BPJS KESEHATAN Cost Prediction

'ODI RAMADHANI ALFARIZ
yodialfa.github.io

BUSINESS PROBLEM

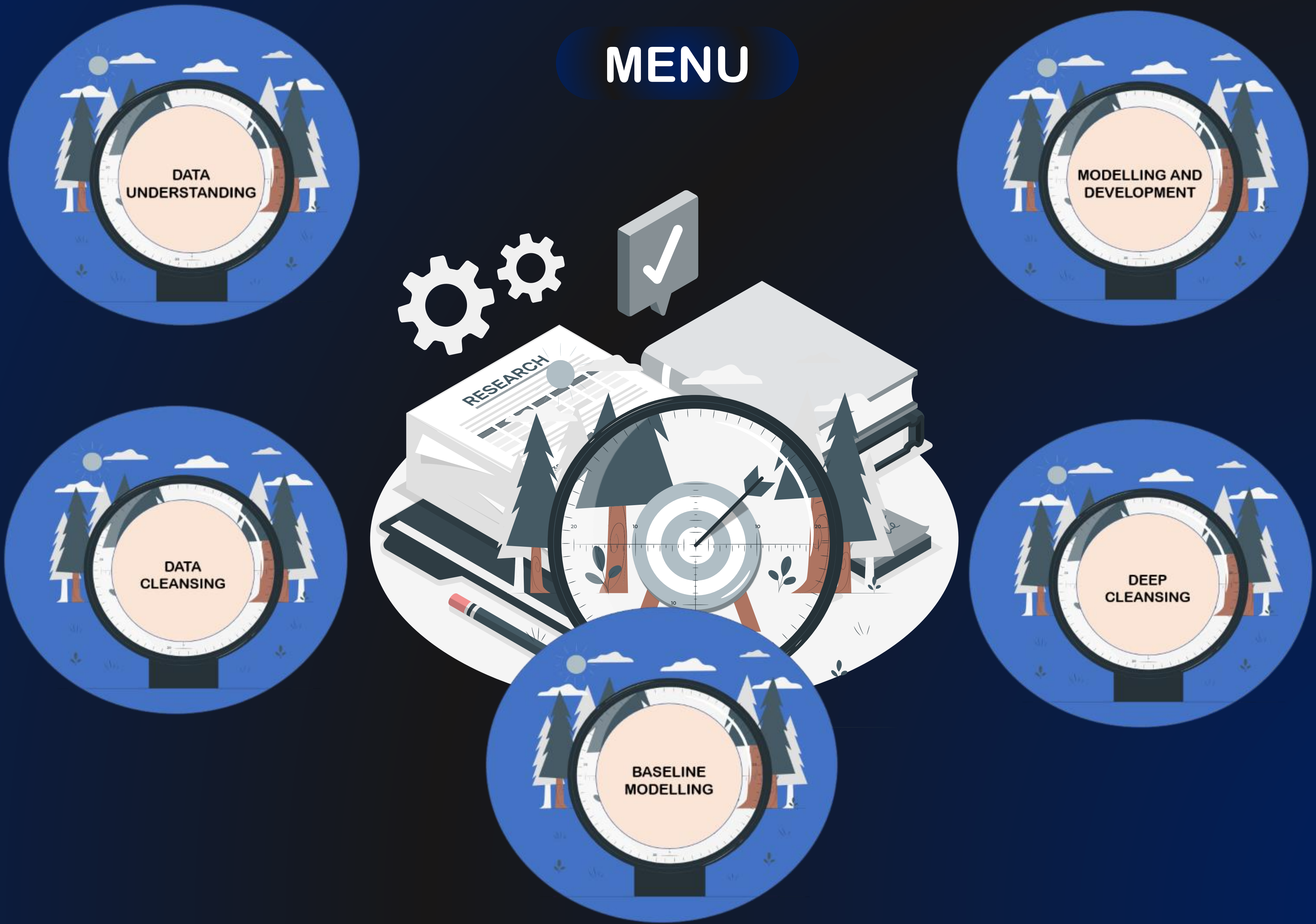
We have historical data from BPJS Kesehatan which include ‘unit_cost’ for payment. And that data useful for prediction.

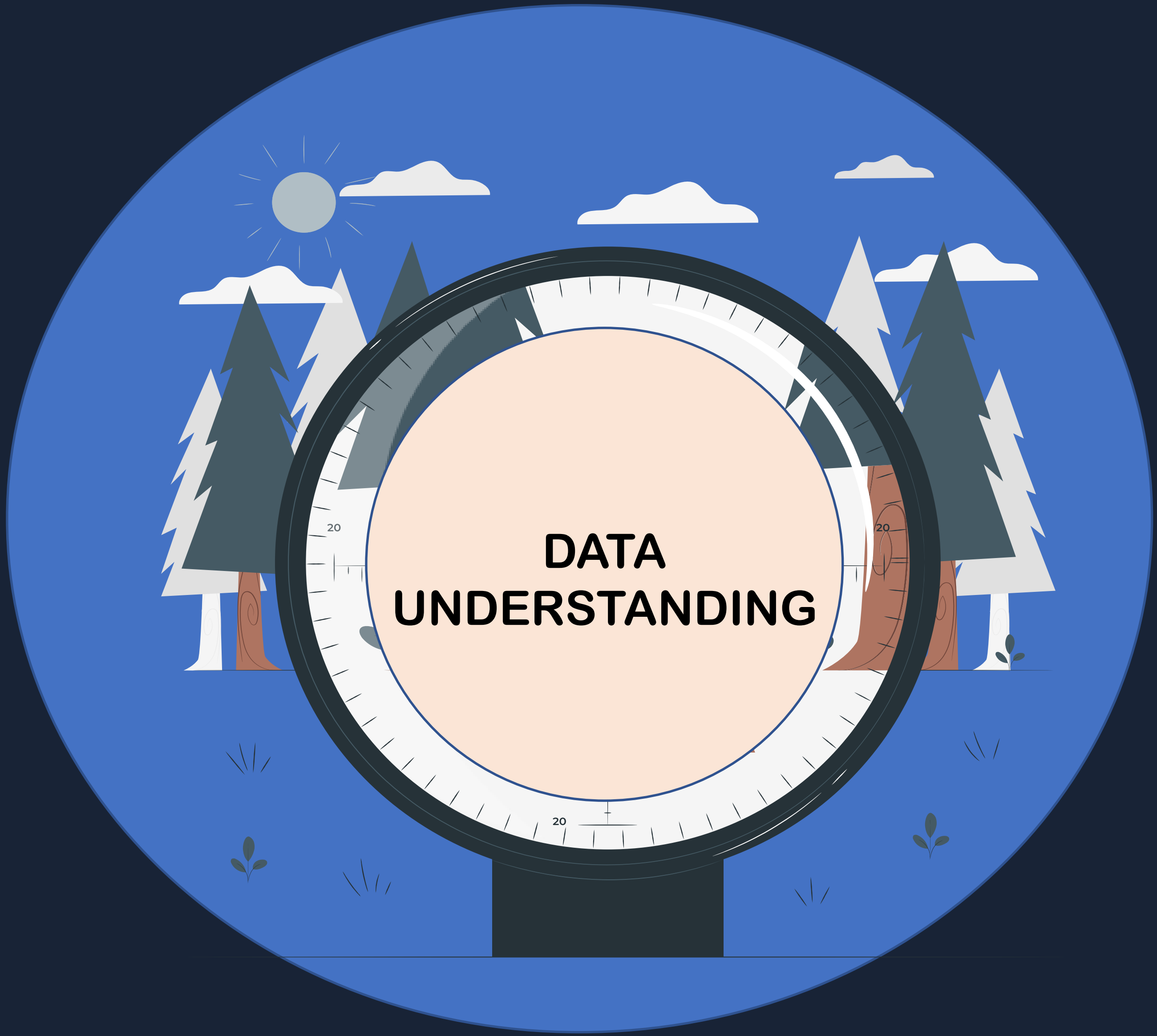
OBJECTIVE

To predict Cost Prediction for BPJS Kesehatan

MENU

MENU


DATA UNDERSTANDING
MODELLING AND DEVELOPMENT
RESEARCH
DATA CLEANSING
DEEP CLEANSING
BASELINE MODELLING


DATA
UNDERSTANDING
20
20
20

DATA UNDERSTANDING

36 Features

57.971 Rows

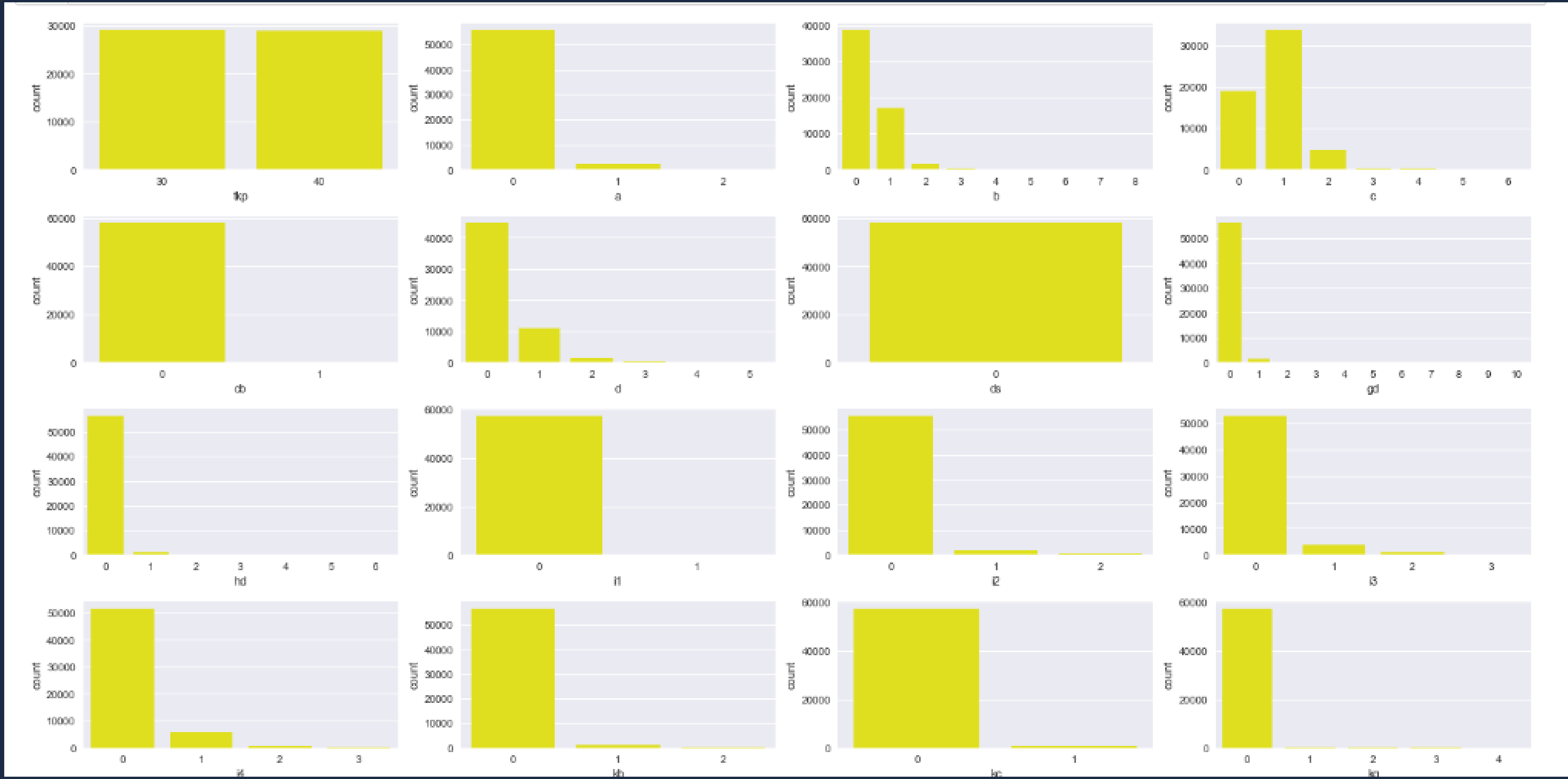
Some Features has been encoded by Author, So I try to Explore categorical data. And the target is unit_cost

DATA UNDERSTANDING

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 57971 entries, 0 to 57970
Data columns (total 36 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   row_id        57971 non-null  int64
 1   tglpelayanan  57971 non-null  object
 2   kddati2       57971 non-null  int64
 3   tkp           57971 non-null  int64
 4   peserta       57971 non-null  int64
 5   a             57971 non-null  int64
 6   b             57971 non-null  int64
 7   c             57971 non-null  int64
 8   cb            57971 non-null  int64
 9   d             57971 non-null  int64
 10  ds            57971 non-null  int64
 11  gd            57971 non-null  int64
 12  hd            57971 non-null  int64
 13  i1            57971 non-null  int64
 14  i2            57971 non-null  int64
 15  i3            57971 non-null  int64
 16  i4            57971 non-null  int64
 17  kb            57971 non-null  int64
```

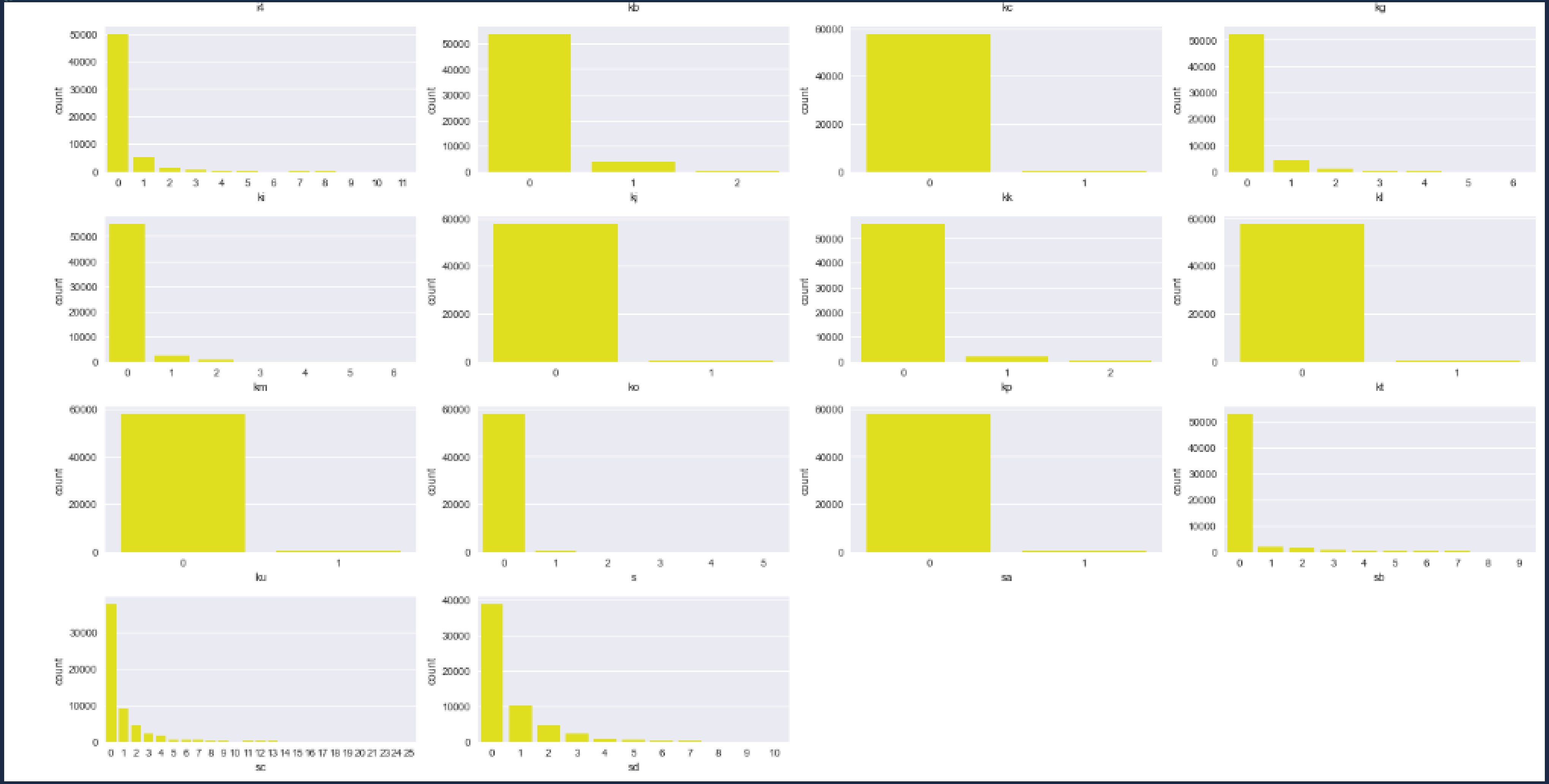
```
 17  kb              57971 non-null  int64
 18  kc              57971 non-null  int64
 19  kg              57971 non-null  int64
 20  ki              57971 non-null  int64
 21  kj              57971 non-null  int64
 22  kk              57971 non-null  int64
 23  kl              57971 non-null  int64
 24  km              57971 non-null  int64
 25  ko              57971 non-null  int64
 26  kp              57971 non-null  int64
 27  kt              57971 non-null  int64
 28  ku              57971 non-null  int64
 29  s               57971 non-null  int64
 30  sa              57971 non-null  int64
 31  sb              57971 non-null  int64
 32  sc              57971 non-null  int64
 33  sd              57971 non-null  int64
 34  case            57971 non-null  int64
 35  unit_cost       57971 non-null  float64
dtypes: float64(1), int64(34), object(1)
memory usage: 15.9+ MB
```

DATA UNDERSTANDING



Count From Categorical Data (1)

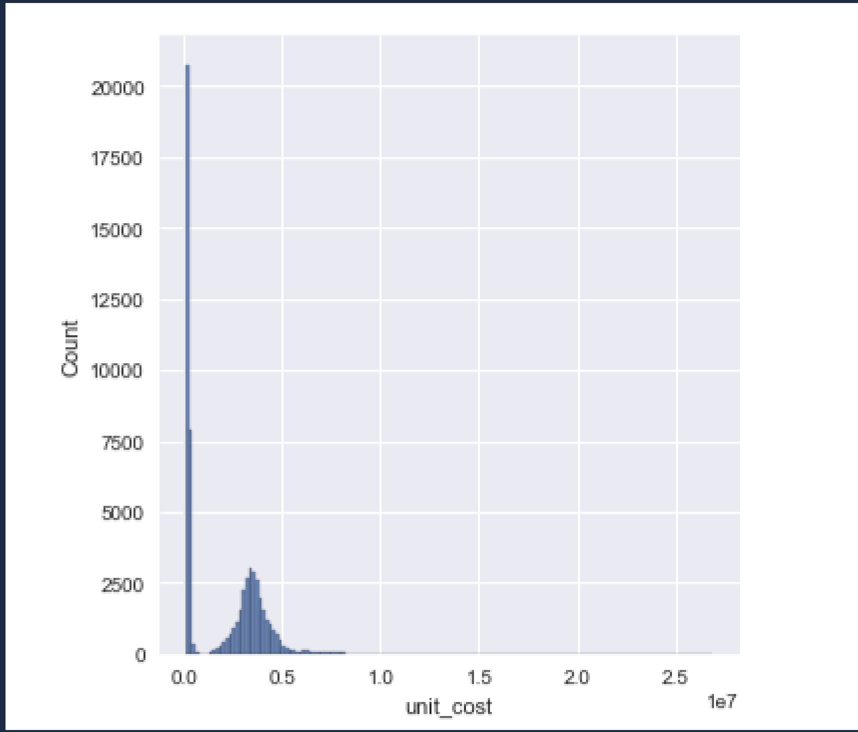
DATA UNDERSTANDING



Count From Categorical Data (2)

DATA UNDERSTANDING

We can see ‘unit_cost’ isn’t normal distribution



Data distribution from unit_cost

```
case
84        62
69        61
44        60
221       60
73        59
          ..
5035       1
8382       1
18169      1
108216     1
16679      1
Name: case, Length: 15332, dtype: int64
```

case

And from categorical data we can see ‘ds’ is has 1 value. The majority of features is categorical data. And numerical data is ‘case’ and ‘unit_cost’.
‘peserta’ is unique value from id member,


DATA
CLEANSING

DATA CLEANSING

Drop Missing Value

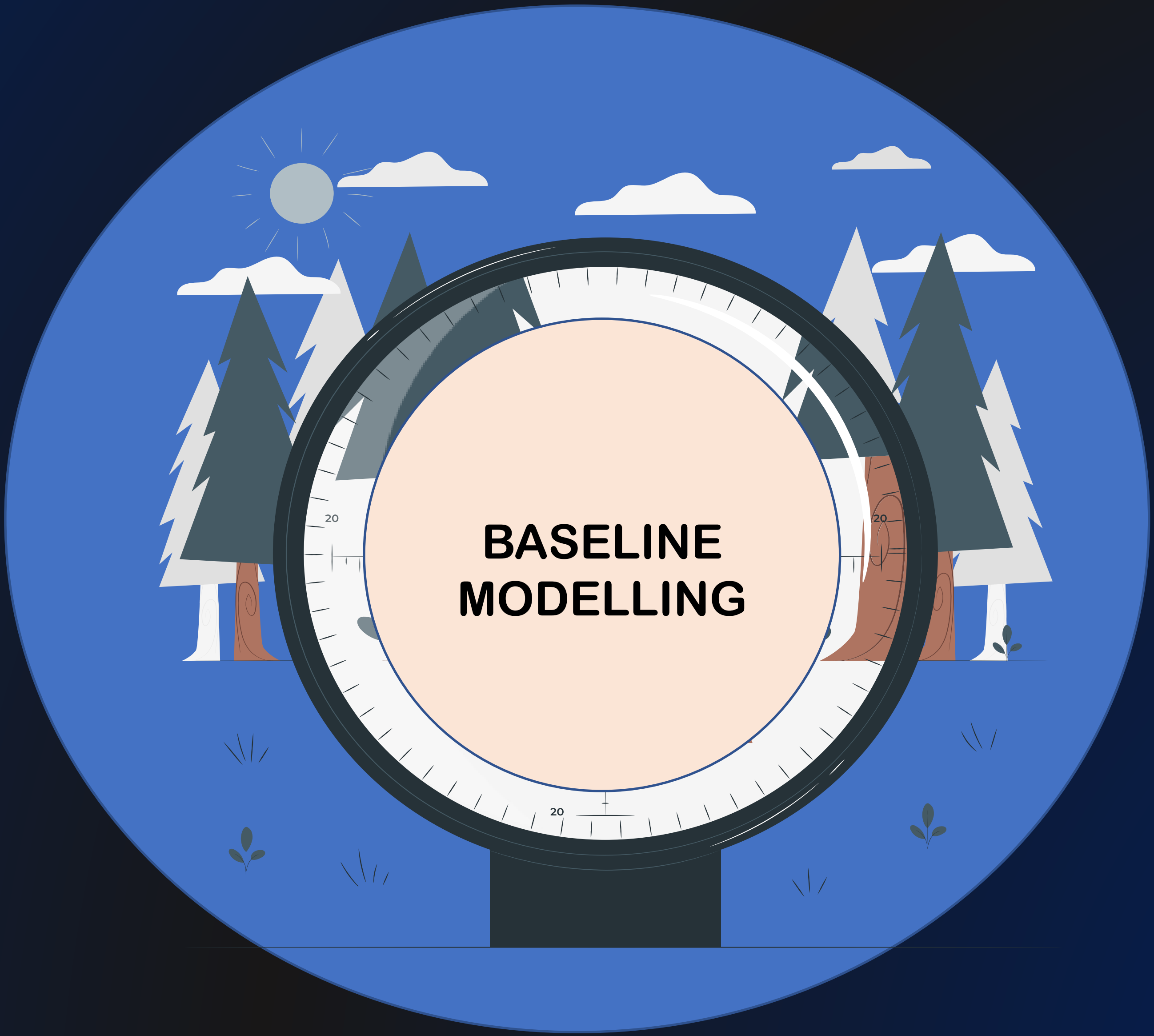
0 Missing Value

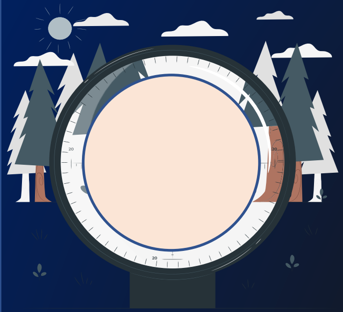
Drop Duplicated

0 Duplicated Data

I try to find missing values, and duplicated data. But I didn’t found there. So I’m going through to next step

BASELINE MODELLING





BASELINE MODELLING

Multiple Linear Regression

```
R2 score is  0.8962316610646258

Mean Absolute Error is 386089.12

Mean Squared Error is 368829594049.93

Root Mean Squared Error is 607313.42

Accuracy of Multiple Linear Regression is  89.62 %
```

Ridge Regression

```
R2 score is  0.8962171968261347

Mean Absolute Error is 386041.17

Mean Squared Error is 368881005099.45

Root Mean Squared Error is 607355.75

Accuracy of Ridge Regression is  89.62 %
```

Lasso Regression

```
R2 score is  0.896230875360505

Mean Absolute Error is 386088.21

Mean Squared Error is 368832386721.91

Root Mean Squared Error is 607315.72

Accuracy of Lasso Regressor is  89.62 %
```

Decision Tree

```
R2 score is  0.9791800908951597

Mean Absolute Error is 121199.76

Mean Squared Error is 74001364019.88

Root Mean Squared Error is 272031.92

Accuracy of Decission Tree Regressor is  97.92 %
```

Random Forest

```
R2 score is  0.9852127642586729

Mean Absolute Error is 104803.12

Mean Squared Error is 52559096652.7

Root Mean Squared Error is 229257.71

Accuracy of Random Forest Regressor is  98.52 %
```

Light GBM

```
R2 score is  0.9762737061696386

Mean Absolute Error is 160387.71

Mean Squared Error is 80095509498.89

Root Mean Squared Error is 283011.5

Accuracy of Random Forest Regressor is  97.63 %
```

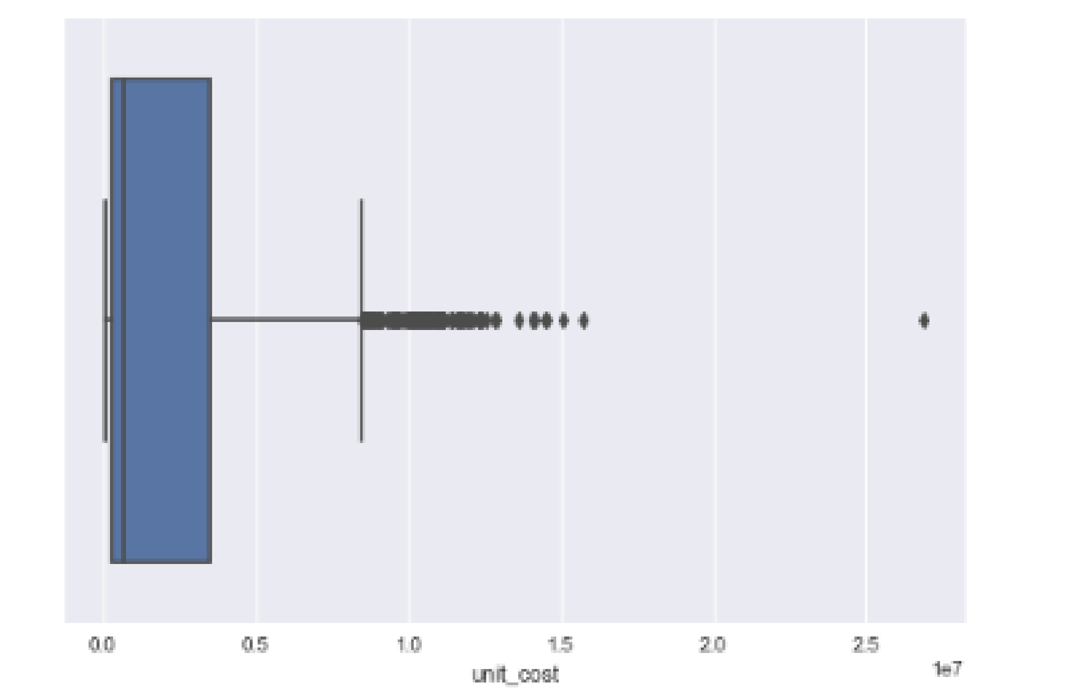
DT, RF and LGBM is high R2 Score and Slowest RMSE than Linear Regression, Ridge and Lasso


DEEP
CLEANSING

DEEP CLEANSING

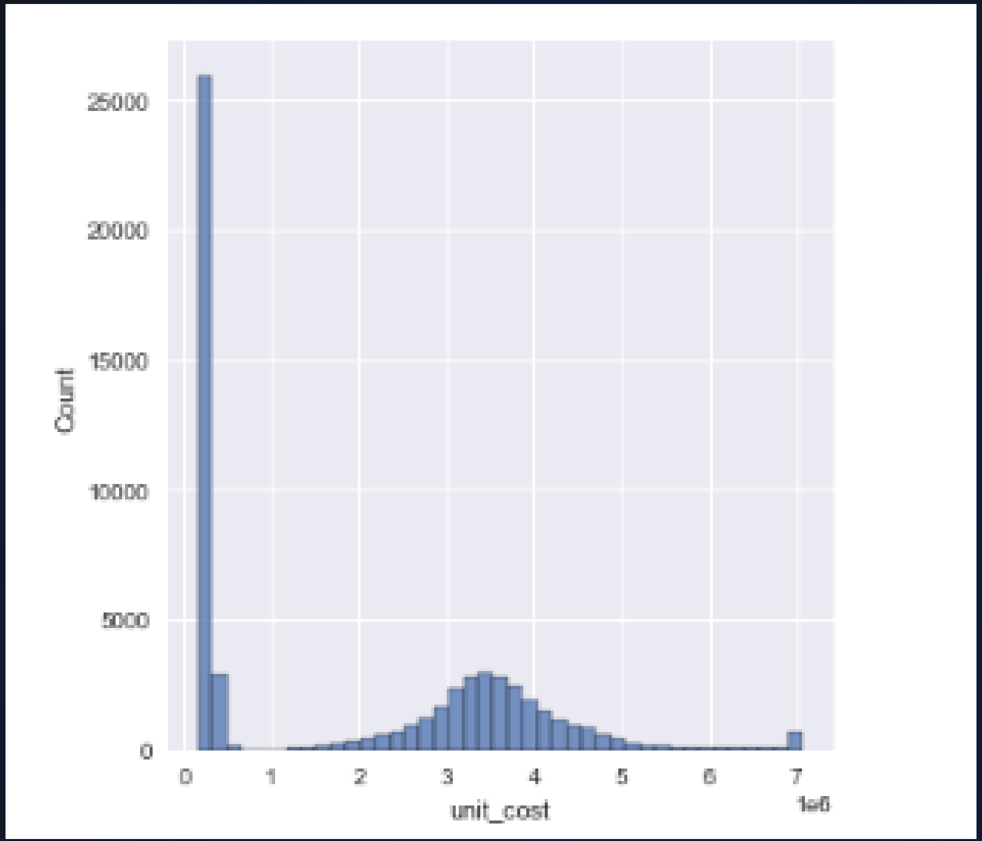
We will drop 'ds' because has 1 value. And 'peserta' because we don't need id member here.

And the target is has an outlier. We will remove the outlier



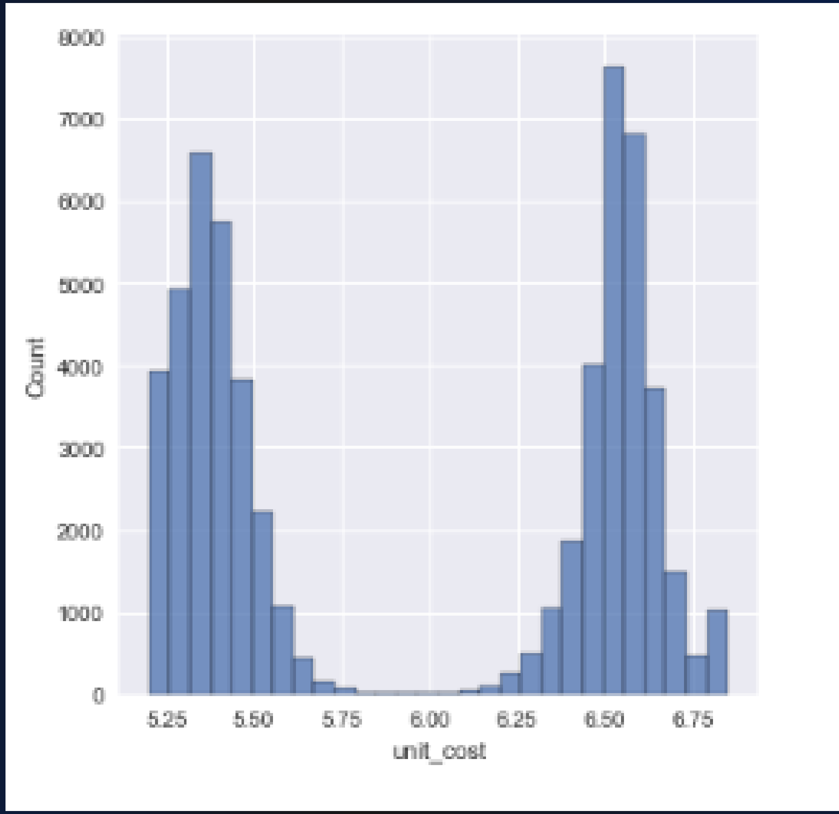
'data in unit_cost' didn't normal

DEEP CLEANSING



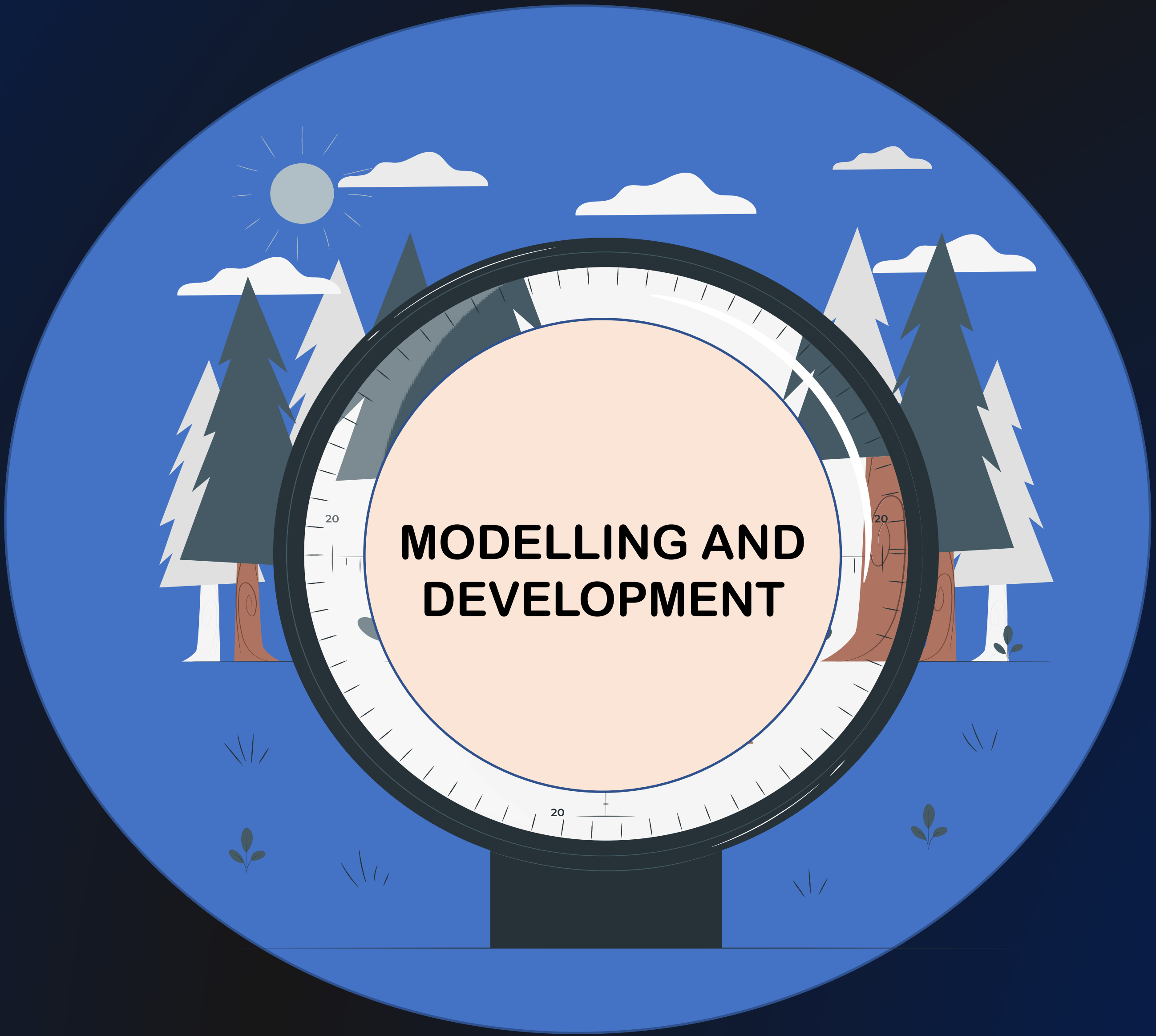
After we remove outlier in unit_cost, we still look the data isn’t normal. And we have to log transform to normalize the data

DEEP CLEANSING



And finaly we have normalized data, but we can see the data is **bimodal (two peak).** But we will fitting into Machine learning which immune that bimodal data. We will try every Machine Learning to see the performance of ML to our data.

MODELLING AND DEVELOPMENT



MODELLING
AND DEVELOPMENT

Before Tunning

Decision Tree

R2 score is 0.9906075173704108

Mean Absolute Error is 0.03444521880387015

Mean Squared Error is 0.003318880796118906

Root Mean Squared Error is 0.057609728311448456

Random Forest

R2 score is 0.9906727815184282

Mean Absolute Error is 0.03442042421812075

Mean Squared Error is 0.0032958193824253607

Root Mean Squared Error is 0.05740922732823845

RANSAC

R2 score is 0.9764543802534339

Mean Absolute Error is 0.0656579084409205

Mean Squared Error is 0.008319962707560892

Root Mean Squared Error is 0.0912138295849971

After Tunning Hyperparameter

R2 score is 0.9735717429383282

Mean Absolute Error is 0.06942558715650762

Mean Squared Error is 0.00933855704566917

Root Mean Squared Error is 0.09663620980599959

R2 score is 0.9888619361428885

Mean Absolute Error is 0.043819094029227824

Mean Squared Error is 0.004324302254119555

Root Mean Squared Error is 0.06575942711216054

R2 score is 0.9765328035752863

Mean Absolute Error is 0.06559954989908592

Mean Squared Error is 0.008292251433861666

Root Mean Squared Error is 0.0910618000802843

DT and RF RMSE seem that Overfitting after HyperParameter Tunning. And RANSAC seem has improvement after tunning, but the performance isn't significant.

MODELLING AND DEVELOPMENT

Before Tunning

Light GBM

```
R2 score is  0.986827515587607

Mean Absolute Error is 0.04537071999791156

Mean Squared Error is 0.004654563364934165

Root Mean Squared Error is 0.06822436049487136
```

XGB Regressor

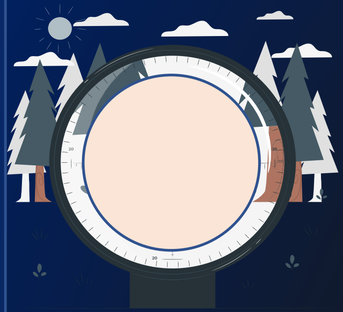
```
R2 score is  0.9897877398415768

Mean Absolute Error is 0.038185884212807074

Mean Squared Error is 0.0036085532932460915

Root Mean Squared Error is 0.060071235156654565
```

After Tunning Hyperparameter

```
R2 score is  0.9896600433245044

Mean Absolute Error is 0.03627908714347487

Mean Squared Error is 0.003653675497348689

Root Mean Squared Error is 0.060445640846538216
```

```
R2 score is  0.9908500000529297

Mean Absolute Error is 0.034357566088943414

Mean Squared Error is 0.003233198325345063

Root Mean Squared Error is 0.0568612198721155
```

LGBM and XGB show improvement after tunning. But XGB is slowest RMSE here. We will choose XGB to Final Model



MODELLING AND DEVELOPMENT

Invers Target into Normal And Fitting Using Best Parameter

In [211]:
```
final_df['unit_cost'] = 10 ** final_df['unit_cost']
```

In [213]:
```
#split feature and target data
X = final_df.drop('unit_cost', axis=1)
y = final_df['unit_cost']

#define var of split result
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)


best_n_estimator = rc_xgb.best_params_['n_estimators']
best_min_child_weight = rc_xgb.best_params_['min_child_weight']
best_max_depth = rc_xgb.best_params_['max_depth']
best_learning_rate = rc_xgb.best_params_['learning_rate']
best_booster = rc_xgb.best_params_['booster']

xgb_best = XGBRegressor(n_estimators = best_n_estimator,
                        min_child_weight = best_min_child_weight,
                        max_depth = best_max_depth,
                        learning_rate = best_learning_rate,
                        booster = best_booster)

xgb_best.fit(X_train, y_train)
```

Out[213]:

XGBRegressor

```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.05, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=10, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=1100,
             n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
             reg_alpha=0, reg_lambda=1, ...)
```

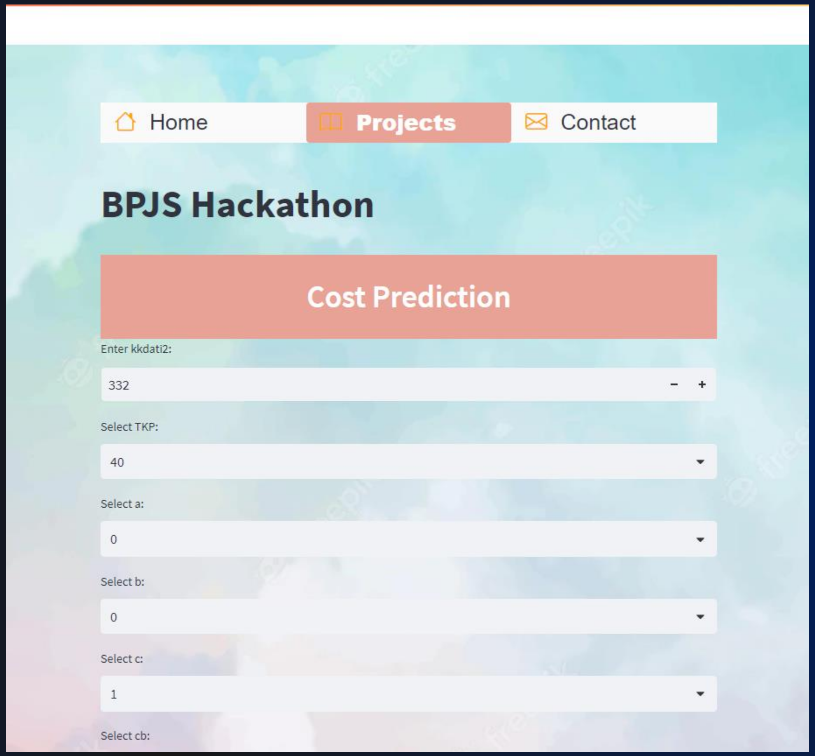
MODELLING
AND DEVELOPMENT

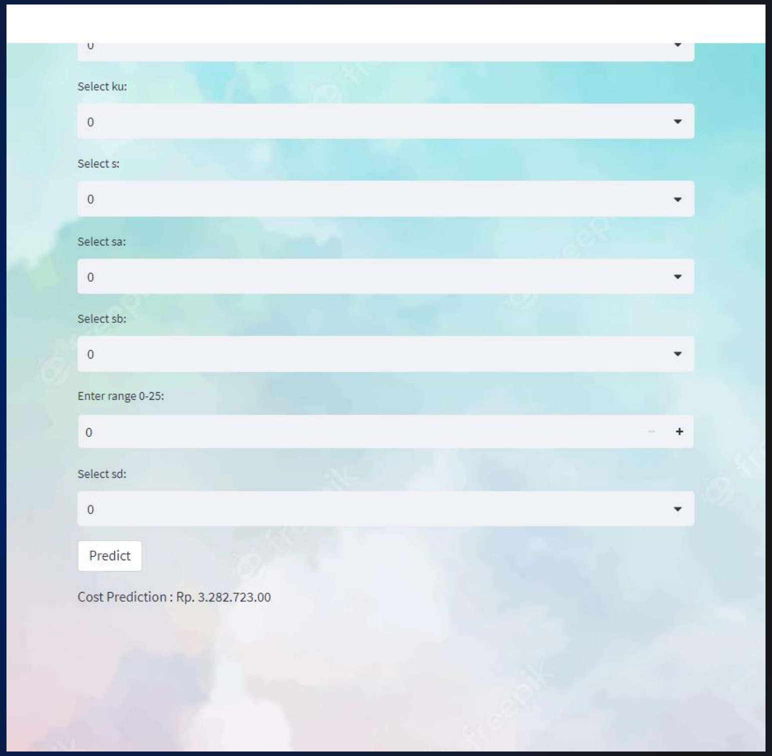
Export Joblib File for Development

```
In [214]:
#Exporting the model using joblib library
import joblib
joblib.dump(xgb_best,"../BPJS_CostPrediction_xgb.pkl")

Out[214]: ['../BPJS_CostPrediction_xgb.pkl']

In [223]:
import xgboost as xgb
xgb_best.save_model("../BPJS_CostPrediction_xgb.txt")
```

We have joblib file for, and we will load that file in Streamlit for development.

MODELLING
AND DEVELOPMENT





Link for The App : [Streamlit]

And now we can predict the ‘unit_cost’

THANK YOU

Github	: https://github.com/yodialfa/
LinkedIn	: https://www.linkedin.com/in/yodialfariz/
Email	: yodialfariz@gmail.com
Phone	: 082218293933
Twitter	: @yodiumh