “SMOTE” method.

https://pycaret.gitbook.io/docs/get-started/preprocessing

	Description	Value
0	Session id	5217
1	Target	final_dx
2	Target type	Binary
3	Original data shape	(1000, 11)
4	Transformed data shape	(1000, 15)
5	Transformed train set shape	(700, 15)
6	Transformed test set shape	(300, 15)
7	Numeric features	3
8	Categorical features	7
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Maximum one-hot encoding	25
14	Encoding method	None
15	Fold Generator	StratifiedKFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	clf-default-name
21	USI	6299

ML: Pycaret

SMOTE



https://pycaret.gitbook.io/docs/get-started/preprocessing/data-preparation

ML: Pycaret

III. Set up the environment

With more parameters

```
[ ]  exp = setup(data = df, target = 'final_dx',
                 session_id = 12345,
                 fix_imbalance = True, #the training dataset is resampled using the algorithm defined in fix_imbalance_method . When None, SMOTE is
                 remove_outliers = True, #remove outliers from the dataset before training the model.
                 normalize = True, #transformed using the method defined under the normalized_method parameter.
                 fold = 5,
                 train_size = 0.80
           )
```

	Description	Value
0	Session id	12345
1	Target	final_dx
2	Target type	Binary
3	Original data shape	(1000, 11)
4	Transformed data shape	(1432, 15)
5	Transformed train set shape	(1232, 15)
6	Transformed test set shape	(200, 15)
7	Numeric features	3
8	Categorical features	7
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Maximum one-hot encoding	25
14	Encoding method	None
15	Remove outliers	True
16	Outliers threshold	0.050000
17	Fix imbalance	True
18	Fix imbalance method	SMOTE
19	Normalize	True
20	Normalize method	zscore
21	Fold Generator	StratifiedKFold
22	Fold Number	5
23	CPU Jobs	-1

https://pycaret.gitbook.io/docs/get-started/functions/initialize

ML: Pycaret

IV. Model Training

Two ways

- ✓ Compare different models
- ✓ Train a particular model

https://pycaret.gitbook.io/docs/get-started/functions/train

ML: Pycaret

IV. Model Training

✓ **Compare different models**

```
compare_models(include = ['gbc', 'rf', 'catboost', 'knn'])
```

Sort

'auc'

'F1'

```
compare_models(include = ['gbc', 'rf', 'catboost', 'knn'])
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
gbc	Gradient Boosting Classifier	0.9575	0.9462	0.8509	0.9330	0.9986	0.8634	0.8651
catboost	CatBoost Classifier	0.9550	0.9336	0.8509	0.9205	0.8838	0.8560	0.8574
rf	Random Forest Classifier	0.9462	0.9115	0.8449	0.8859	0.8641	0.8306	0.8316
knn	K Neighbors Classifier	0.9350	0.9043	0.8447	0.8374	0.8402	0.7994	0.8001

GradientBoostingClassifier

```
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='log_loss', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_samples_leaf=1,
                           min_samples_split=2, min_weight_fraction_leaf=0.0,
                           n_estimators=100, n_iter_no_change=None,
                           random_state=12345, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

https://pycaret.gitbook.io/docs/get-started/functions/train

ML: Pycaret

IV. Model Training

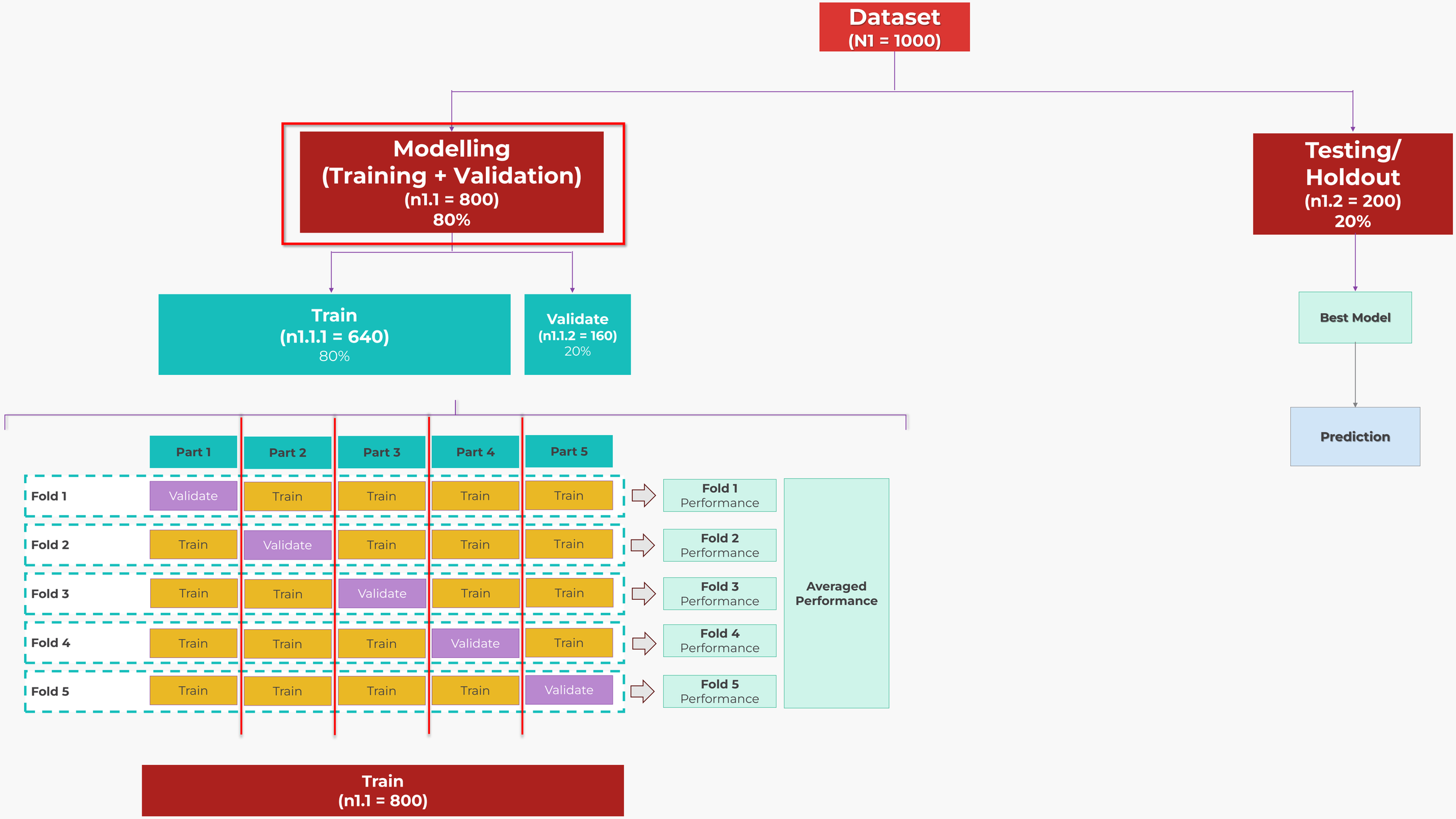
✓ **Train a particular model (Example: catboost)**

```
catboost = create_model('catboost')
```

```
catboost = create_model('catboost')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9688	0.9786	0.9062	0.9355	0.9206	0.9012	0.9014
1	0.9562	0.9231	0.8125	0.9630	0.8814	0.8548	0.8594
2	0.9438	0.9182	0.8438	0.8710	0.8571	0.8221	0.8223
3	0.9500	0.9216	0.8438	0.9000	0.8710	0.8400	0.8407
4	0.9562	0.9266	0.8485	0.9333	0.8889	0.8617	0.8632
Mean	0.9550	0.9336	0.8509	0.9205	0.8838	0.8560	0.8574
Std	0.0083	0.0227	0.0305	0.0318	0.0213	0.0264	0.0264

https://pycaret.gitbook.io/docs/get-started/functions/train


Dataset
(N1 = 1000)
Modelling
(Training + Validation)
(n1.1 = 800)
80%
Testing/
Holdout
(n1.2 = 200)
20%
Train
(n1.1.1 = 640)
80%
Validate
(n1.1.2 = 160)
20%
Best Model
Prediction
Part 1
Part 2
Part 3
Part 4
Part 5
Fold 1
Validate
Train
Train
Train
Train
Fold 2
Train
Validate
Train
Train
Train
Fold 3
Train
Train
Validate
Train
Train
Fold 4
Train
Train
Train
Validate
Train
Fold 5
Train
Train
Train
Train
Validate
Fold 1 Performance
Fold 2 Performance
Fold 3 Performance
Fold 4 Performance
Fold 5 Performance
Averaged Performance
Train
(n1.1 = 800)

ML: Pycaret

IV. Model Training

✓ **Train a particular model (Example: catboost)**

```
catboost = create_model('catboost')
```

```
catboost = create_model('catboost')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9688	0.9786	0.9062	0.9355	0.9206	0.9012	0.9014
1	0.9562	0.9231	0.8125	0.9630	0.8814	0.8548	0.8594
2	0.9438	0.9182	0.8438	0.8710	0.8571	0.8221	0.8223
3	0.9500	0.9216	0.8438	0.9000	0.8710	0.8400	0.8407
4	0.9562	0.9266	0.8485	0.9333	0.8889	0.8617	0.8632
Mean	0.9550	0.9336	0.8509	0.9205	0.8838	0.8560	0.8574
Std	0.0083	0.0227	0.0305	0.0318	0.0213	0.0264	0.0264

https://pycaret.gitbook.io/docs/get-started/functions/train

ML: Pycaret

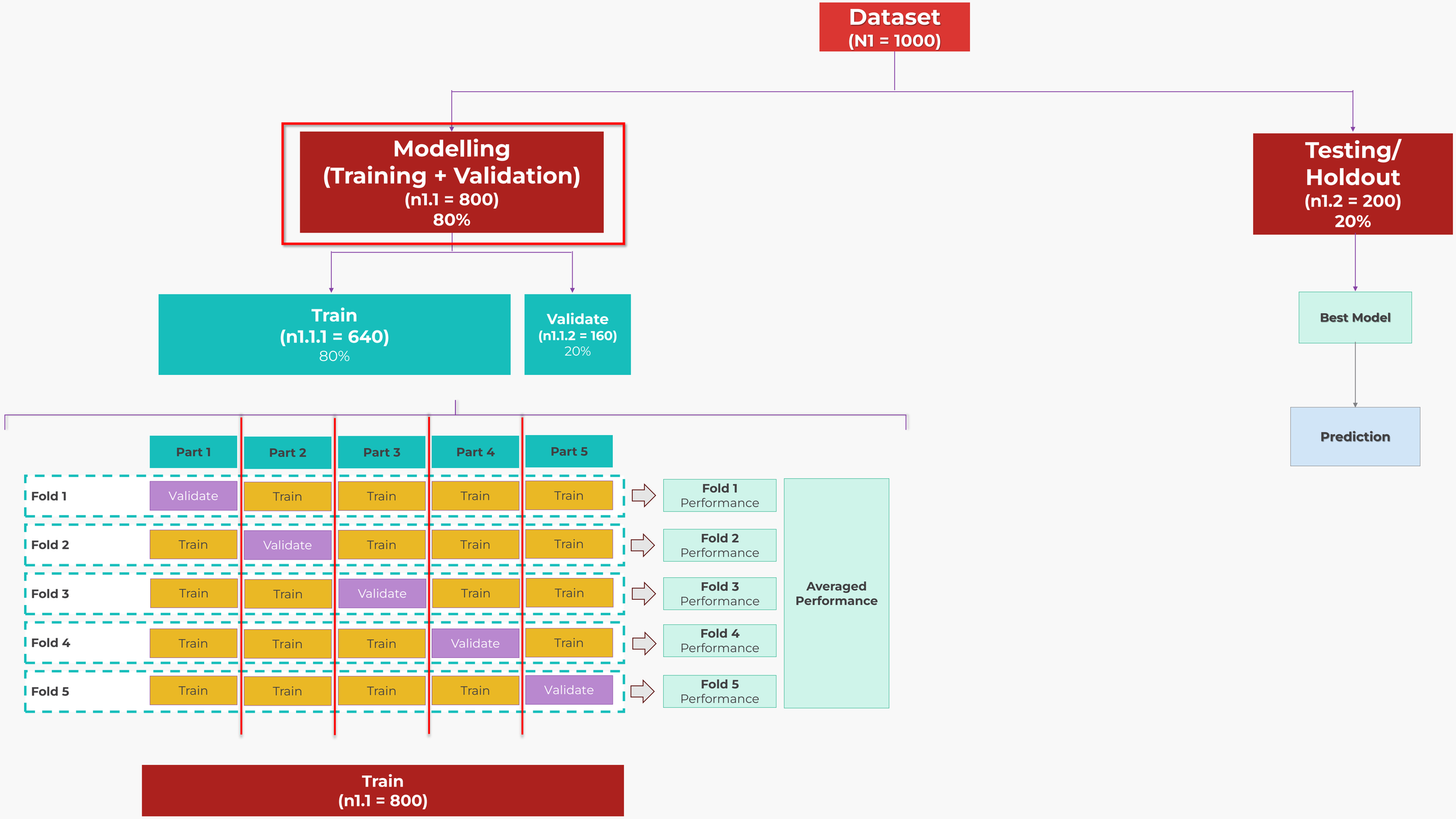
V. Predict on the testing/holdout dataset

```
predictions = predict_model(catboost)
```

```
[33] predictions = predict_model(catboost)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	CatBoost Classifier	0.9800	0.9593	0.9000	1.0000	0.9474	0.9351	0.9370

https://pycaret.gitbook.io/docs/get-started/functions/deploy


Dataset
(N1 = 1000)
Modelling
(Training + Validation)
(n1.1 = 800)
80%
Testing/
Holdout
(n1.2 = 200)
20%
Train
(n1.1.1 = 640)
80%
Validate
(n1.1.2 = 160)
20%
Best Model
Prediction
Part 1
Part 2
Part 3
Part 4
Part 5
Fold 1
Validate
Train
Train
Train
Train
Fold 2
Train
Validate
Train
Train
Train
Fold 3
Train
Train
Validate
Train
Train
Fold 4
Train
Train
Train
Validate
Train
Fold 5
Train
Train
Train
Train
Validate
Fold 1 Performance
Fold 2 Performance
Fold 3 Performance
Fold 4 Performance
Fold 5 Performance
Averaged Performance
Train
(n1.1 = 800)

ML: Pycaret

VI. Model Evaluation

✓ **Train a particular model**

```
plot_model(catboost, 'xxx')
```

Sort

‘**auc**’: Area under the Curve

‘**feature**’: Feature of Importance

'**confusion_matrix**’: Confusion Matrix

https://pycaret.gitbook.io/docs/get-started/functions/analyze

PyCaret 3.0

GET STARTED

Installation

Quickstart

Tutorials

Modules

Data Preprocessing

Functions

Initialize

Train

Optimize

Analyze

Deploy

Others

LEARN PYCARET

Blog

Videos

Cheat sheet

FAQs

Examples

IMPORTANT LINKS

Classification

Plot Name	Plot
Area Under the Curve	‘auc’
Discrimination Threshold	‘threshold’
Precision Recall Curve	‘pr’
Confusion Matrix	‘confusion_matrix’
Class Prediction Error	‘error’
Classification Report	‘class_report’
Decision Boundary	‘boundary’
Recursive Feature Selection	‘rfe’
Learning Curve	‘learning’
Manifold Learning	‘manifold’
Calibration Curve	‘calibration’
Validation Curve	‘vc’
Dimension Learning	‘dimension’
Feature Importance (Top 10)	‘feature’

plot_model

Example

Change the scale

Save the plot

Customize the plot

Use train data

Examples by module

evaluate_model

interpret_model

Example

Save the plot

Change plot type

Use train data

dashboard

check_fairness

get_leaderboard

assign_model

ML: Pycaret

VI. Model Evaluation

✓ **Train a particular model**

```
plot_model(catboost, 'xxx')
```

Types

‘**auc**’: Area under the Curve

‘**feature**’: Feature of Importance

'**confusion_matrix**’: Confusion Matrix

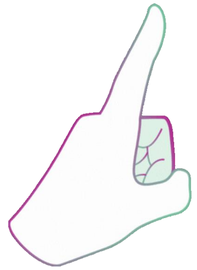
https://pycaret.gitbook.io/docs/get-started/functions/analyze

ML: Pycaret

VI. Model Evaluation

✓ **Train a particular model**

```
plot_model(catboost, 'auc')
```

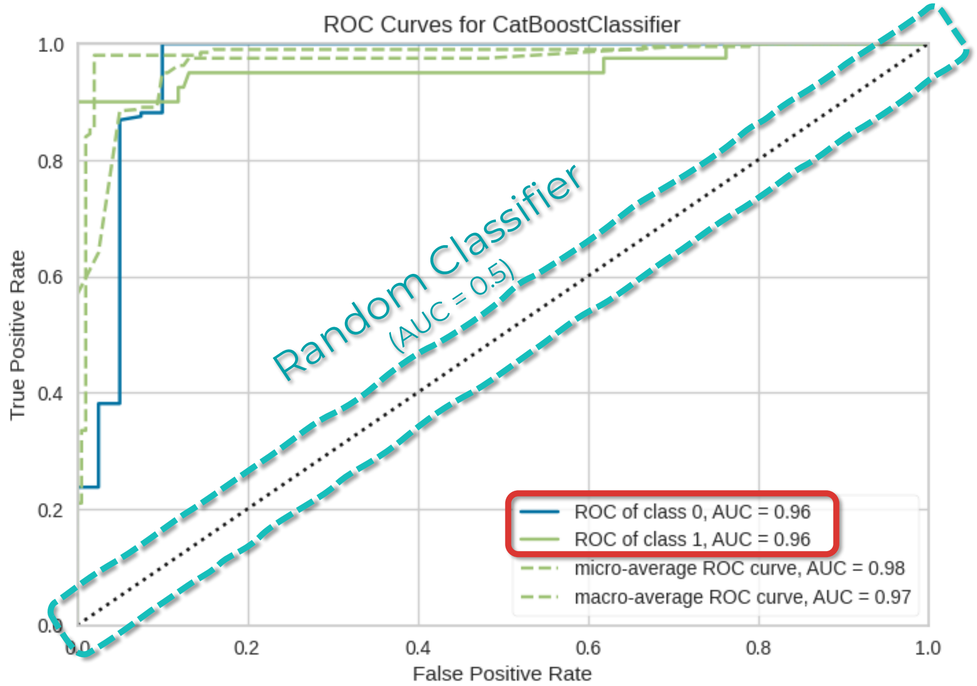


Types

'**auc**': Area under the Curve

'**confusion_matrix**': Confusion Matrix

'**feature**': Feature of Importance



https://pycaret.gitbook.io/docs/get-started/functions/analyze

ML: Pycaret

VI. Model Evaluation

✓ **Train a particular model**

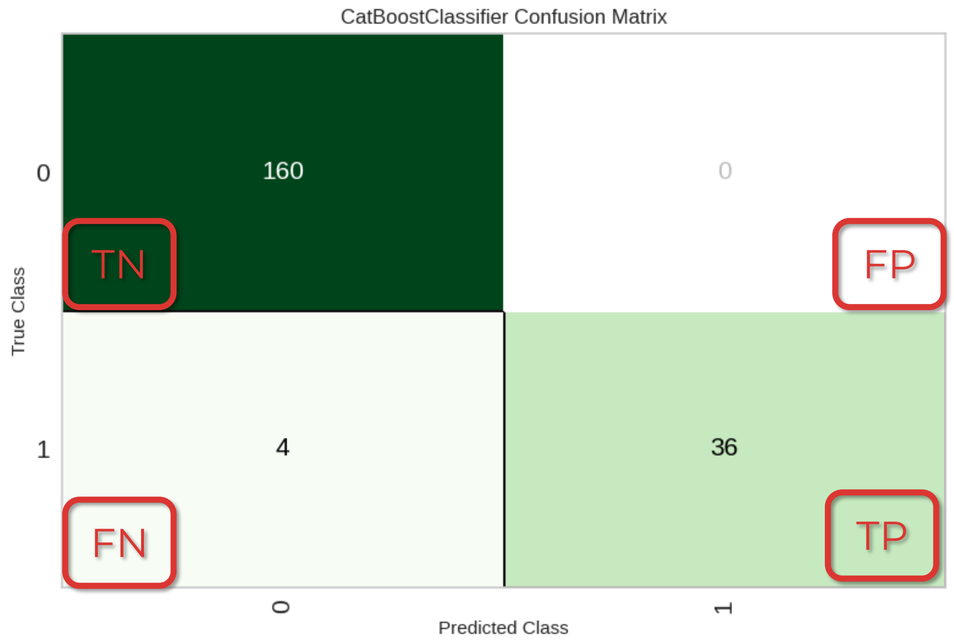
```
plot_model(catboost, 'confusion_matrix')
```

Types

'**auc**': Area under the Curve

'confusion_matrix': Confusion Matrix

'**feature**': Feature of Importance



https://pycaret.gitbook.io/docs/get-started/functions/analyze

ML: Pycaret

VI. Model Evaluation

✓ **Train a particular model**

```
plot_model(catboost, 'feature')
```

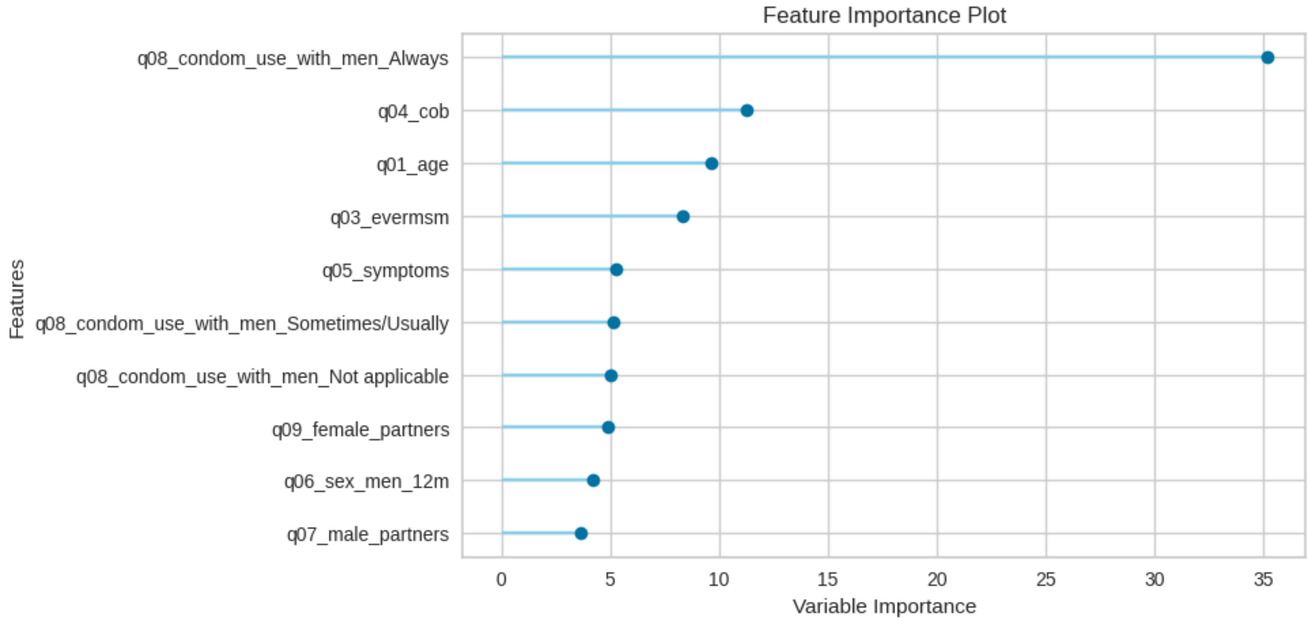
Types

'**auc**': Area under the Curve

'**confusion_matrix**': Confusion Matrix

'**feature**': Feature of Importance

... ... many more

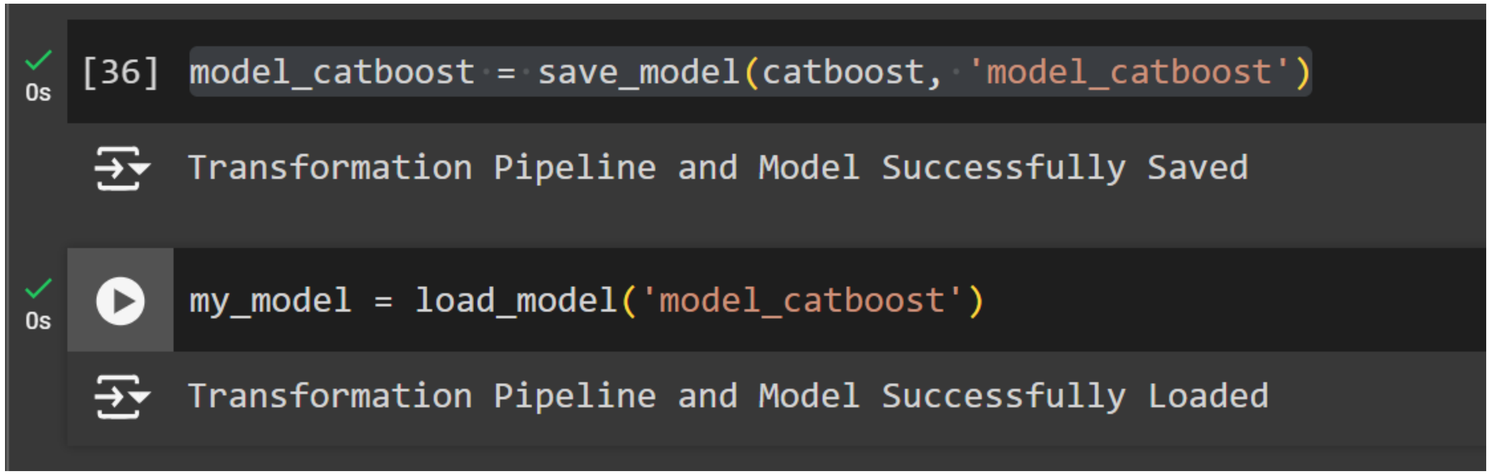


https://pycaret.gitbook.io/docs/get-started/functions/analyze

ML: Pycaret

VII. Saving & re-loading the model

```
model_catboost = save_model(catboost, 'model_catboost')
```



https://pycaret.gitbook.io/docs/get-started/functions/deploy

ML: Pycaret

VIII. Prediction on the unseen/external validation dataset

```
unseen_predict = predict_model(my_model, data = df_unseen)
```

https://pycaret.gitbook.io/docs/get-started/functions/deploy


Thank you