

MACHINE LEARNING PROJECT

Paper 2

Supervised by:

Mrs. Trabelsi Wiem

esprit 
Se former autrement



MEMBERS OF OUR GROUP



Chtourou Mohamed Jasser



Ben Rhaiem Ghofrane



Traidi Yassine



Ben Romdhane Aziz



Jaouadi Yasmine

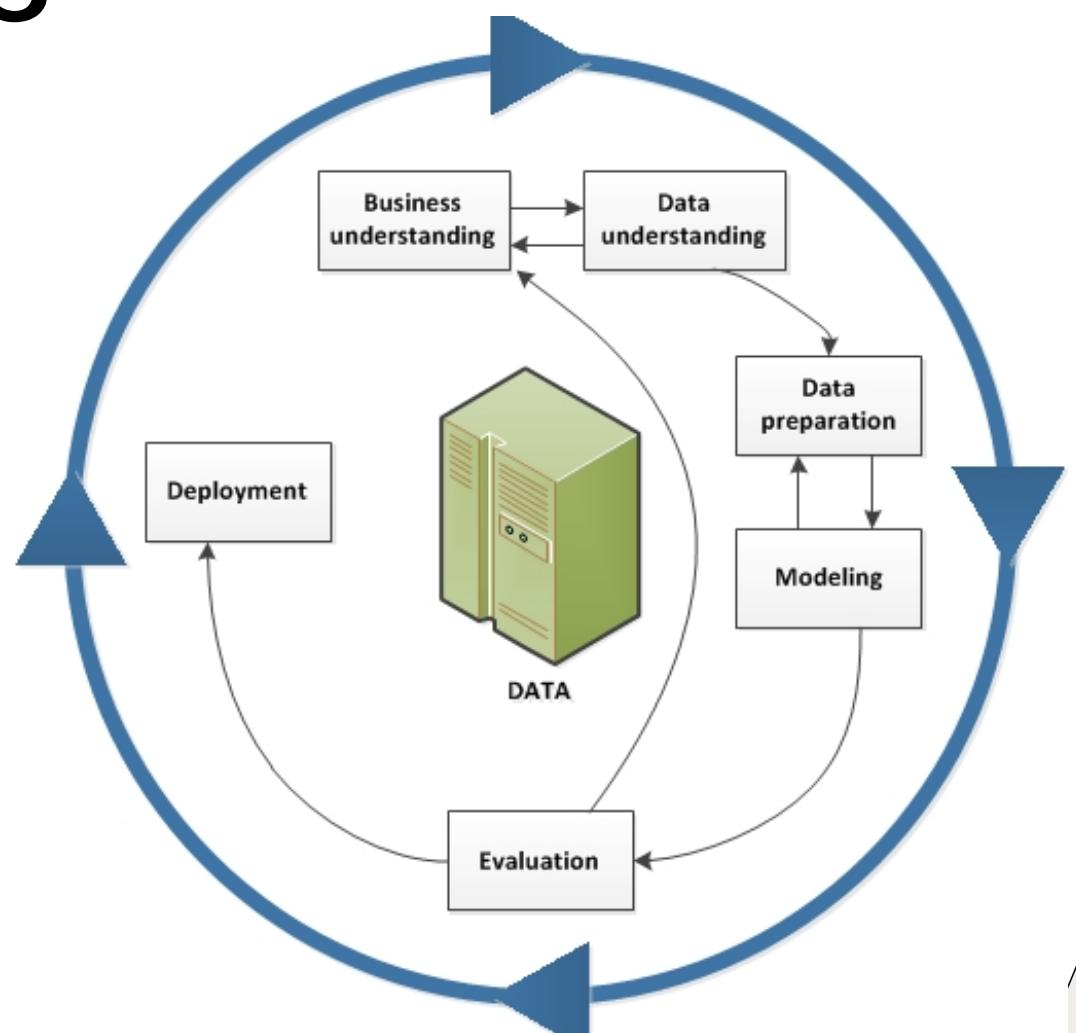


Gotrane Saifeddine

PRESENTATION PLAN

CRISP-DM STEPS

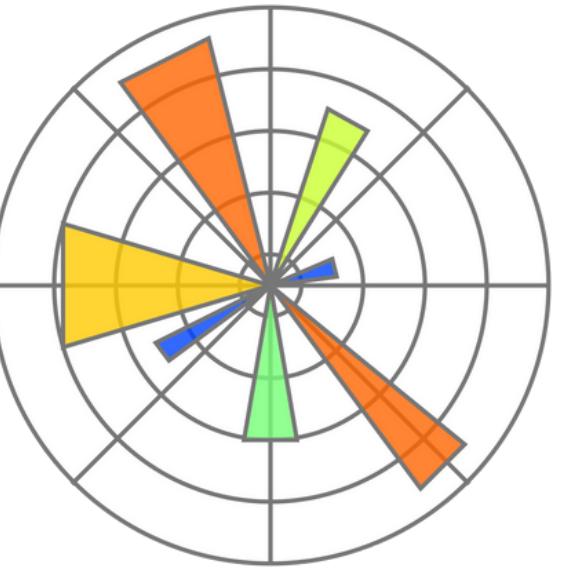
- 1- Business Understanding
- 2- Data Understanding
- 3- Data Preparation
- 4- Modeling
- 5- Evaluation
- 6- Deployment



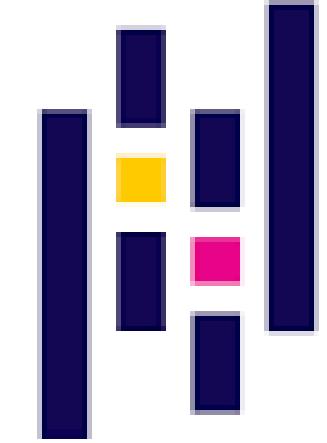
LIBRARIES



Seaborn



Matplotlib



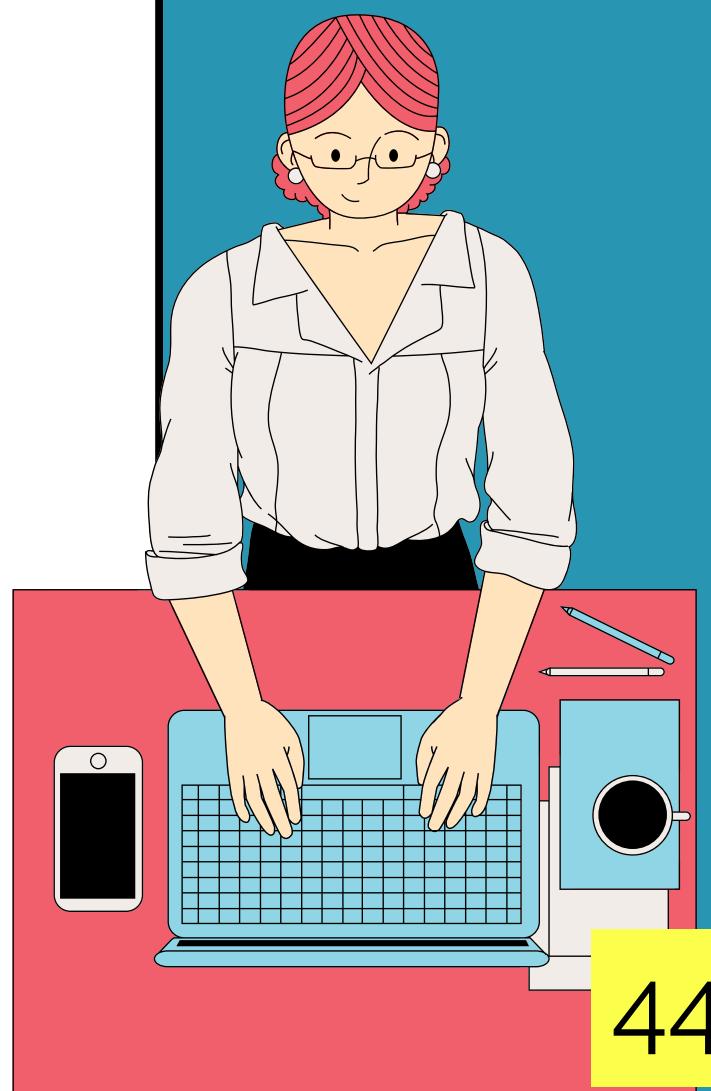
Pandas



Scikit Learn



NumPy



BUSINESS UNDERSTANDING

- **Problem:**

Need to detect network intrusions and cyberattacks in real-time

- **Goals:**

- Data Understanding through Visualization and Exploratory Analysis
- Dataset Variations for Comprehensive Evaluation
- Model Training and Evaluation across Multiple Algorithms
- Performance Assessment and Comparison.



DATA UNDERSTANDING

What is the dimensions of our dataset?



`train.shape`

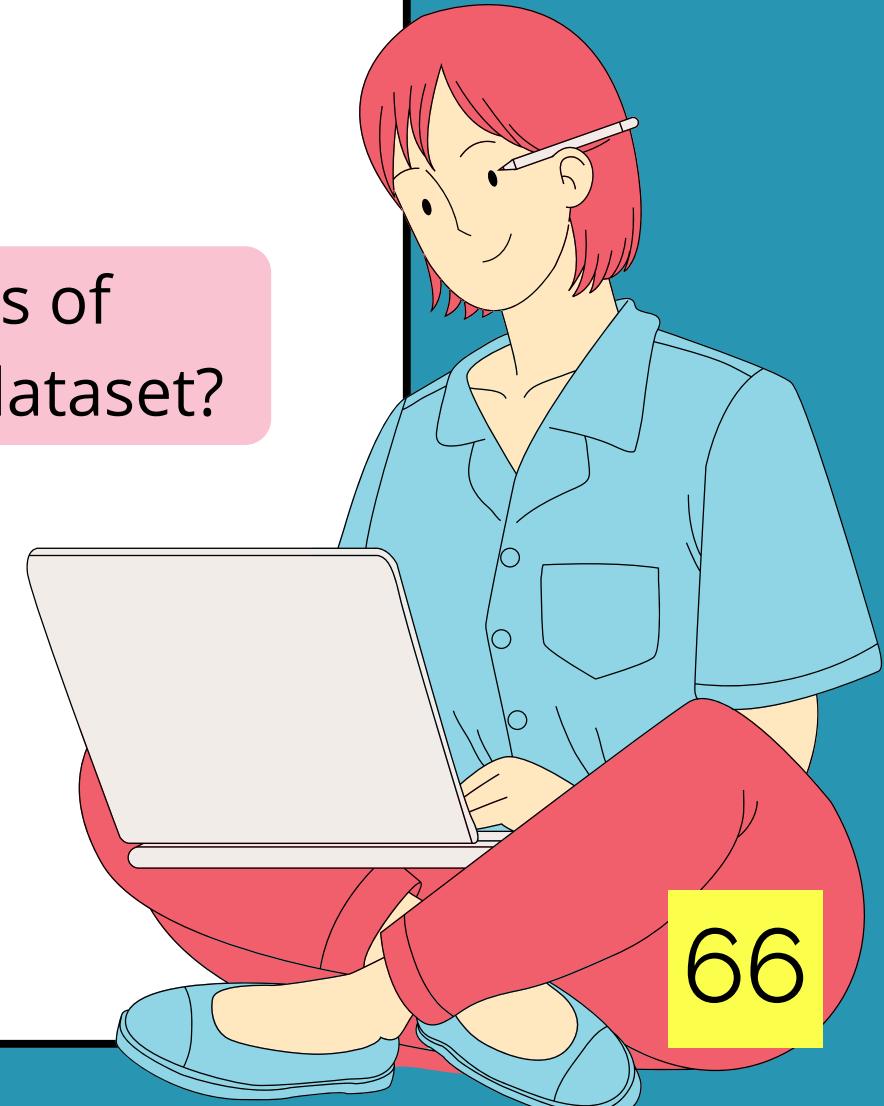
(125973, 43)

`test.shape`

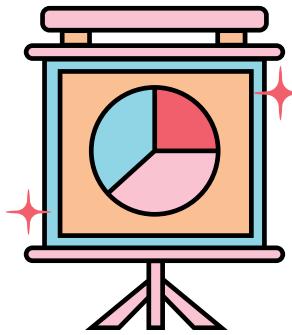
(22544, 43)

#	Column	Non-Null Count	Dtype
0	duration	22544	non-null int64
1	protocol_type	22544	non-null object
2	service	22544	non-null object
3	flag	22544	non-null object
4	src_bytes	22544	non-null int64
5	dst_bytes	22544	non-null int64
6	land	22544	non-null int64
7	wrong_fragment	22544	non-null int64
8	urgent	22544	non-null int64
9	hot	22544	non-null int64
10	num_failed_logins	22544	non-null int64
11	logged_in	22544	non-null int64
12	num_compromised	22544	non-null int64
13	root_shell	22544	non-null int64
14	su_attempted	22544	non-null int64
15	num_root	22544	non-null int64
16	num_file_creations	22544	non-null int64
17	num_shells	22544	non-null int64
18	num_access_files	22544	non-null int64
19	num_outbound_cmds	22544	non-null int64
20	is_host_login	22544	non-null int64
21	is_guest_login	22544	non-null int64
22	count	22544	non-null int64
23	srv_count	22544	non-null int64
24	serror_rate	22544	non-null float64
25	srv_serror_rate	22544	non-null float64
26	rerror_rate	22544	non-null float64
27	srv_rerror_rate	22544	non-null float64
28	same_srv_rate	22544	non-null float64
29	diff_srv_rate	22544	non-null float64
30	srv_diff_host_rate	22544	non-null float64
31	dst_host_count	22544	non-null int64
32	dst_host_srv_count	22544	non-null int64
33	dst_host_same_srv_rate	22544	non-null float64
34	dst_host_diff_srv_rate	22544	non-null float64
35	dst_host_same_src_port_rate	22544	non-null float64
36	dst_host_srv_diff_host_rate	22544	non-null float64
37	dst_host_serror_rate	22544	non-null float64
38	dst_host_srv_serror_rate	22544	non-null float64
39	dst_host_rerror_rate	22544	non-null float64
40	dst_host_srv_rerror_rate	22544	non-null float64
41	attack	22544	non-null object
42	level	22544	non-null int64

What are the distinct types of information available in our dataset?

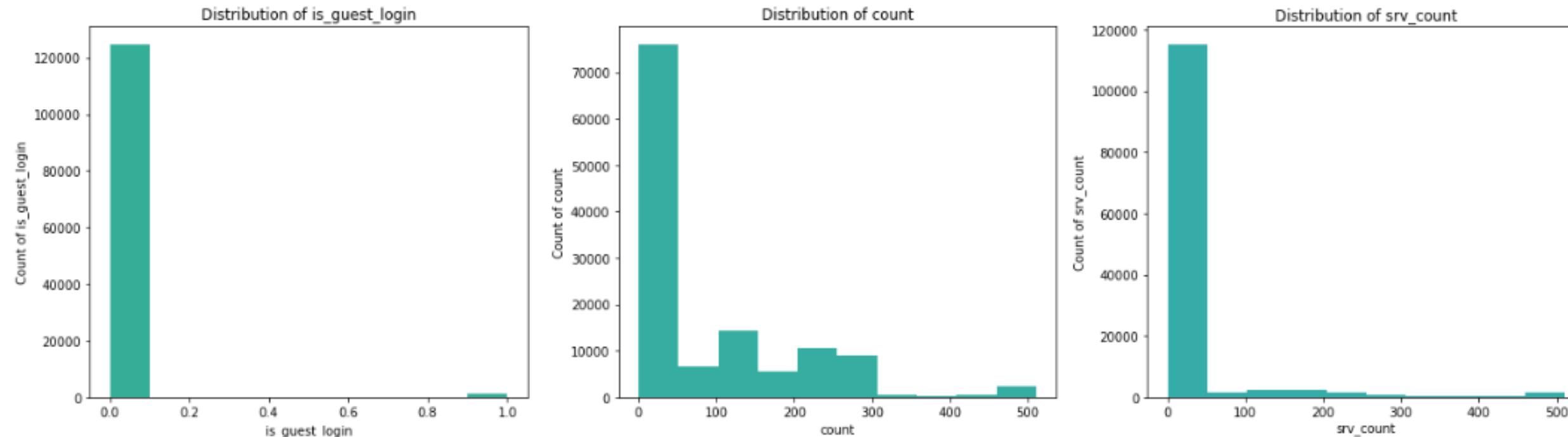
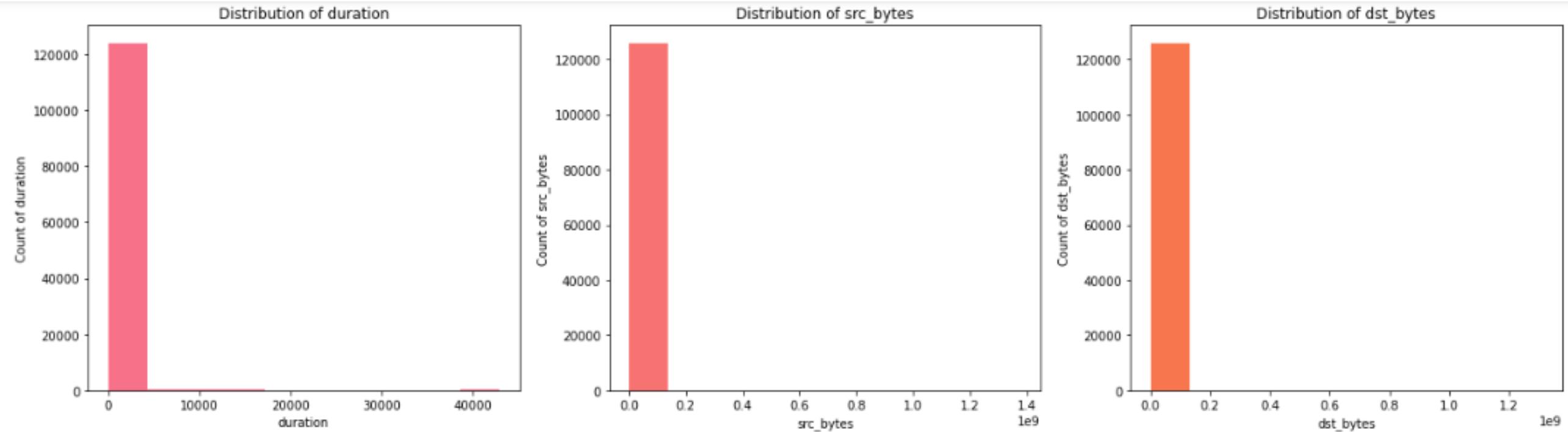


DATA UNDERSTANDING



Data visualisation

Some of
Quantitative



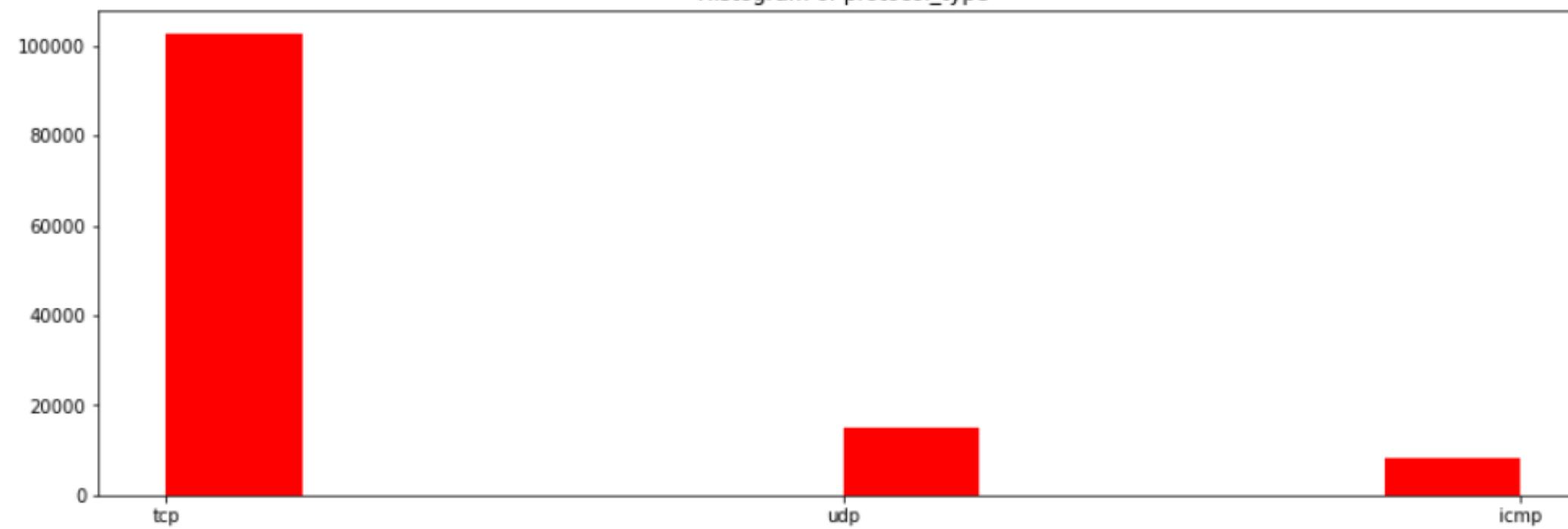
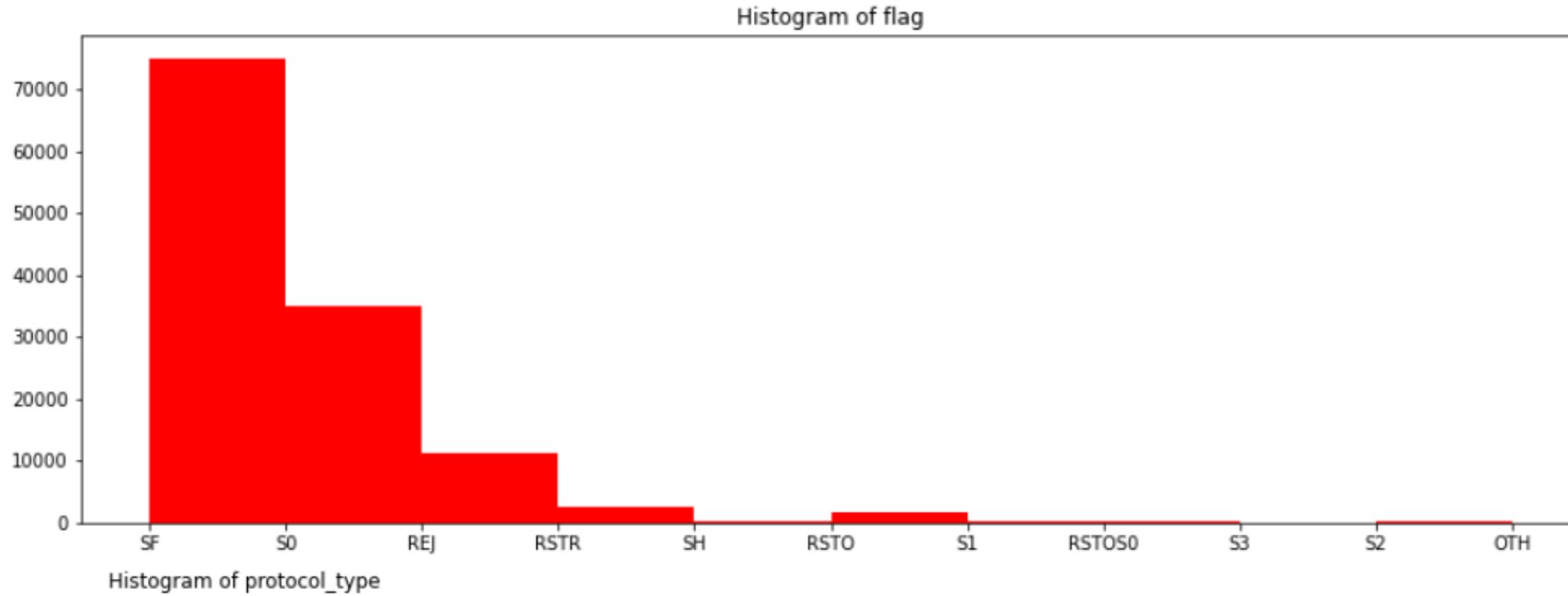
Variables



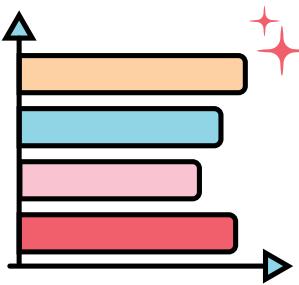
DATA UNDERSTANDING

Data visualisation

Some of
Qualitative



Variables



DATA PREPARATION

DATA CLEANING

Checking for missing values :

```
print('We have {} missing values in our Train set\nWe have {} missing values in the Test set')
```

```
We have 0 missing values in our Train set  
We have 0 missing values in the Test set
```

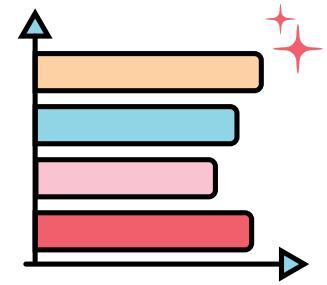
Checking for duplicated values :

```
print(test.duplicated().sum())  
print(train.duplicated().sum())
```

```
0  
0
```

==> No missing values/duplicates



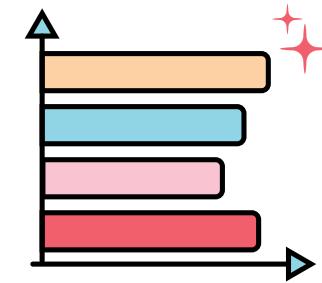


DATA PREPARATION

DATA CLEANING

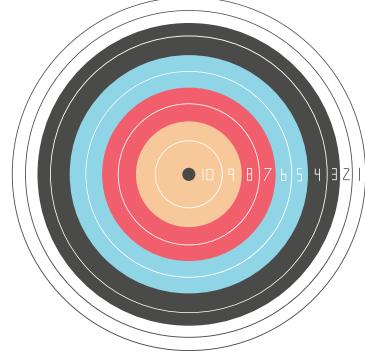
- *3 categorical dependant features*
==> *Remove ‘service’ and ‘flag’*
- *Remove the content related features*
- *For the ‘protocol’ feature : ‘tcp’ is the most important protocol.*
==> *Delete any protocol type value != ‘tcp’*





DATA PREPARATION

DATA TRANSFORMATION

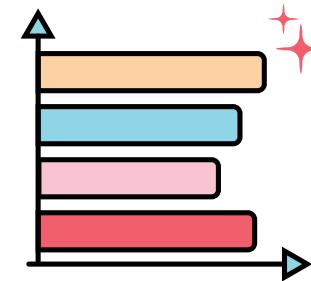


we have 8 datasets to create and alter , first we have to create the functions needed to apply on the raw datasets

We created 4 functions :

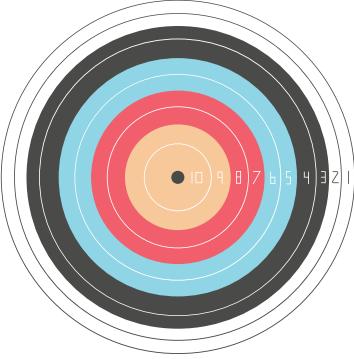
- Filter_row : Filtering the raw Data
- Normalize : Normalizing the data
- Perform_PCA : In this context, the focus is on PCA's ability to create uncorrelated features rather than reducing dimensionality
- GMM_Combine_Transform : It calculates the anomaly probabilities for each line in the supplied dataset against an original dataset.

We dropped the target from both test and train datasets .

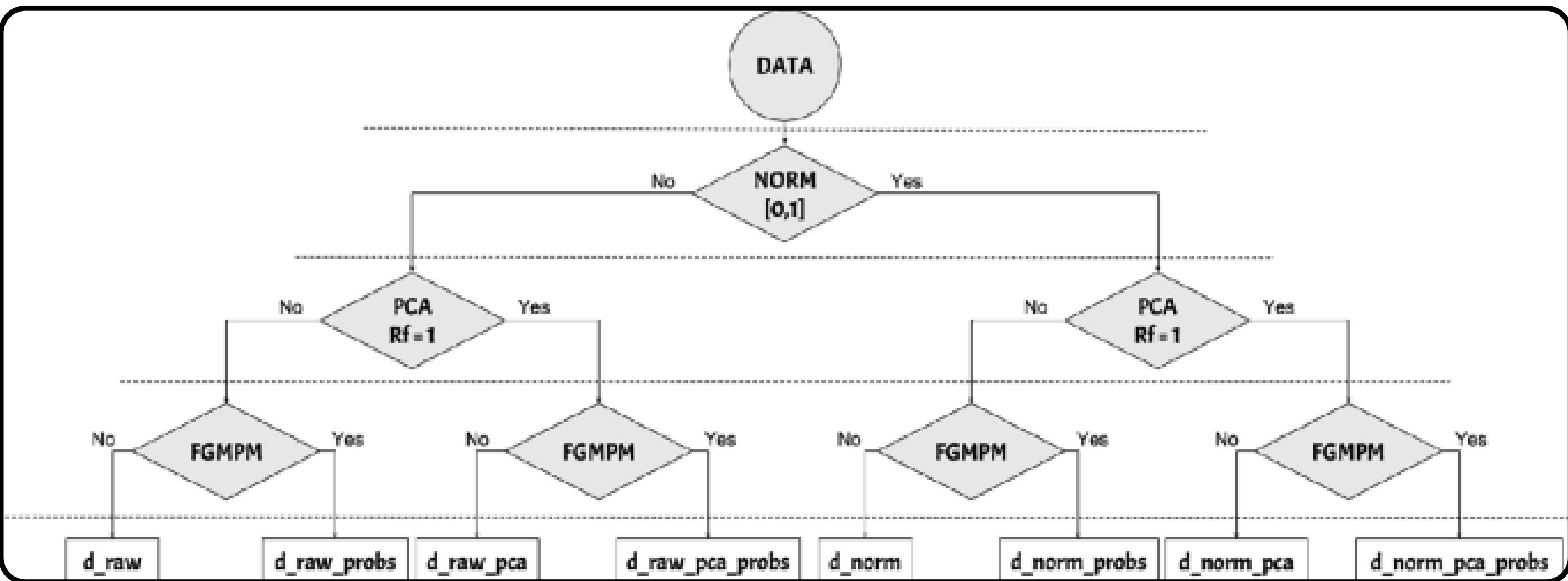


DATA PREPARATION

DATA TRANSFORMATION



These are the 8 datasets to create :



DATA PREPARATION



Applying filter function on :

Train and test datasets

```
[ ] print('Shape of raw train : {}\nshape of raw test : {}'.format(d_raw_train.shape,d_raw_test.shape))
```

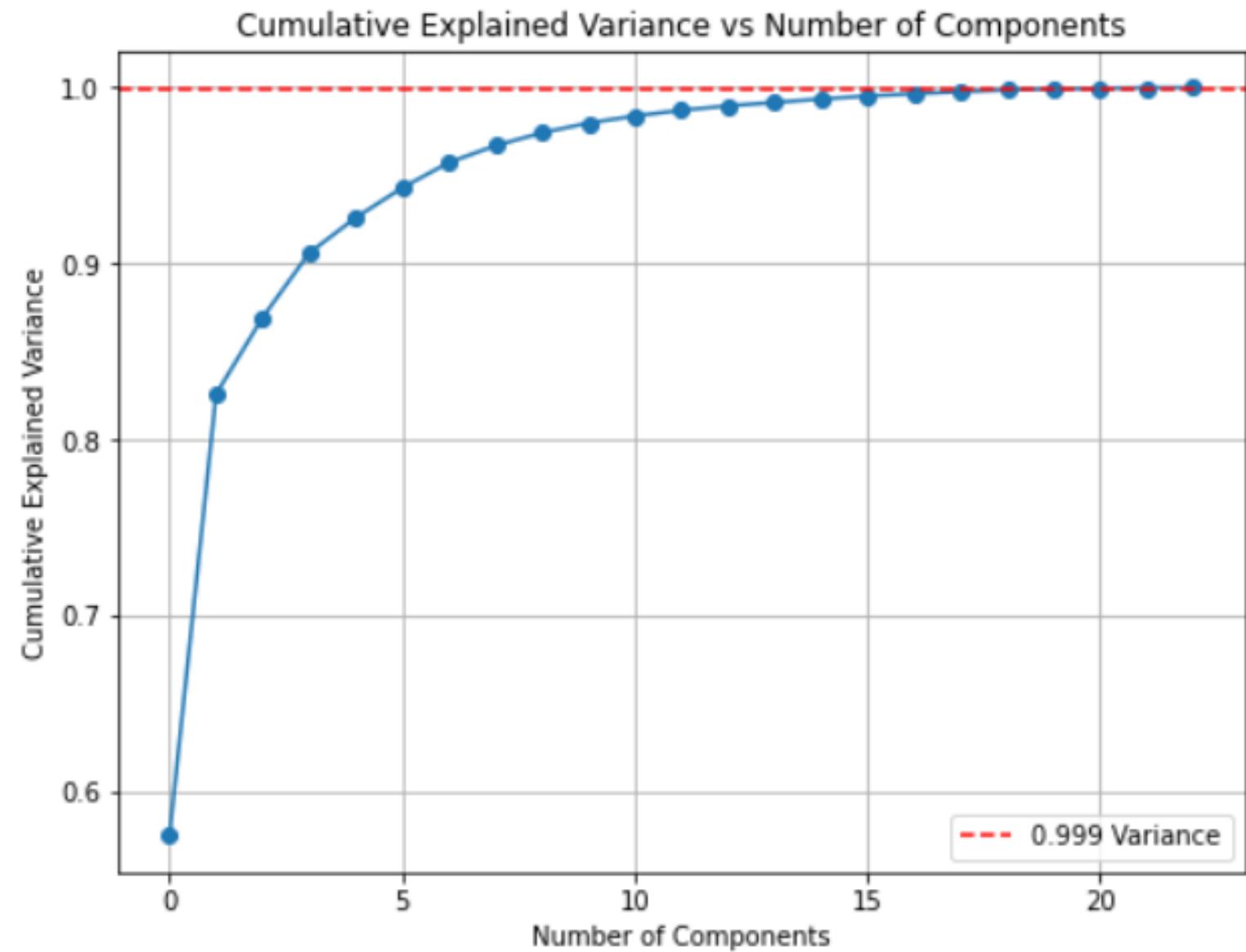
```
Shape of raw train : (102688, 24)
Shape of raw test : (18879, 24)
```

==> Now the number of features in both datasets are reduced to 24.
after applying filtering function mentionned before

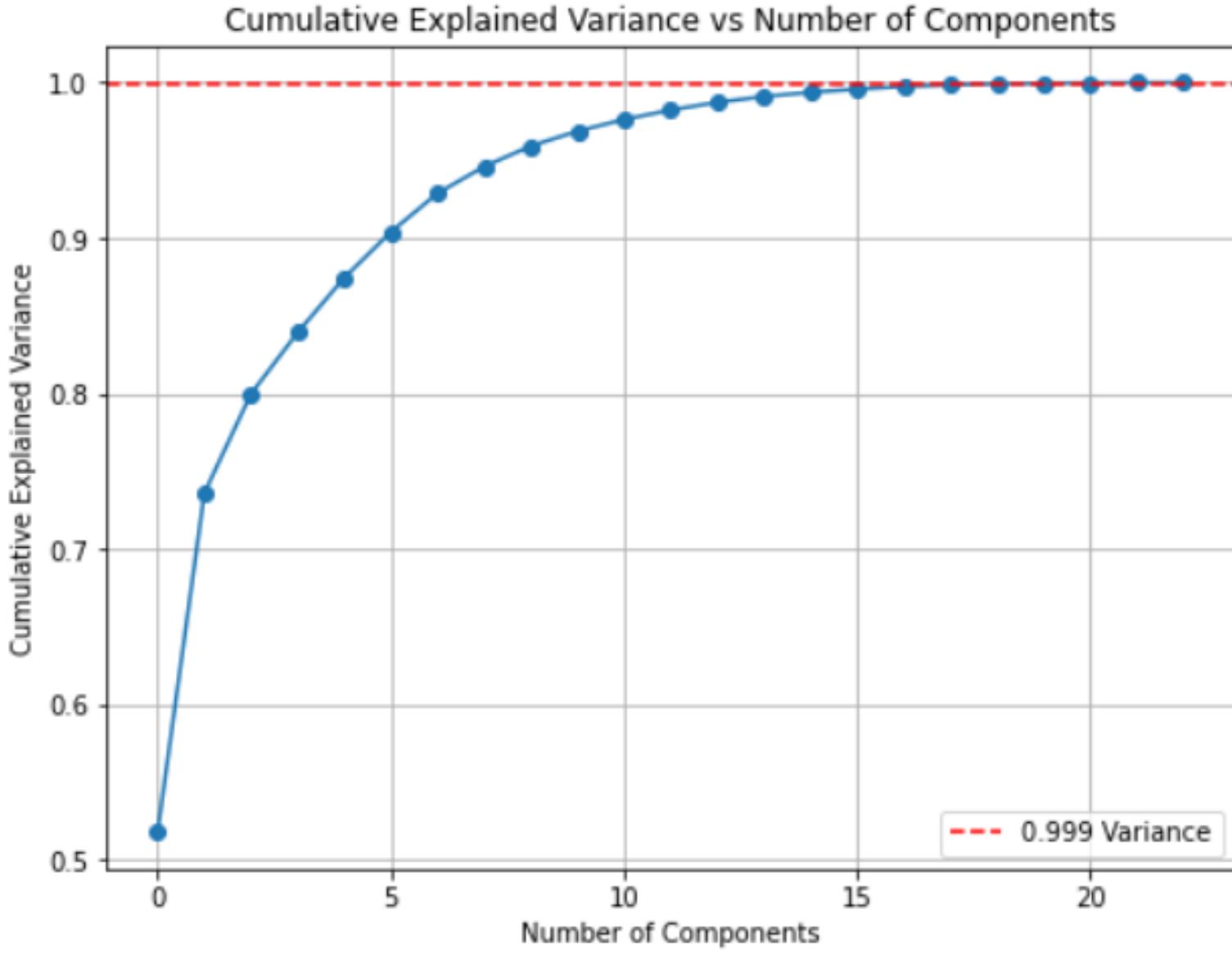
DATA PREPARATION

Applying PCA function on :

Train dataset.

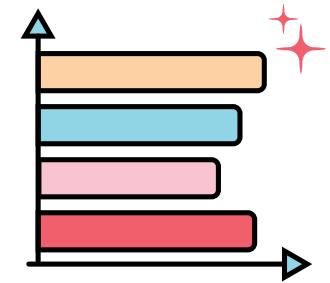


Test dataset.

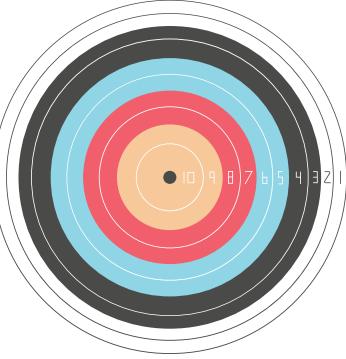


We applied the PCA on the train and test sets and we concluded that there is no information lost from the original data cause Varriance is equal to 0.99





MODELING

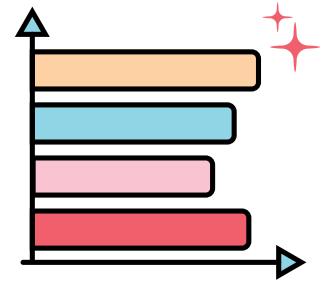


We created 6 models :

- Voting
- KM-D
- SVM
- KM-C
- DT
- MLP

We are going to apply them on the datasets we made.

EVALUATION



KM-D

```
• Evaluation of KM-D on d_raw :  
F1_score : 0.7378835483655325  
Sensitivity : 1.0  
CAP : 0.0
```

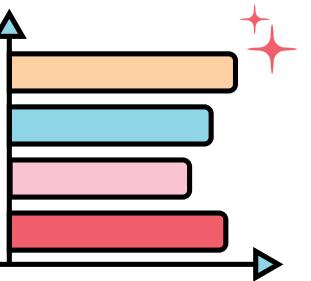
```
array([[    0,  7842],  
       [    0, 11038]], dtype=int64)
```

MLP

```
Evaluation of MLP on d_raw :  
F1_score : 0.7841424124938569  
Sensitivity : 0.6504801594491756  
CAP : 0.9007509353494735
```

KM-C

```
Evaluation of KM-C on d_raw_pca :  
F1_score : 0.0001810610175629187  
Sensitivity : 9.05961224859576e-05  
CAP : 0.14531217543547287
```



EVALUATION

The highest performance was observed with **K-Means** clusters on the **d_raw_pca_probs** dataset. Despite this, the voting scheme outperformed KM-C due to superior overall model performance, even though KM-CG demonstrated higher anomaly detection rates. Notably, KM-C required attack information during training and lacked a strictly normal model, whereas the voting scheme relied on computed occurrence probabilities. Fine-tuning the voting scheme's hyperparameters revealed a trade-off between alpha and consensus for optimal performance.