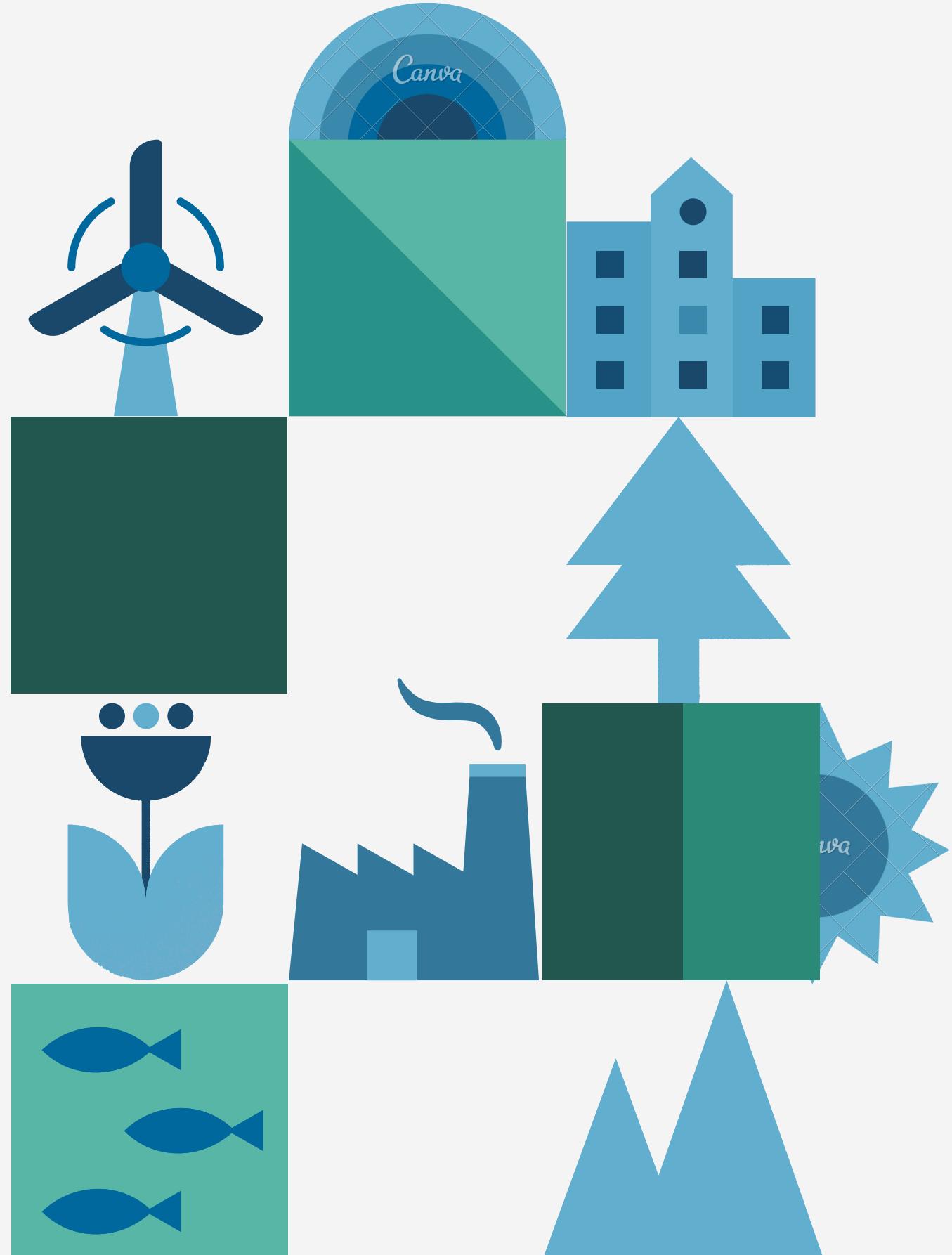




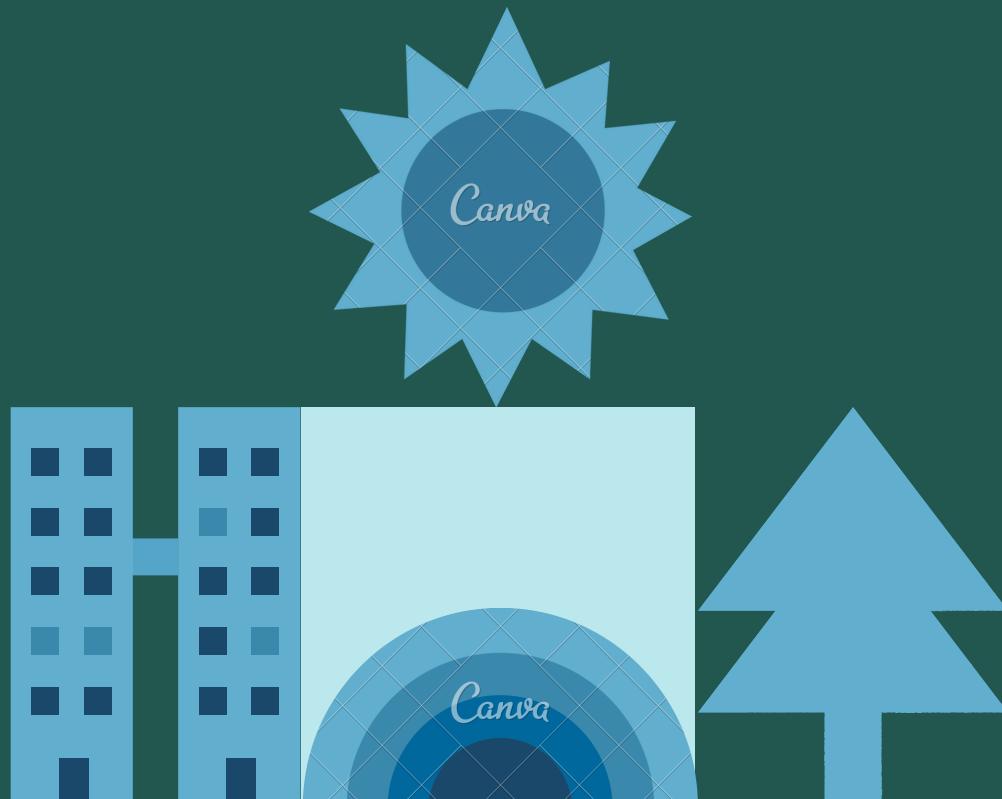
HEALTHKATHON BPJS
2022 - BISMILLAHSEHAT

BPJS CLAIM INEFFICIENCY DETECTION WITH MACHINE LEARNING

Rizky Alif Ramadhan - UGM



Contents of the Presentation



01
Introduction
and Goals

04
Exploratory
Data Analysis

02
Data
Understanding

05
Feature
Engineering &
Selection

03
Data
Preparation &
Cleansing

06
Modeling &
Evaluation



1. Introduction and Goals

Inefisiensi atau pemborosan dapat didefinisikan sebagai penggunaan lebih banyak input (atau sumber daya) daripada yang diperlukan untuk menghasilkan unit perawatan atau layanan pasien yang bermanfaat dan ini terkait dengan variasi yang tidak perlu dalam proses operasional dan klinis. (<https://www.aafp.org/pubs/fpm/issues/2015/0300/p18.html>)

Dengan mendeteksi inefisiensi pada klaim BPJS, diharapkan BPJS dapat melakukan penghematan anggaran dan mengalokasikannya pada hal-hal yang tepat secara efisien.

2. Data Understanding

No	Variable	Keterangan
1.	id	: Id kunjungan
2.	id_peserta	: Id Peserta
3.	dati2	: Lokasi Fasilitas Kesehatan (Kab/Kota)
4.	typefaskes	: Tipe Fasilitas Kesehatan
5.	usia	: Usia Peserta
6.	jenkel	: Jenis Kelamin Peserta
7.	pisat	: Hubungan Kepesertaan
8.	tgldatang	: Tanggal Kedatangan
9.	tglpulang	: Tanggal Kepulangan
10.	jenispel	: Jenis Pelayanan
11.	politujuan	: Poli Tujuan
12.	diagfktp	: Diagnosa dari FKTP
13.	biaya	: Biaya
14.	Jenispulang	: Kondisi Peserta saat Pulang
15.	cbg	: Kode Case Based Group
16.	kelasrawat	: Kelas Perawatan
17.	kdsa	: Kode Special Sub-Acute Group
18.	kdsp	: Kode Special Procedures
19.	kdsr	: Kode Special Prothesis
20.	kdsi	: Kode Special Investigations
21.	kdsd	: Kode Special Drugs
22.	label	: Label Potensi Efisiensi

Train Data

11401882

Record

22

Feature/Variabel

Label

Feature Target

Validation Data

998941

Record

21

Feature/Variabel

Label

Feature Predict

2. Data Understanding

Fitur Kategorik

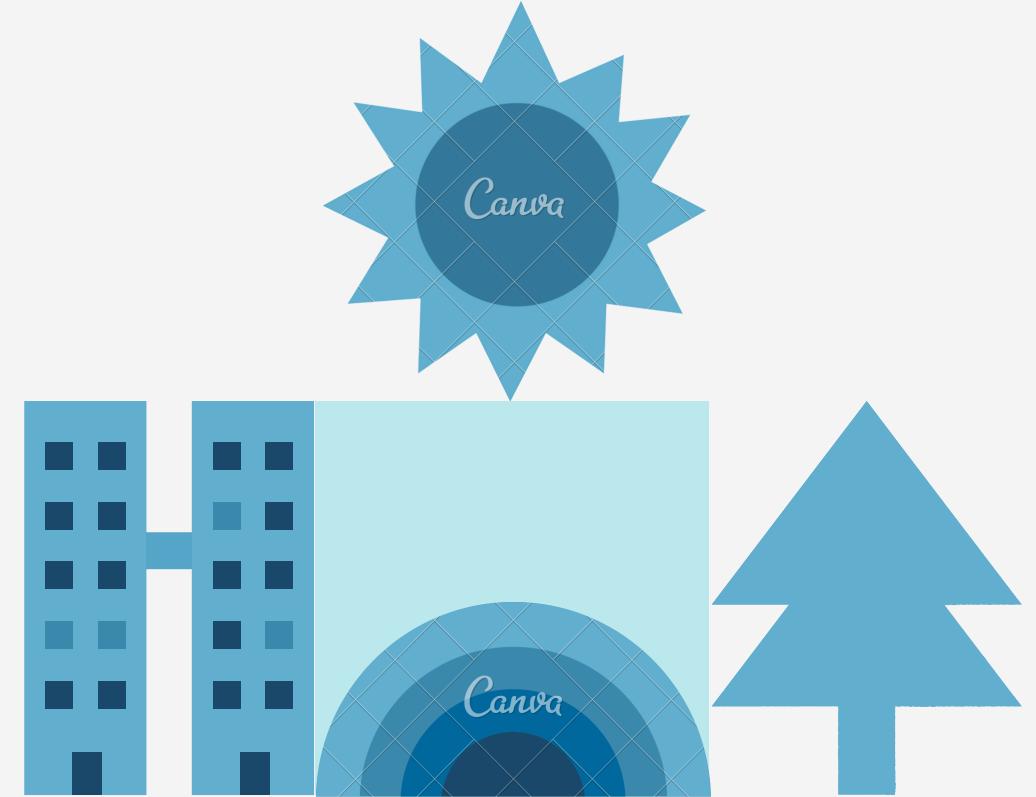
- id
- id_peserta
- dati2
- typefaskes
- jenkel
- pisat
- jenispel
- politujuan
- diagfktp
- jenispulang
- cbg
- kelasrawat
- kdsa
- kdsp
- kdsr
- kdsi
- kdsd
- label

Fitur Numerik

- usia
- biaya

Fitur Datetime

- tgldatang
- tglpulang



2. Data Understanding

Imbalance Classification

```
Nilai unik pada feature label adalah
0    11244993
1    156889
Name: label, dtype: int64
```

Klaim Berulang

```
Nilai unik pada feature id_peserta adalah
5460422    236
5676434    230
5676435    219
4490519    216
3373373    203
...
4654308    1
4654271    1
4654193    1
4654083    1
8527876    1
Name: id_peserta, Length: 8527919, dtype: int64
```

2. Data Understanding

Missing Value

Train data

	Total	Percent
politujuan	4041455	35.45
kdsa	198670	1.74
kdsi	198463	1.74
kdsr	198459	1.74
kdsp	197007	1.73
kdsd	195181	1.71
biaya	57815	0.51
diagfktp	2530	0.02
pisat	190	0.00
jenkel	49	0.00
jenispulang	39	0.00

Validation data

	Total	Percent
politujuan	108148	10.83
kdsa	20804	2.08
kdsr	20803	2.08
kdsd	20789	2.08
kdsi	20785	2.08
kdsp	20732	2.08
biaya	5049	0.51
diagfktp	64	0.01
jenkel	6	0.00
pisat	6	0.00
jenispulang	3	0.00

3. Data Preparation & Cleansing

Membuat fitur 'peserta_unik'

id_peserta pada data latih dan data validasi tidak sinkron. Padahal, id_peserta itu penting karena kita bisa melacak kebiasaan dari orang tersebut dalam mengklaim bpjs apakah sering melakukan pemborosan atau tidak.

peserta_unik = dati2 + typefaskes + usia_2020 + jenkel + pisat + kelas rawat

3. Data Preparation & Cleansing

Menghapus spasi pada 'typefaskes'



```
# Menghapus spasi setelah tipefaskes  
test['typefaskes']=test['typefaskes'].str.rstrip()
```

Menjadikan kapital pada fitur 'politujuan' dan 'diagfktp'



```
train["politujuan"]=train["politujuan"].str.upper()  
test["politujuan"]=test["politujuan"].str.upper()  
  
train["diagfktp"]=train["diagfktp"].str.upper()  
test["diagfktp"]=test["diagfktp"].str.upper()
```

3. Data Preparation & Cleansing

Membuat fitur 'lama_rawat'



```
train["lama_rawat"] = (train["tglpulang"]-train["tgldatang"]).dt.days.astype(int)
test["lama_rawat"] = (test["tglpulang"]-test["tgldatang"]).dt.days.astype(int)
```

Mengubah tipe data tanggal menjadi datetime



```
# Mengubah tipe data tanggal datang dan pulang
train["tgldatang"] = pd.to_datetime(train["tgldatang"], format='%Y-%m-%dT%H:%M:%S.%f%z')
train["tglpulang"] = pd.to_datetime(train["tglpulang"], format='%Y-%m-%dT%H:%M:%S.%f%z')
```

```
# Mengubah tipe data tanggal datang dan pulang
test["tgldatang"] = pd.to_datetime(test["tgldatang"], format='%Y-%m-%dT%H:%M:%S.%f%z')
test["tglpulang"] = pd.to_datetime(test["tglpulang"], format='%Y-%m-%dT%H:%M:%S.%f%z')
```

3. Data Preparation & Cleansing

Mengisi missing value



```
# Mengisi nilai NaN pada biaya dengan 0
train["biaya"] = train["biaya"].fillna(0)
# Mengisi nilai NaN pada biaya dengan 0
test["biaya"] = test["biaya"].fillna(0)

# Mengisi nilai NaN pada pisat dengan nilai modus yaitu 1.0
train["pisat"] = train["pisat"].fillna(1.0)
# Mengisi nilai NaN pada pisat dengan nilai modus yaitu 1.0
test["pisat"] = test["pisat"].fillna(1.0)

# Mengisi nilai NaN pada jenkel dengan nilai modus yaitu P
train["jenkel"] = train["jenkel"].fillna("P")
# Mengisi nilai NaN pada jenkel dengan nilai modus yaitu P
test["jenkel"] = test["jenkel"].fillna("P")

# Mengisi nilai NaN pada jenispulang dengan nilai modus yaitu 1
train["jenispulang"] = train["jenispulang"].fillna(1)
# Mengisi nilai NaN pada jenkel dengan nilai modus yaitu 1
test["jenispulang"] = test["jenispulang"].fillna(1)
```



```
# Mengisi nilai NaN pada kolom lain dengan none
train = train.fillna("None")
# Mengisi nilai NaN pada kolom lain dengan none
test = test.fillna("None")
```

4.Exploratory Data Analysis

Label terhadap jenis kelamin

label	0	1
jenkel		
L	0.985784	0.014216
P	0.986628	0.013372

Label terhadap jenpel

label	0	1
jenispel		
2	0.985963	0.014037
1	0.986745	0.013255

Label terhadap dati2

label	0	1
dati2		
528	0.742424	0.257576
483	0.868594	0.131406
364	0.876754	0.123246
466	0.891204	0.108796
245	0.898135	0.101865

Label terhadap kelas rawat

label	0	1
kelasrawat		
2	0.983389	0.016611
1	0.985824	0.014176
3	0.986542	0.013458

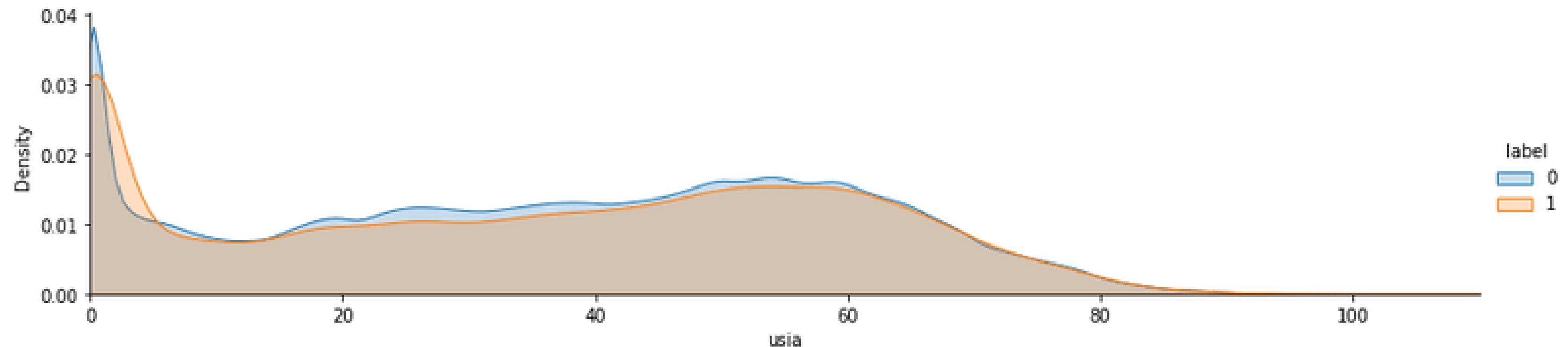
Label terhadap pisat

label	0	1
pisat		
5.0	0.984199	0.015801
4.0	0.984649	0.015351
2.0	0.986711	0.013289
1.0	0.986957	0.013043
3.0	0.987315	0.012685

Label terhadap politujuan

label	0	1
politujuan		
URL	0.000000	1.000000
BAN	0.600000	0.400000
DIG	0.750000	0.250000
ANT	0.835341	0.164659
146	0.875000	0.125000

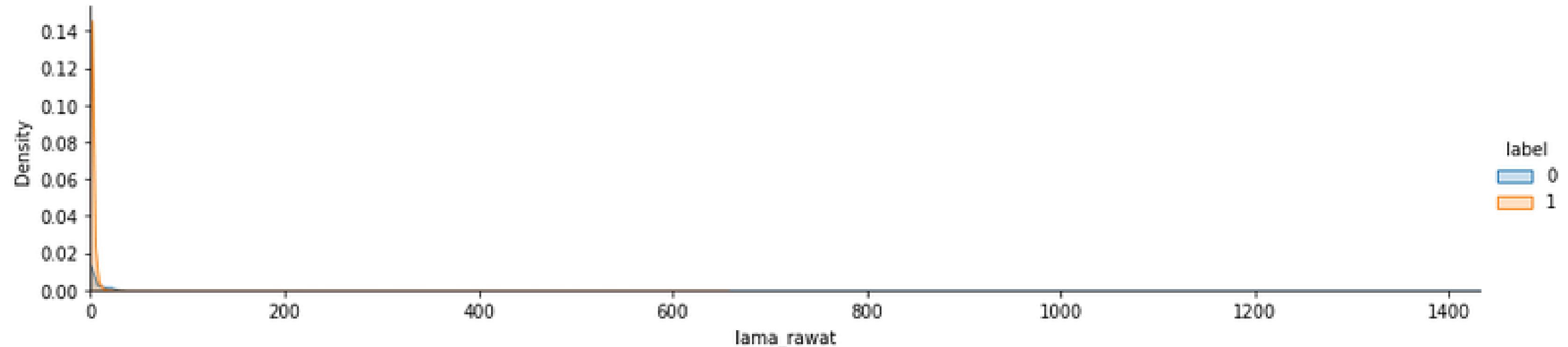
4. Exploratory Data Analysis



Usia 2-7 Tahun

Banyak yang dikategorikan
sebagai Inefficiency

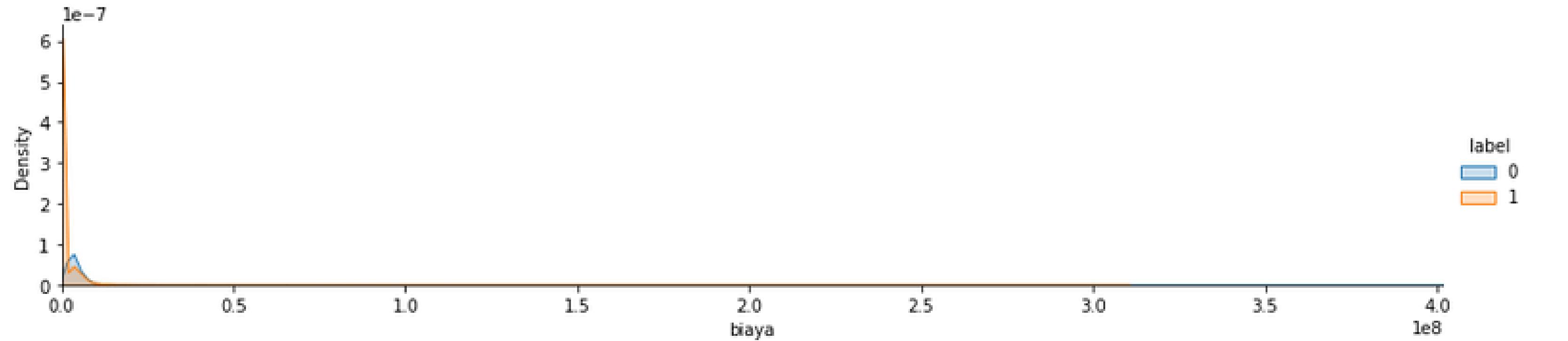
4. Exploratory Data Analysis



Lama Rawat 0-25 Hari

Banyak yang dikategorikan
sebagai Inefficiency

4. Exploratory Data Analysis



95.2%
Biaya 0 adalah
Inefficiency

Biaya $\geq 12,5$ Juta
Banyak yang dikategorikan
sebagai Inefficiency

5. Feature Engineering & Selection

Menambahkan fitur 'probability_inefficiency'

probability_inefficiency merupakan kemungkinan pasien tersebut melakukan inefficiency kembali. Misalnya pasien dengan peserta_unik 137SC60L1.03 melakukan 3 kali klaim, 1 dari 3 kali klaim tersebut terdeteksi inefficiency, maka probability_inefficiencynya bernilai 0.3333

probability_inefficiency = jumlah melakukan inefficiency/jumlah total klaim

5. Feature Engineering & Selection

Menambahkan fitur 'biaya_per_hari'



```
train["biaya_per_hari"] = train["biaya"]/(train["lama_rawat"]+1)  
test["biaya_per_hari"] = test["biaya"]/(test["lama_rawat"]+1)
```

Menambahkan fitur 'is_biaya_0'



```
# Berdasarkan EDA biaya 0 cenderung melakukan fraud  
  
a = np.array(train['biaya'].values.tolist())  
b = np.array(test['biaya'].values.tolist())  
  
train["is_biaya_0"] = np.where(a == 0, 1, 0).tolist()  
test["is_biaya_0"] = np.where(b == 0, 1, 0).tolist()
```

5. Feature Engineering & Selection

LabelEncoder Fltur Kategorik



```
cat_=['typefaskes','jenkel', 'politujuan', 'diagfktp',
      'cbg', 'kdsa', 'kdsp', 'kdsr', 'kdsi','kdsd']

for x in cat_ :
    from sklearn.preprocessing import LabelEncoder
    labelencoder = LabelEncoder()
    data[x] = labelencoder.fit_transform(data[x])
```

5. Feature Engineering & Selection

Melihat korelasi terhadap Label

	label
label	1.000000
is_bilaya_0	0.597825
probability_inefficiency	0.371094
biaya_per_hari	0.015216
biaya	0.013566
usia	0.011001

Fitur yang dibuat pada Feature engineering berhasil menempati 3 nilai tertinggi pada korelasi terhadap label.

5. Feature Engineering & Selection

Uji ANOVA pada fitur numerik

- H₀ : variable x tidak memiliki hubungan dengan variabel y (label)
- H₁ : variable x memiliki hubungan dengan variabel y (label)

	Variabel	f_score	p_value
0	probability_inefficiency	1.820927e+06	0.000000e+00
4	biaya_per_hari	2.640556e+03	0.000000e+00
1	biaya	2.098759e+03	0.000000e+00
3	usia	1.380077e+03	4.676804e-302
2	lama_rawat	6.972652e+02	1.188615e-153

Karena p-value < 0.05 semua variabel numerik dipakai dalam pemodelan

5. Feature Engineering & Selection

Uji Chi-Squared pada fitur kategorik

- H₀ : variable x tidak memiliki hubungan dengan variabel y (label)
- H₁ : variable x memiliki hubungan dengan variabel y (label)

	Variabel	chi_score	p_value
15	is_bisaya_0	4.052647e+06	0.000000e+00
8	diagfktp	1.277221e+06	0.000000e+00
9	cbg	9.129081e+04	0.000000e+00
0	dati2	3.627014e+04	0.000000e+00
5	typedefaskes	1.162680e+04	0.000000e+00
7	politujuan	5.767348e+03	0.000000e+00
3	pisat	5.300037e+02	2.819369e-117
10	kdsae	2.802563e+02	6.603144e-63
12	kdsr	1.600653e+02	1.094942e-36
6	jenkel	6.841833e+01	1.322425e-16
11	kdsp	4.092171e+01	1.584514e-10
1	kelasrawat	2.460484e+01	7.037494e-07
4	jenispel	1.633315e+01	5.312639e-05
2	jenispulang	1.582353e+01	6.953244e-05
13	kdsd	4.071245e+00	4.361912e-02
14	kdsi	6.359789e-04	9.798806e-01

Karena p-value < 0.05 semua variabel kategorik dipakai dalam pemodelan kecuali kdsi (p-value = 0.9 > 0.05)

5. Feature Engineering & Selection

Catatan

- Oversampling dan undersampling tidak dilakukan untuk menangani imbalance dataset karena pada tahap pemodelan akan menggunakan model ensemble (XGBoost dan CatBoost) dimana pada model tersebut terdapat parameter `scale_pos_weight` yang dapat mengatur bobot dari setiap label

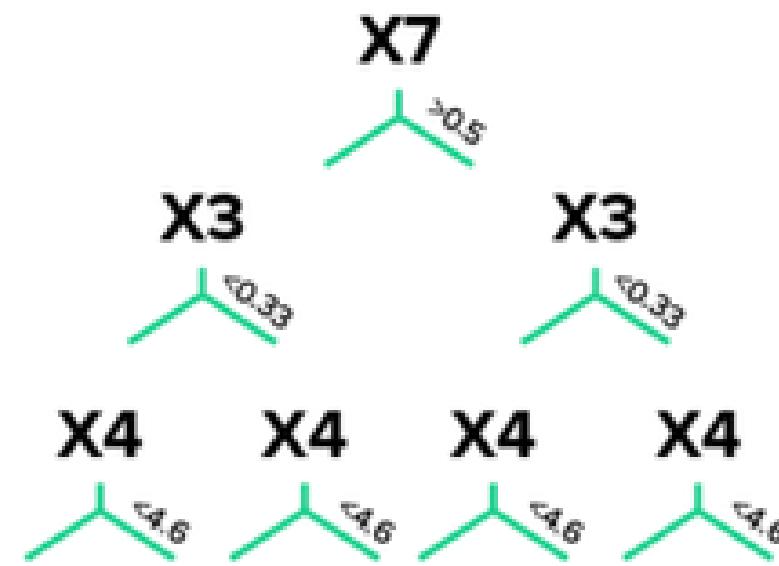
**Kehilangan Banyak
Informasi Kategorik**

**Beban Komputasi Tinggi
dan Overfitting**

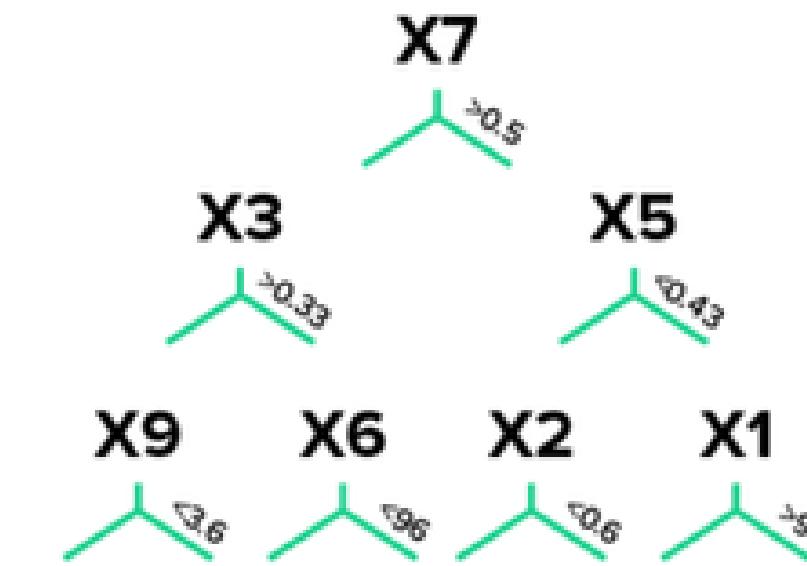
- MinMaxScaling tidak dilakukan karena pada tree-based model kita tidak memerlukan scaling

6. Modeling & Evaluation

Tree growth examples:



CatBoost



XGBoost

6. Modeling & Evaluation



Yandex
CatBoost

- Gradient Boosting
- Support GPU
- Imbalance Handler
- No MinMaxScaler needed
- Best for categorical feature (CatBoost)
- Can be used for Real-Time Prediction

6. Modeling & Evaluation

dmlc
XGBoost

+
**Halving
RandomizedSearch**

Test Data

0.99

Precision

0.89

Recall

0.999+

Accuracy

0.999+

Specificity

0.967

F1_Macro CV

Validation Data

0.002

Precision

0.013

Recall

0.988

Accuracy

0.999

Specificity

3.03 ms

Per Row Real-time
Prediction



```
param = {'tree_method': 'gpu_hist',
         'subsample': 0.7,
         'n_estimators': 700,
         'min_child_weight': 7,
         'max_depth': 10,
         'learning_rate': 0.001,
         'colsample_bytree': 0.7}
```

6. Modeling & Evaluation



Yandex
CatBoost

+

Bayesian
Optimization

Test Data

0.66

Precision

0.67

Recall

0.999+

Accuracy

0.999+

Specificity

0.828

F1_Macro CV

Validation Data

0.006

Precision

0.009

Recall

0.982

Accuracy

0.993

Specificity

2.88 ms

Per Row Real-time
Prediction

```
best_params = {'bagging_temperature': 0.36426463210895726,  
               'border_count': 206,  
               'depth': 8,  
               'iterations': 342,  
               'l2_leaf_reg': 5,  
               'random_strength': 3.3149120967201682e-06,  
               'scale_pos_weight': 4.210304871687331}
```

Penerapan Pada Real-time Prediction

- Menyimpan model dalam bentuk .pkl atau .cbm
- Membuat API
- Deploy API ke WebService
- Melakukan praproses data
- Membuat kolom/nilai baru : lama_rawat, biaya_per_hari, is_biaya_0
- Melakukan mapping pada probability_inefficiency. Jika terdapat peserta baru maka nilainya bisa 0 atau rata-rata dari probability_inefficiency
- Melakukan mapping pada fitur-fitur kategorik untuk data yang akan diprediksi sesuai labelencoder
- Melakukan prediksi real-time (row by row)

Kesimpulan

- XGBoost+HalvingSearchCV dan CatBoost+BayesianOptimization cukup baik dalam pemodelan inefficiency pada klaim bpjs berdasarkan skor pada data test
- XGBoost+HalvingSearchCV dan CatBoost+BayesianOptimization layak digunakan sebagai model real-time karena untuk prediksi setiap baris hanya membutuhkan waktu ≤ 3 ms
- Skor pada leaderboard kecil bisa jadi disebabkan karena distribusi antara data train dan data validation yang tidak sama, serta bisa jadi id dengan label antara prediksi dan aktual tidak cocok



The background features a light green gradient with three overlapping circles: a large light green circle at the top left, a medium light green circle below it, and a smaller grey circle to the right. A thick blue diagonal swoosh starts from the top right corner and curves down towards the bottom right.

TERIMA KASIH