



Klasifikasi Inefisiensi Klaim Peserta BPJS Kesehatan Menggunakan XGBoost

The JamMaths

Anggota The JamMaths



Cornelius Justin S. Hadi
Universitas Indonesia



M. Hanif Pramudya Z.
Universitas Indonesia



Tulus Setiawan
Universitas Indonesia

Table of Contents

01

Data
Preparation

02

Data
Cleaning

03

Feature
Selection

04

Feature
Engineering

05

Modelling



01

Data Preparation

Dataset yang Digunakan

sampling_healthkathon2022.csv

	id	id_peserta	dati2	typefaskes	usia	jenkel	pisat	tgldatang	tgipulang	jenispele	...
0		165666	486	17	KL	48	P	1.0	2018-07-25T17:00:00.000Z	2018-07-25T17:00:00.000Z	2 ...
1		1010828	520	17	A	63	L	1.0	2019-05-27T17:00:00.000Z	2019-05-30T17:00:00.000Z	1 ...

merupakan data kunjungan peserta JKN ke fasilitas kesehatan rujukan tingkat lanjut

Jumlah baris :
11401882

Jumlah Kolom :
22

sampling_healthkathon2022_diagnosa.csv

	id	diag	levelid
0	6	O06.9	1
1	57	J02.9	1

merupakan data yang berisi diagnosa penyakit peserta, di mana dalam satu kunjungan peserta bisa memiliki lebih dari satu diagnosa.

Jumlah baris :
11401882

Jumlah Kolom :
3

sampling_healthkathon2022_procedure.csv

	id	proc
0	6	90.59
1	6	69.01

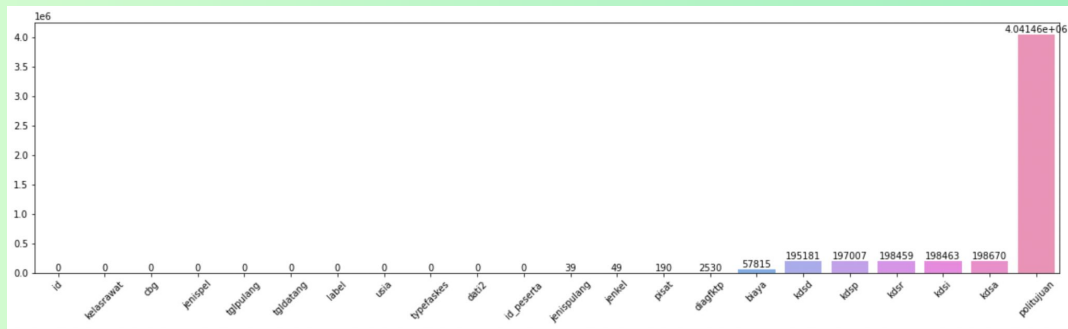
merupakan data yang berisi prosedur/tindakan medis yang didapatkan peserta JKN. Dalam setiap kunjungan, peserta bisa mendapatkan satu atau lebih prosedur/tindakan medis.

Jumlah baris :
12202871

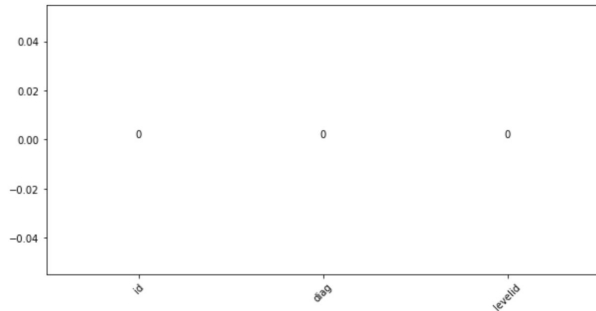
Jumlah Kolom :
2



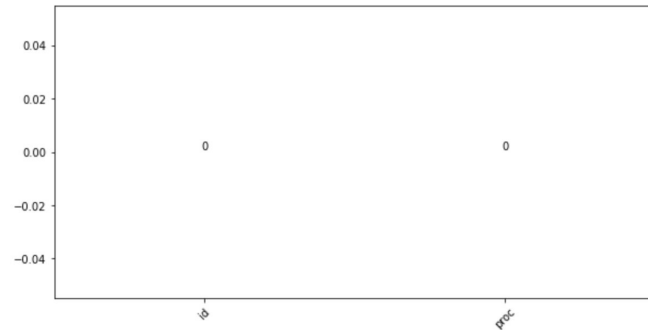
Missing Value pada sampling_healthkathon2022.csv



Missing Value pada sampling_healthkathon2022_diagnosa.csv



Missing Value pada sampling_healthkathon2022_procedure.csv



Data Duplikat pada sampling_healthkathon2022.csv

```
df.duplicated().sum()
```

14

Data Duplikat pada sampling_healthkathon2022_diagnosa.csv

```
df_diagnosa.duplicated().sum()
```

56

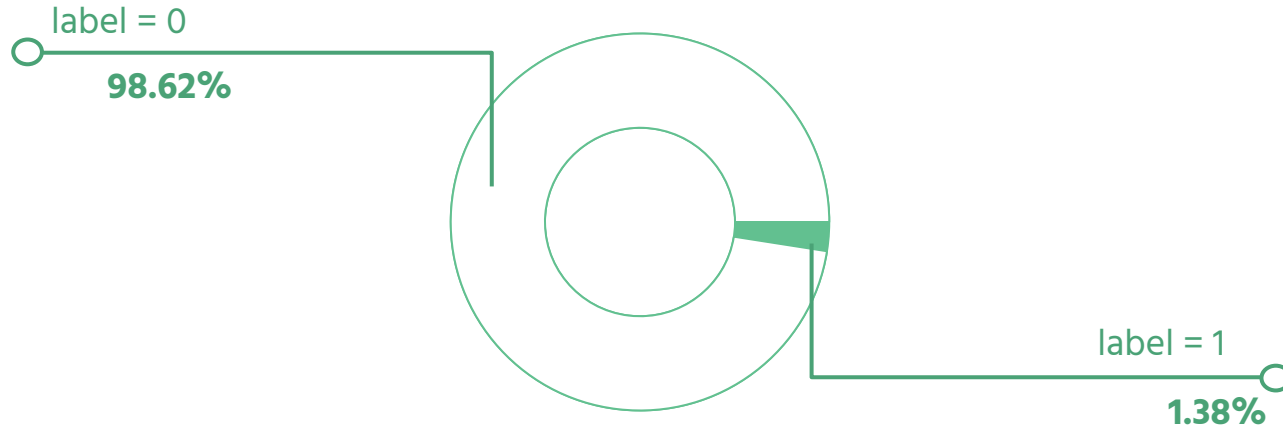
Data Duplikat pada sampling_healthkathon2022_procedure.csv

```
df_proc.duplicated().sum()
```

47

Counts pada Kolom label

(Dataset sampling_healthkathon2022.csv)





02

Data Cleaning



Dataset sampling_healthkathon2022.csv

1

Meng-handle Nilai *Null*

2

Meng-handle Data Duplikat

3

Menyamakan Format
Penamaan Karakter

Meng-handle Nilai Null

Kolom Kategorik

'jenispulang', 'jenkel', 'pisat',
'diagfktp', 'kdsa', 'kdsp', 'kdsr',
'kdsi', 'kdsd'

Handle :

ubah menjadi modus kolom kategorik tsb.

Syntax :

```
SimpleImputer(strategy='most_frequent')
```



Kolom Numerik

'biaya'

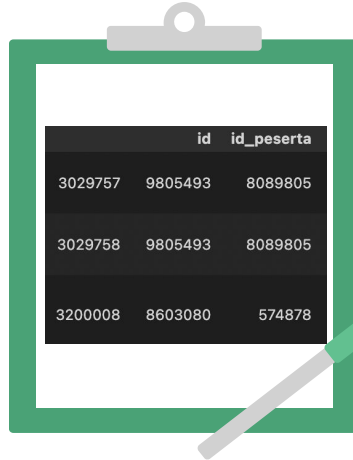
Handle :

ubah menjadi median biaya tsb..

Syntax :

```
SimpleImputer(strategy='median')
```

Meng-handle Data Duplikat



	id	id_peserta
3029757	9805493	8089805
3029758	9805493	8089805
3200008	8603080	574878

Melakukan metode drop data duplikat

syntax:
`df.drop_duplicates()`



	id	id_peserta
3029757	9805493	8089805

Menyamakan Format Penamaan Karakter

Kolom Kategorik

'typefaskes', 'jenkel', 'politujuan',
'diagfktpt', 'cbg', 'kdsa', 'kdsp',
'kdsr', 'kdsi', 'kdsd'

Format Karakter :

Samakan huruf kapital

Syntax :

`df[obj_col].str.upper()`



Kolom Kategorik

'typefaskes', 'jenkel',
'politujuan', 'diagfktpt', 'cbg',
'kdsa', 'kdsp', 'kdsr', 'kdsi',
'kdsd'

Format Karakter :

Hilangkan Whitespaces

Syntax :

`df[obj_col].str.strip()`

Dataset

sampling_healthkathon2022_diagnosa.csv

1

Meng-*handle* Data Duplikat

2

Menyamakan Format Karakter Kolom “diag”

3

Mengambil hanya diagnosa primer (baris dengan levelid = 1)

4

Diagnosa sekunder digunakan untuk membuat kolom berisi frekuensi diagnosa sekunder masing-masing id

Hasil dari Proses Cleaning Dataset sampling_healthkathon2022_diagnosa.csv

	id	diag	diag_sekunder_counts
0	6	O06	0
1	57	J02	0
2	91	R10	0
3	109	R18	0
4	111	N81	1

Ket :

diag = diagnosa primer dari FKRTL

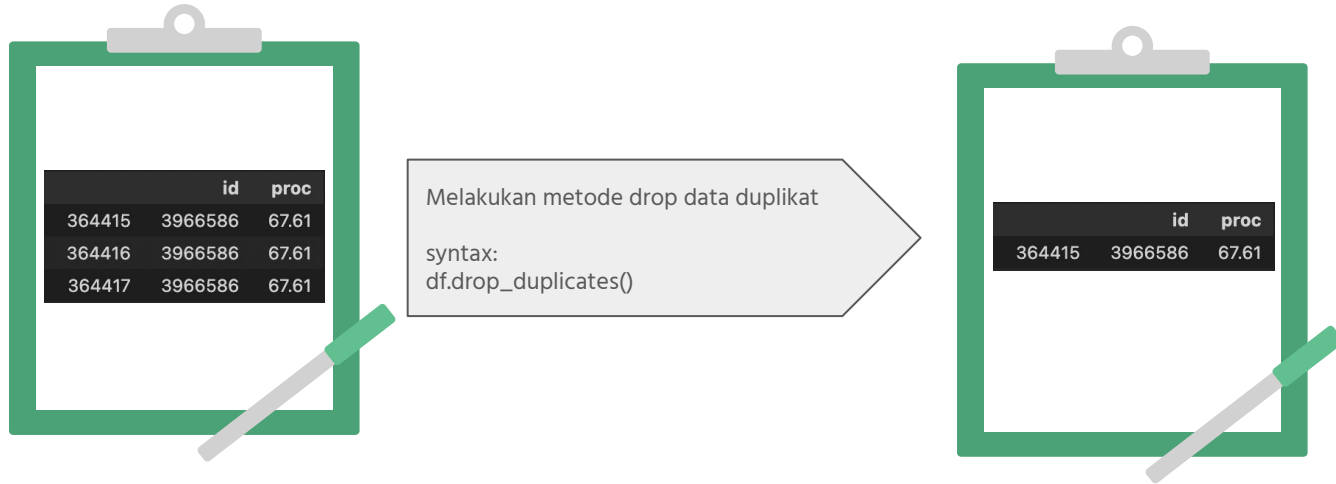
Dataset

sampling_healthkathon2022_procedure.csv

1

Meng-handle Data Duplikat

Meng-handle Data Duplikat





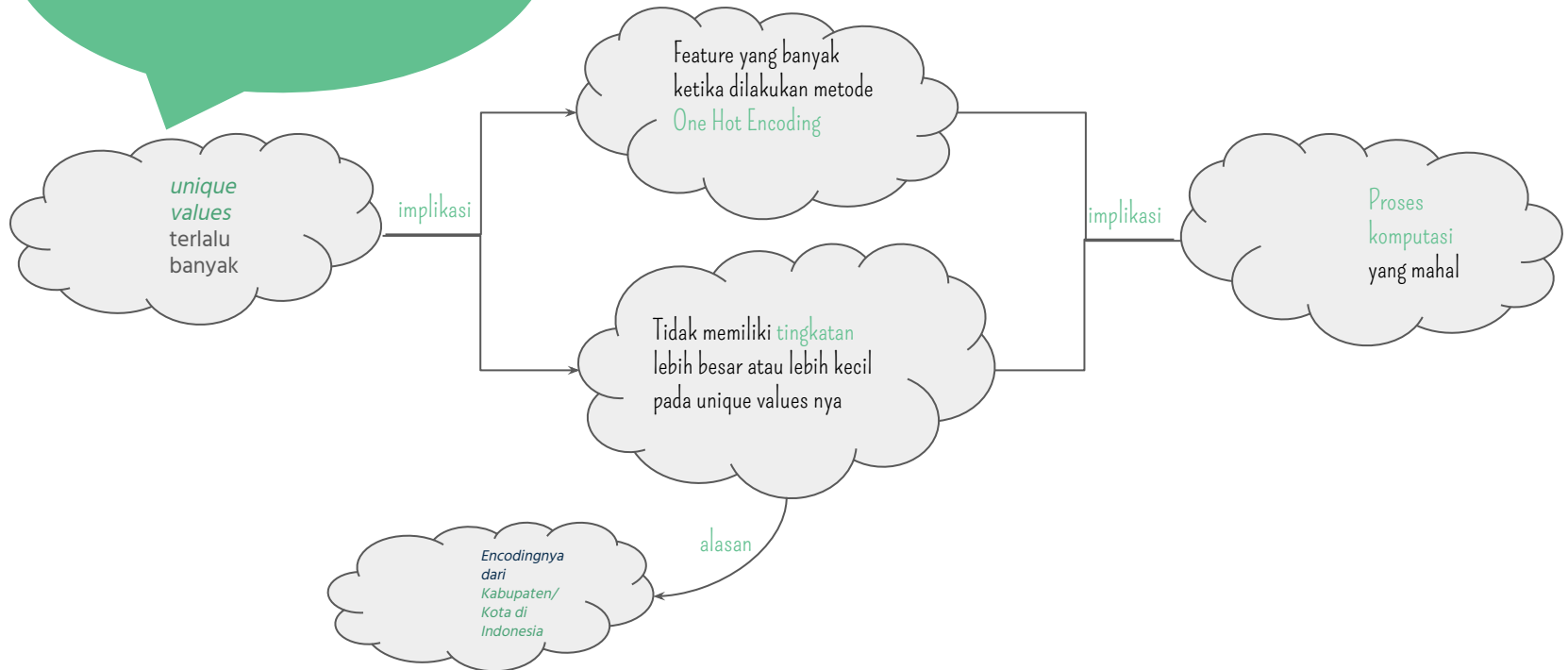
03

Feature Selection



Drop fitur dati2

Mengapa fitur dati2 di-drop atau dihilangkan dari dataset?

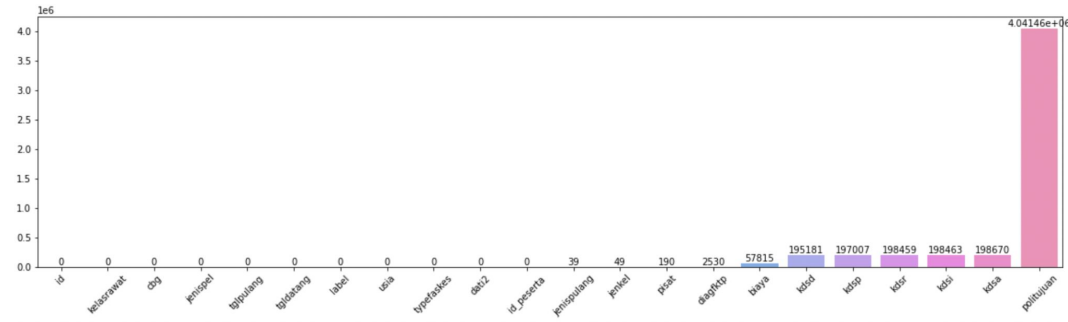


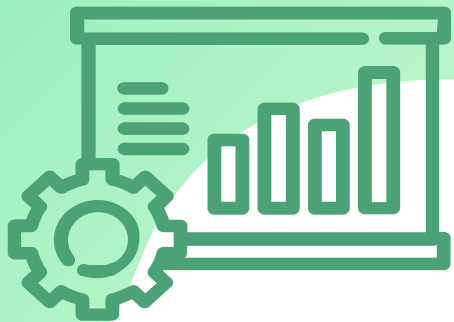
Drop fitur politujuan

Mengapa fitur politujuan di-drop dari dataset?



Karena jumlah missing value yang terlalu banyak.





04

Feature Engineering



Membuat Kolom Lama Perawatan Peserta

(jika nilai tiap baris > 0 , maka peserta rawat inap)

```
7  tgl datang    11401882 non-null object
8  tgl pulang    11401882 non-null object
```

Ubah menjadi
tipe datetime

```
15 tgl datang    datetime64[ns, UTC]
16 tgl pulang    datetime64[ns, UTC]
```

tgl pulang - tgl datang

lama_rawat

0

3

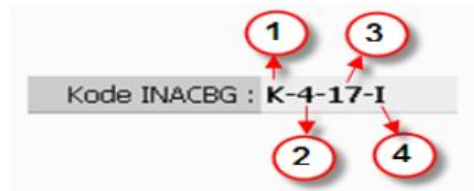
0

0

0

Memisahkan Nilai-Nilai pada Kolom "cbg", "kdsa", "kdsp", "kdsr", "kdsi", "kdsd" (sesuai struktur kodenya) Menjadi Kolom Baru

Gambar 1
Struktur Kode INA-CBG



Keterangan :

1. Digit ke-1 merupakan CMG (*Casemix Main Groups*)
2. Digit ke-2 merupakan tipe kasus
3. Digit ke-3 merupakan spesifik CBG kasus
4. Digit ke-4 berupa angka romawi merupakan *severity level*

Sumber: Permenkes RI Nomor 27 Tahun 2014 Tentang Petunjuk Teknis Sistem INA-CBGs

'Q-5-42-0'

Pisahkan tiap digit,
lalu masukkan ke
dalam kolom baru

cbg_CMG	cbg_tipekasus	cbg_spesifikkasus	cbg_severity
Q	5	42	0

Membuat Dua Kolom Baru Berdasarkan 'diagfktp'

diagfktp
L02.8

Pisah menjadi 2 digit
dengan mengacu pada
website untuk
mengecek diagnosa



diagfktp_letter	diagfktp_num
L	2

ICD-10 Version:2016

Search

- ▼ XII Diseases of the skin and subcutaneous tissue
 - ▼ L00-L08 Infections of the skin and subcutaneous tissue
 - L00 Staphylococcal scalded skin syndrome
 - ▶ L01 Impetigo
 - ▶ L02 Cutaneous abscess, furuncle and carbuncle

Merge Dataframe Utama dengan Dataframe Diagnosa

Dataframe utama (left)

diagfktl_letter	diagfktl_num
L	2
R	23
E	10
H	54
M	54



Dataframe diagnosa (right)

id	diagfktl_letter	diagfktl_num	diagfktl_sekunder_counts
6	O	6	0
57	J	2	0
91	R	10	0
109	R	18	0

Left-join

diagfktl_letter	diagfktl_num	diagfktl_letter	diagfktl_num	diagfktl_sekunder_counts
L	2	H	60.0	0.0
R	23	D	64.0	1.0
E	10	E	11.0	2.0
H	54	H	52.0	2.0
M	54	M	54.0	0.0

Membuat Kolom Berisikan Kode ICD-10, Yang Berkaitan Dengan 'diagfktp' dan 'diagfkrtl' Masing-Masing Baris

diagfktp_letter diagfktp_num

L 2

```
diagfktp_icd10 = []
for i in range(df_no_dup_prepared.shape[0]):
    letter = df_no_dup_prepared.loc[i, 'diagfktp_letter']
    num = df_no_dup_prepared.loc[i, 'diagfktp_num']
    if (letter in ['A', 'B']):
        diagfktp_icd10.append('I')
    elif (letter == 'C') or (letter == 'D' and num <= 48):
        diagfktp_icd10.append('II')
    elif letter == 'D' and num >= 50:
        diagfktp_icd10.append('III')
    elif letter == 'E':
        diagfktp_icd10.append('IV')
    elif letter == 'F':
        diagfktp_icd10.append('V')
    elif letter == 'G':
        diagfktp_icd10.append('VI')
    elif letter == 'H' and num <= 59:
        diagfktp_icd10.append('VII')
    elif letter == 'H' and num >= 60:
        diagfktp_icd10.append('VIII')
    elif letter == 'I':
        diagfktp_icd10.append('IX')
    elif letter == 'J':
        diagfktp_icd10.append('X')
    elif letter == 'K':
        diagfktp_icd10.append('XI')
    elif letter == 'L':
        diagfktp_icd10.append('XII')
```

diagfktp_icd10

XII

diagfkrtl_letter diagfkrtl_num

H 80.0

```
diagfkrtl_icd10 = []
for i in range(df_no_dup_prepared.shape[0]):
    letter = df_no_dup_prepared.loc[i, 'diagfkrtl_letter']
    num = df_no_dup_prepared.loc[i, 'diagfkrtl_num']
    if (letter in ['A', 'B']):
        diagfkrtl_icd10.append('I')
    elif (letter == 'C') or (letter == 'D' and num <= 48):
        diagfkrtl_icd10.append('II')
    elif letter == 'D' and num >= 50:
        diagfkrtl_icd10.append('III')
    elif letter == 'E':
        diagfkrtl_icd10.append('IV')
    elif letter == 'F':
        diagfkrtl_icd10.append('V')
    elif letter == 'G':
        diagfkrtl_icd10.append('VI')
    elif letter == 'H' and num <= 59:
        diagfkrtl_icd10.append('VII')
    elif letter == 'H' and num >= 60:
        diagfkrtl_icd10.append('VIII')
```

diagfkrtl_icd10

VIII

Manipulasi Data Frame Procedure

id	proc_count
6525898	41
6545941	37
6552868	36
6557462	32
6549772	32
...	...
2294151	1
2294137	1
443730	1
443780	1
11406846	1

Dibuat data frame baru dimana untuk setiap id yang berbeda, dihitung frekuensi jenis procedure yang dilakukan

Merge All Data Frame

Berikut merupakan hasil dari *merge data frame* seluruhnya yang sudah dilakukan *engineering* sebelumnya.

	jenispulang	jenkel	pisat	biaya	id	id_peserta	typefaskes	usia	jenispel	kelasrawat	label	lama_rawat	cbg_CMG	cbg_tipekasus	cbg_spesifikkasus	cbg_severity	kdsa_CMG	kdsa_tipekasus	
0	1.0	P	1.0	184300.0	165666	486	KL	48	2	3	0	0	Q	5	42	0	-	-	
1	1.0	L	1.0	10628400.0	1010628	520	A	63	1	1	0	3	D	4	13	III	-	-	
2	1.0	P	1.0	187300.0	166042	523	KL	53	2	3	0	0	Q	5	44	0	-	-	
3	1.0	P	1.0	187300.0	168937	549	KL	54	2	3	0	0	Q	5	44	0	-	-	
4	1.0	P	1.0	381600.0	1005899	549	A	53	2	3	0	0	Q	5	44	0	-	-	

esifikkasus	kdsa_severity	kdsp_spesifikkasus	kdsp_severity	kdsr_spesifikkasus	kdsr_severity	kdsi_spesifikkasus	kdsd_spesifikkasus	kdsd_severity	diagfkr1l_sekunder_counts	diagfkt1c1d10	diagfkr1l_1cd10	proc_count
-	-	-	-	-	-	-	-	-	0.0	XII	VIII	NaN
-	-	-	-	-	-	-	-	-	1.0	XVIII	III	1.0
-	-	-	-	-	-	-	-	-	2.0	IV	IV	NaN
-	-	-	-	-	-	-	-	-	2.0	VII	VII	NaN
-	-	-	-	-	-	-	-	-	0.0	XIII	XIII	NaN

Adanya data *null* merupakan akibat dari ketidak adanya beberapa id di data frame utama+diagnosa pada data frame procedure setelah dilakukannya proses merge data frame

Data Frame Prepared

	jenispulang	jenkel	pisat	biaya	id	id_peserta	typefaskes	usia	jenispel	kelasrawat	label	lama_rawat	cbg_CMG	cbg_tipekasus	cbg_spesifikkasus	cbg_severity	kdsa_CMG	kdsa_tipekasus	
0	1.0	P	1.0	184300.0	165666	486	KL	48	2	3	0	0	Q	5	42	0	-	-	
1	1.0	L	1.0	10628400.0	1010628	520	A	63	1	1	0	3	D	4	13	III	-	-	
2	1.0	P	1.0	187300.0	166042	523	KL	53	2	3	0	0	Q	5	44	0	-	-	
3	1.0	P	1.0	187300.0	168937	549	KL	54	2	3	0	0	Q	5	44	0	-	-	
4	1.0	P	1.0	381600.0	1005899	549	A	53	2	3	0	0	Q	5	44	0	-	-	

esifikkasus	kdsa_severity	kdsp_spesifikkasus	kdsp_severity	kdsr_spesifikkasus	kdsr_severity	kdsi_spesifikkasus	kdsd_spesifikkasus	kdsd_severity	diagfkrtl_sekunder_counts	diagfktp_icd10	diagfkrtl_icd10	proc_count
-	-	-	-	-	-	-	-	-	0.0	XII	VIII	0.0
-	-	-	-	-	-	-	-	-	1.0	XVIII	III	1.0
-	-	-	-	-	-	-	-	-	2.0	IV	IV	0.0
-	-	-	-	-	-	-	-	-	2.0	VII	VII	0.0
-	-	-	-	-	-	-	-	-	0.0	XIII	XIII	0.0

Untuk setiap data *null* pada fitur 'proc_count' sudah digantikan dengan nilai 0. Kenapa 0? Karena penyebab dari adanya data *null* tersebut seperti yang sudah dijelaskan pada slide sebelumnya. Hal itu berarti untuk setiap id yang 'proc_count' nya bernilai 'Nan', maka tidak ada jenis procedure apapun yang dilakukan terhadap pasien dengan nomor id tersebut. Hal ini yang mendasari kenapa nilai 'Nan' digantikan dengan nilai 0.

Preprocessing

Sebelum melakukan modelling, dilakukan **data preprocessing** dengan metode sebagai berikut

One Hot Encoding:

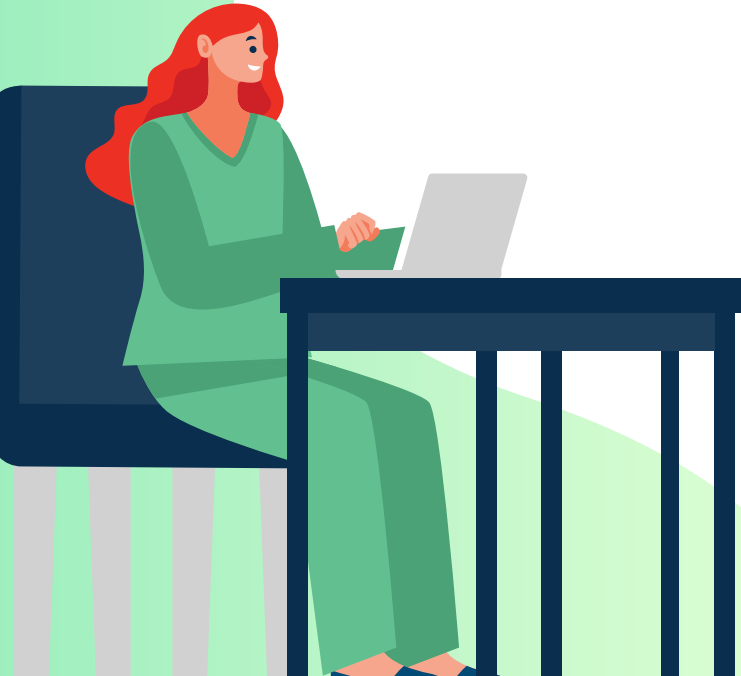
```
['cbg_CMG', 'cbg_tipekasus', 'cbg_spesifikkasus',  
'kdsa_CMG', 'kdsa_tipekasus', 'kdsa_spesifikkasus',  
'kdsp_spesifikkasus', 'kdsr_spesifikkasus',  
'kdsi_spesifikkasus', 'kdsd_spesifikkasus',  
'diagfktp_icd10', 'diagfkrtl_icd10', 'typefaskes']
```

Ordinal Encoding:

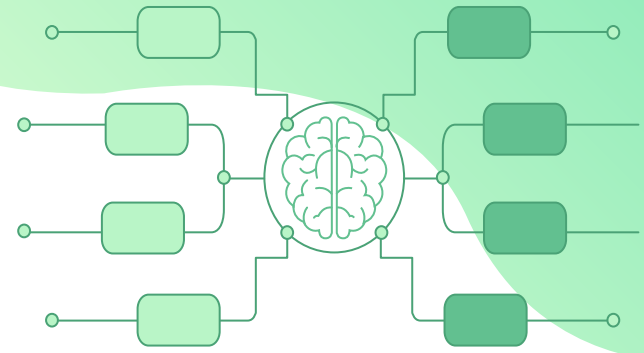
```
['jenkel', 'cbg_severity', 'kdsa_severity',  
'kdsp_severity', 'kdsr_severity', 'kdsd_severity']
```

```
<11401868x777 sparse matrix of type '<class 'numpy.float64'>'  
with 247693071 stored elements in Compressed Sparse Row format>
```

Setelah preprocessing diperoleh sebanyak **777 kolom** (akibat dari One Hot Encoding) yang berbentuk **sparse matrix**



-
-



05 Modelling

XGBoost / Light GBM

eXtreme Gradient Boosting / Light Gradient Boosting Machine

Imbalanced Dataset

Gradient Boosting bekerja dengan baik pada dataset yang tidak seimbang karena pada setiap pembuatan tree, algoritma ini memperbaiki error tree sebelumnya

Fast

XGBoost dan Light GBM dapat melakukan komputasi paralel ketika membuat tree, serta kita dapat memanfaatkan GPU

Optimized

Kedua algoritma ini menggunakan algoritma Gradient Boosting yang sudah dioptimalkan, seperti memiliki hyperparameter regularisasi untuk mencegah *overfitting*

Pemilihan Algoritma (Optimasi Model): Hyperparameter Tuning XGBoost

Hyperparameter tuning menggunakan algoritma **Bayesian Search Optimization** dengan cross-validation

Data yang diambil untuk hyperparameter tuning sebanyak 25% dari keseluruhan

```
X_train, X_test, y_train, y_test = train_test_split(X_prepared, y,
                                                    test_size=.75)
X_train.shape, y_train.shape
((2850467, 776), (2850467,))
```

Dibuat pipeline model dengan over-undersampling

```
model_pipeline = imbpipeline([
    ('over', SMOTE(sampling_strategy=.15)),
    ('under', RandomUnderSampler(sampling_strategy=.5)),
    ('xgb', xgb.XGBClassifier(n_estimators=50, objective='binary:logistic',
                             verbosity=2))
])
```

Hyperparameter yang dilakukan *tuning*:

1. Banyak data yang diover-under sampling
2. K-neighbors pada SMOTE
3. Learning rate
4. Max depth
5. Banyak subsample untuk training
6. Banyak sampel kolom untuk training
7. Nilai regularisasi L1 dan L2

```
params = {'over__sampling_strategy': Real(.1, .6, 'log-uniform'),
          'over__k_neighbors': Integer(5, 1000),
          'under__sampling_strategy': Real(.6, .9, 'log-uniform'),
          'xgb__learning_rate': Real(1e-2, 1),
          'xgb__max_depth': Integer(6, 64),
          'xgb__subsample': Real(0.5, 1),
          'xgb__colsample_bytree': Real(0.5, 1),
          'xgb__reg_alpha': Real(1e-2, 10, 'log-uniform'),
          'xgb__reg_lambda': Real(1e-2, 10, 'log-uniform'),
          'xgb__gamma': Real(0, 5),
          'xgb__min_child_weight': Integer(1, 10)}

search = BayesSearchCV(model_pipeline, params, scoring='roc_auc', cv=3,
                       verbose=10, n_jobs=-1, n_iter=100,
                       return_train_score=True)
```

```
search.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 1 candidates, totalling 3 fits
Fitting 3 folds for each of 1 candidates, totalling 3 fits
```

Pemilihan Algoritma (Optimasi Model): Hyperparameter Tuning XGBoost

Hasil

Hyperparameter	Value
Over-sampling strategy	0.20
SMOTE k-neighbors	858
Under-sampling strategy	0.88
Learning rate	0.15
Max depth	64
Subsample	1.0
Column sample	0.5
L1 regularization	1.88
L2 regularization	0.01

Pemilihan Algoritma (Optimasi Model): Hyperparameter Tuning Light GBM

Hyperparameter tuning menggunakan algoritma **Bayesian Search Optimization** dengan cross-validation

Data yang diambil untuk hyperparameter tuning sebanyak 25% dari keseluruhan

```
X_train, X_test, y_train, y_test = train_test_split(X_prepared, y,
                                                    test_size=.75)
X_train.shape, y_train.shape
((2850467, 776), (2850467,))
```

Dibuat pipeline model dengan over-undersampling

```
model_pipeline = imbpipeline([
    ('over', SMOTE(sampling_strategy=.15)),
    ('under', RandomUnderSampler(sampling_strategy=.5)),
    ('lgbm', lgbm.LGBMClassifier(n_estimators=100, objective='binary',
                                verbosity=1, n_jobs=-1))
])
```

Hyperparameter yang dilakukan *tuning*:

1. Banyak data yang diover-under sampling
2. K-neighbors pada SMOTE
3. Tipe algoritma boosting
4. Learning rate
5. Max depth
6. Banyak sampel kolom untuk training
7. Nilai regularisasi L1 dan L2
8. Minimum split gain
9. Banyak maksimum leave pada tree
10. Banyaknya data minimum pada child (leaf)

```
params = {'over__sampling_strategy': Real(.1, .6, 'log-uniform'),
          'over__k_neighbors': Integer(5, 1000),
          'under__sampling_strategy': Real(.6, .9, 'log-uniform'),
          'lgbm__boosting_type': Categorical(['gbdt', 'dart', 'goss']),
          'lgbm__learning_rate': Real(1e-2, 1),
          'lgbm__max_depth': Integer(3, 32),
          'lgbm__colsample_bytree': Real(0.5, 1),
          'lgbm__reg_alpha': Real(1e-2, 10, 'log-uniform'),
          'lgbm__reg_lambda': Real(1e-2, 10, 'log-uniform'),
          'lgbm__min_split_gain': Real(0, 5),
          'lgbm__num_leaves': Integer(10, 10000),
          'lgbm__min_child_samples': Integer(200, 2000)}
```

```
search = BayesSearchCV(model_pipeline, params, scoring='roc_auc', cv=3,
                        verbose=10, n_jobs=-1, n_iter=200,
                        return_train_score=True)
```

```
search.fit(X_train, y_train)
```

Fitting 3 folds for each of 1 candidates, totalling 3 fits

Pemilihan Algoritma (Optimasi Model): Hyperparameter Tuning **Light GBM**

Hasil

Hyperparameter	Value
Over-sampling strategy	0.10
Under-sampling strategy	0.70
SMOTE k-neighbors	1000
Boosting type	dart
Learning rate	0.13
Max Depth	25
Column sample	0.55
L1 regularization	0.01
L2 regularization	0.01
Min Split Gain	0.13
Num leaves	3213
Min child samples	200

Pemilihan Algoritma: Train **XGBoost** and **Light GBM** with Tuned Hyperparameter

digunakan data **train** sebanyak 32% dari keseluruhan data

XGBoost

```
X_train_full, X_test, y_train_full, y_test = train_test_split(X_prepared, y,
                                                             test_size=.6,
                                                             random_state=42)

X_train, X_val, y_train, y_val = train_test_split(X_train_full,
                                                  y_train_full,
                                                  test_size=.2,
                                                  random_state=42)
```

```
X_train.shape, X_val.shape, X_test.shape
```

```
((3648597, 777), (912150, 777), (6841121, 777))
```

```
es = xgb.callback.EarlyStopping(rounds=100, metric_name='auc',
                                data_name='validation_0',
                                maximize=True,
                                save_best=True)
```

```
model_pipeline = imbpipeline([
    ('over', SMOTE(sampling_strategy=0.19676221847208702, n_jobs=-1,
                  k_neighbors=858)),
    ('under', RandomUnderSampler(sampling_strategy=0.8849275028627146)),
    ('xgb', xgb.XGBClassifier(n_estimators=1000, colsample_bytree=0.5,
                             gamma=0, learning_rate=0.05377367769556463,
                             max_depth=64, reg_alpha=1.8796306666826916,
                             reg_lambda=0.01, subsample=1.0,
                             min_child_weight=1,
                             scale_pos_weight=1/0.8849275028627146,
                             callbacks=[es], objective='binary:logistic',
                             eval_metric=['auc', 'logloss'],
                             verbosity=2, n_jobs=-1, tree_method='approx'))
])
```

```
model_pipeline.fit(X_train, y_train, xgb_eval_set=[(X_val, y_val)])
```

Light GBM

```
X_train_full, X_test, y_train_full, y_test = train_test_split(X_prepared, y,
                                                             test_size=.6,
                                                             random_state=42)

X_train, X_val, y_train, y_val = train_test_split(X_train_full,
                                                  y_train_full,
                                                  test_size=.2,
                                                  random_state=42)
```

```
X_train.shape, X_val.shape, X_test.shape
```

```
((3648597, 777), (912150, 777), (6841121, 777))
```

```
model_pipeline = imbpipeline([
    ('over', SMOTE(sampling_strategy=0.10107109574278936, k_neighbors=1000)),
    ('under', RandomUnderSampler(sampling_strategy=0.6967907665329958)),
    ('lgbm', lgbm.LGBMClassifier(n_estimators=220, boosting_type='dart',
                                colsample_bytree=0.5510238606246517,
                                learning_rate=0.13366708157335833,
                                max_depth=25, min_child_samples=200,
                                min_split_gain=0.12624055231690842,
                                num_leaves=3213, reg_alpha=0.01,
                                reg_lambda=0.01, objective='binary',
                                n_jobs=-1))
])
```

```
model_pipeline.fit(X_train, y_train, lgbm_eval_set=[(X_val, y_val)],
```

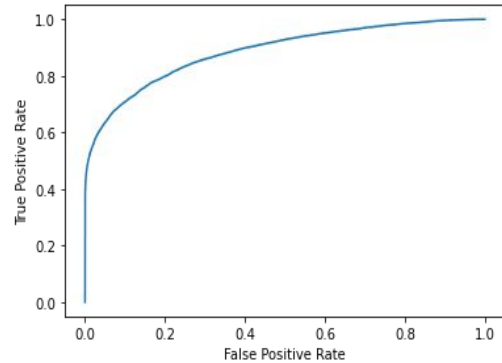
(split train, val, dan test menggunakan random_state yang sama pada kedua model, sehingga menghasilkan splitting data yang sama persis)

Pemilihan Algoritma (Optimasi Model): Threshold Tuning **XGBoost** dan **Light GBM**

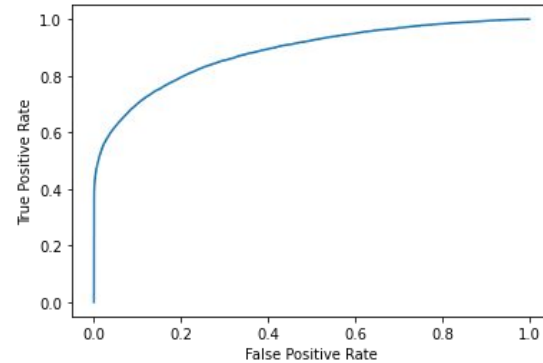
Dilakukan **prediksi probabilitas** pada **data validasi**, lalu probabilitas tersebut diplot menjadi ROC dan Precision-Recall Curve.

ROC Curve

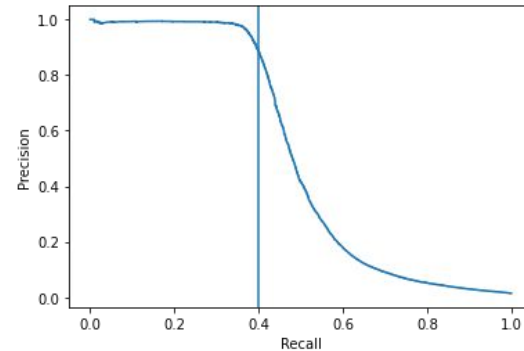
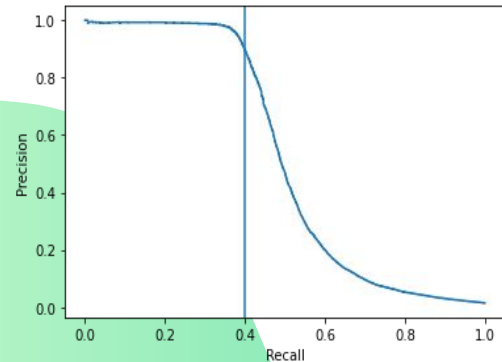
XGBoost



Light GBM



Precision-Recall
Curve



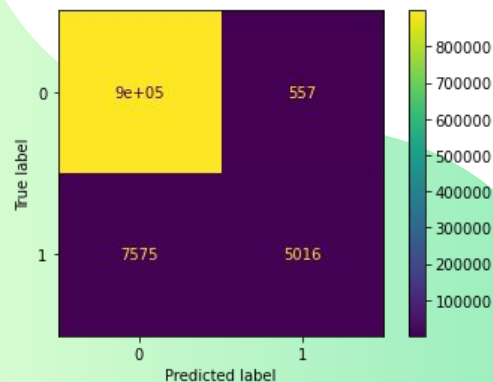
Pemilihan Algoritma: Evaluasi XGBoost dan Light GBM pada Data Validasi

`th_prec_09 = th_pr[np.argmax(precision > 0.9)]` diambil **threshold** yang menghasilkan setidaknya **90% precision** pada kedua model

XGBoost

ROC-AUC score: 0.8868023879870464

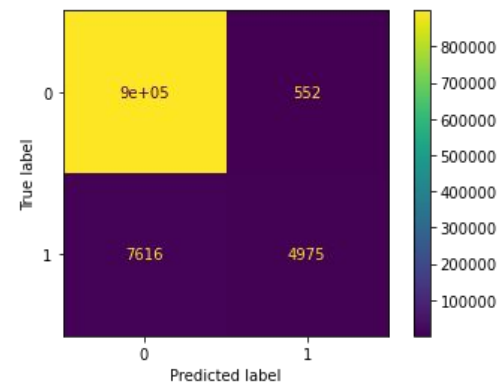
		precision	recall	f1-score	support
	0	0.99	1.00	1.00	899559
	1	0.90	0.40	0.55	12591
accuracy				0.99	912150
macro avg		0.95	0.70	0.77	912150
weighted avg		0.99	0.99	0.99	912150



Light GBM

ROC-AUC score: 0.8837498612214778

		precision	recall	f1-score	support
	0	0.99	1.00	1.00	899559
	1	0.90	0.40	0.55	12591
accuracy				0.99	912150
macro avg		0.95	0.70	0.77	912150
weighted avg		0.99	0.99	0.99	912150



Pemilihan Algoritma: Evaluasi XGBoost dan Light GBM pada Data Test

XGBoost

XGBoost Predict time on test set: 592.9001433849335 seconds

	precision	recall	f1-score	support
0	0.99	1.00	1.00	6747001
1	0.91	0.40	0.56	94120
accuracy			0.99	6841121
macro avg	0.95	0.70	0.78	6841121
weighted avg	0.99	0.99	0.99	6841121

Light GBM

Light GBM Predict time on test set: 398.96043038368225 seconds

	precision	recall	f1-score	support
0	0.99	1.00	1.00	6747001
1	0.90	0.40	0.55	94120
accuracy			0.99	6841121
macro avg	0.95	0.70	0.77	6841121
weighted avg	0.99	0.99	0.99	6841121

Re-Train XGBoost sebagai Model Final

Setelah memilih XGBoost sebagai model final, kami training kembali menggunakan **data train sebanyak 60%** dari keseluruhan data. Juga digunakan **early stopping**, jika hingga **100** tree selanjutnya skor **ROC-AUC** pada **data validasi** tidak meningkat, training akan berhenti.

```
X_train_full, X_test, y_train_full, y_test = train_test_split(X_prepared, y, test_size=.2, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_train_full, y_train_full, test_size=.25, random_state=42)
```

```
X_train.shape, X_val.shape, X_test.shape
```

```
((6841120, 777), (2280374, 777), (2280374, 777))
```

```
es = xgb.callback.EarlyStopping(rounds=100, metric_name='auc', data_name='validation_0', maximize=True,
                                save_best=True)
```

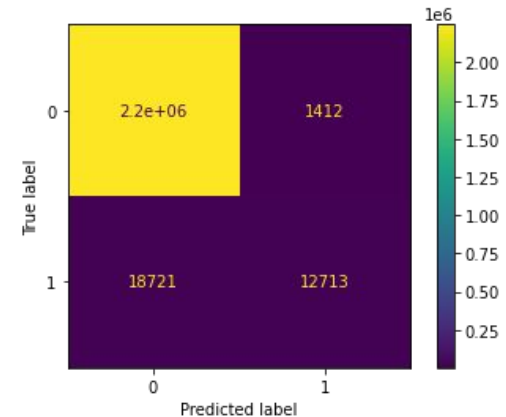
```
model_pipeline = imbpipeline([
    ('over', SMOTE(sampling_strategy=0.19676221847208702, n_jobs=-1, k_neighbors=858)),
    ('under', RandomUnderSampler(sampling_strategy=0.8849275028627146)),
    ('xgb', xgb.XGBClassifier(n_estimators=1000, colsample_bytree=0.5, gamma=0, learning_rate=0.05377367769556463,
                             max_depth=64, reg_alpha=1.8796306666826916, reg_lambda=0.01,
                             subsample=1.0, min_child_weight=1, scale_pos_weight=1/0.8849275028627146,
                             callbacks=[es], objective='binary:logistic', eval_metric=['auc', 'logloss'],
                             verbosity=2, n_jobs=-1, tree_method='approx'))
])
```

```
model_pipeline.fit(X_train, y_train, xgb__eval_set=[(X_val, y_val)])
```

Evaluasi XGBoost pada Data Validasi menggunakan Threshold yang Telah Ditentukan

`th_prec_09 = th_pr[np.argmax(precision > 0.9)]` diambil **threshold** yang menghasilkan setidaknya **90% precision** pada kedua model

	precision	recall	f1-score	support
0	0.99	1.00	1.00	2248940
1	0.90	0.40	0.56	31434
accuracy			0.99	2280374
macro avg	0.95	0.70	0.78	2280374
weighted avg	0.99	0.99	0.99	2280374



Evaluasi model pada data **test**

	precision	recall	f1-score	support
0	0.99	1.00	1.00	2248993
1	0.91	0.41	0.56	31381
accuracy			0.99	2280374
macro avg	0.95	0.70	0.78	2280374
weighted avg	0.99	0.99	0.99	2280374

Evaluasi model pada data **tahap final Healthkathon BPJS**

Nama Team	Recall	Accuracy	Precision	Specifity
The JamMaths	0.5	0.994	0.952	1

Precision

0.952

Recall

0.5

Specificity

1

Accuracy

0.994

F1-Score

0,656

Kemudahan Implementasi Model

Scikit-learn API

Terdapat syntax XGBoost yang sama seperti menggunakan scikit-learn. Sehingga mempermudah seseorang yang terbiasa dengan *interface* scikit-learn

Model XGBoost yang sudah di-*train* sebelumnya, dapat di-*train* kembali menggunakan data baru

Incremental Learning

Automatically Handle Null

XGBoost dapat secara otomatis *handle* data null, baik untuk training maupun prediksi

XGBoost dapat meng-*output* langsung kelas dari data yang diprediksi

Output 0 or 1

Saran dalam Implementasi Model

Kami telah mengembangkan dua jenis model yang berbasis algoritma Gradient Boosting, yaitu **XGBoost** dan **Light GBM**. Keduanya memiliki kelebihan dan kekurangan masing-masing. **XGBoost** lebih baik dalam hal **ketepatan memprediksi**, sedangkan **Light GBM** memiliki proses komputasi yang **lebih cepat**. Sehingga, implementasi model di *real life* dapat bergantung pada skenario yang dialami:

- Jika sumber daya komputasi yang dimiliki mencukupi untuk menggunakan model XGBoost, maka menggunakan **XGBoost** lebih disarankan karena akan menghasilkan prediksi yang lebih akurat dibanding Light GBM.
- Jika sumber daya komputasi yang dimiliki terbatas atau pengguna hanya ingin membandingkan algoritma Gradient Boosting dengan algoritma lain (misal, ANN), maka **Light GBM** lebih disarankan karena akan mengurangi waktu komputasi dibanding XGBoost.

Namun, perlu diingat bahwa untuk kasus klasifikasi inefisiensi klaim peserta BPJS Kesehatan ini berhubungan langsung dengan finansial perusahaan. Sehingga, untuk kasus ini kami lebih menyarankan untuk menggunakan **XGBoost** (yaitu, model final kami) karena akan menghasilkan prediksi yang baik walaupun mengorbankan sedikit waktu komputasi yang lebih banyak, demi finansial perusahaan.



Thank
You!