

Soal Mentoring Meet 3 Machine Learning Process

Job Preparation Program, Data Science, Pacmann AI

Introduction

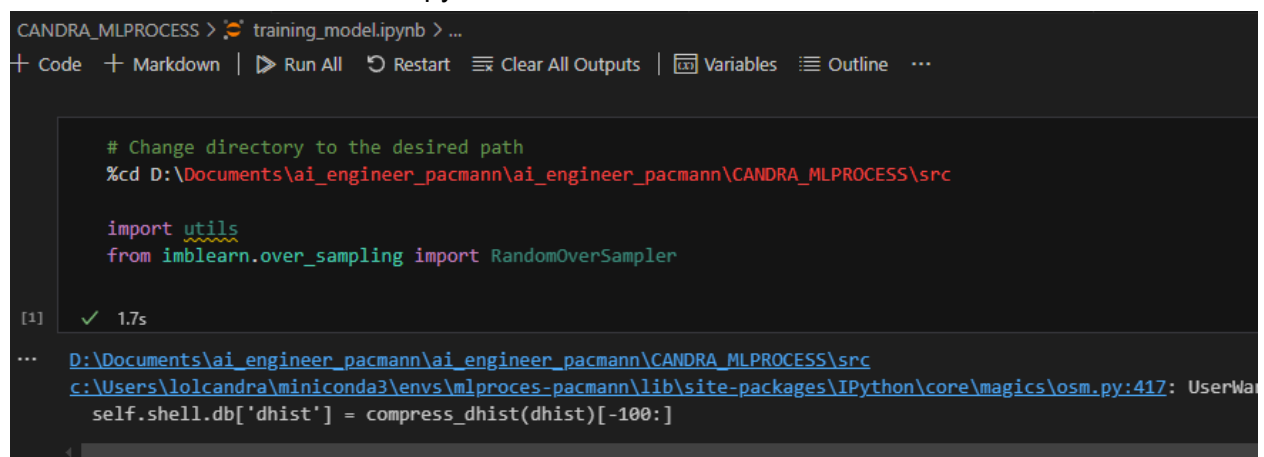
Kelanjutan dari Week 2, dengan menggunakan dataset yang sama, Anda diminta untuk melanjutkan Pemodelan dan Evaluasi. Pemodelan akan dilakukan dengan optimisasi hyperparameter serta tuning threshold (klasifikasi).

Catatan:

- Tulis jawaban Anda dalam jupyter notebook
- Copy file docs ini untuk menjawab dan sertakan screenshotnya
- Ekspor docs yang telah ada jawabannya ke PDF
- Archive file notebook dan file PDF Anda ke ZIP, **kecuali folder venv**
- Beri nama `[NAMA_LENGKAP]_MLPROCESS_3` file archive Anda

Soal

1. **[50 poin]** Pemodelan
 - a. Muat library yang dibutuhkan
 - i. Muat paling tidak 3 algoritma model klasifikasi berbeda (Saya ingin menggunakan Baseline model, k-NN (classifier), Decision Tree Classifier)
 - ii. Muat library hyperparameter tuning
 - iii. Muat utils dari src
 - iv. Muat numpy



```
CANDRA_MLPROCESS > training_model.ipynb > ...
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ...

# Change directory to the desired path
%cd D:\Documents\ai_engineer_pacmann\ai_engineer_pacmann\CANDRA_MLPROCESS\src

import utils
from imblearn.over_sampling import RandomOverSampler

[1] ✓ 1.7s

... D:\Documents\ai_engineer_pacmann\ai_engineer_pacmann\CANDRA_MLPROCESS\src
c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\IPython\core\magics\osm.py:417: UserWarning
self.shell.db['dhist'] = compress_dhist(dhist)[-100:]
```

Import some sklearn models

```
#####  
# Import sklearn library of those six models + gridsearchcv  
#  
#####  
  
from sklearn.dummy import DummyClassifier  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.neighbors import KNeighborsClassifier
```

3] ✓ 0.0s

Import GridSearchCV for hyperparameter Tuning, and numpy

```
from sklearn.model_selection import GridSearchCV  
from sklearn.metrics import accuracy_score  
import numpy as np
```

✓ 0.0s

- b. Lakukan pelatihan model
 - i. Deserialize data latih
 - ii. Buat daftar hyperparameter dari tiap algoritma model
 1. Buat dalam bentuk dictionary dengan key sebagai nama hyperparameternya dan value sebagai daftar nilai hyperparameter tersebut
 2. Value dari dictionary tersebut bertipe data list untuk menyimpan daftar nilai hyperparameternya
 3. Berikan nilai untuk masing-masing hyperparameter minimal 2 nilai
 - iii. Buat instance tiap model dan berikan nilai -1 untuk parameter n_jobs (jika didukung oleh model yang Anda gunakan), hal ini untuk memberikan akses seluruh sumber daya komputasi pada model saat pelatihan
 - iv. Buat fungsi untuk melakukan pelatihan model
 1. Fungsi ini memiliki parameter untuk menerima data:
 - a. Instance model yang telah dibuat
 - b. Daftar hyperparameter yang telah dibuat
 - c. Data latih yang telah dimuat
 - d. Data target yang telah dimuat
 2. Buat instance untuk melakukan hyperparameter tuning

- a. Assign instance model ke instance ini agar pelatihan model melibatkan hyperparameter tuning
 - b. Assign daftar parameter yang telah dibuat ke instance ini
 - c. Set parameter `n_jobs` ke -1 untuk instance ini
 - d. Set parameter `verbose` ke 3 untuk instance ini
3. Lakukan pelatihan model melalui instance hyperparameter tuning tersebut
4. Kembalikan instance hyperparameter tuning tersebut dari fungsi jika telah dilakukan pelatihan
5. Buat docstring untuk fungsi ini
- v. Lakukan pelatihan pada semua instance model yang telah Anda buat melalui fungsi pelatihan model
- c. Serialize model yang telah dilatih
 - i. Simpan model yang telah Anda latih pada folder model

Deserialize from X dan y after RandomOversampling

```
# Call the deserialize function to save the dataset into variable  
  
X_train_prep = deserialize_data(path='../data/processed/X_train_ros.pkl')  
y_train_prep = deserialize_data(path='../data/processed/y_train_ros.pkl')
```

[14] ✓ 0.8s

Create a list of Hyperparameter for every algorithm model

```
# grid hyperparameter for Baseline model
##
base_param_grid = {
}

# grid hyperparameter for Baseline model
##
knn_param_grid = {
    'n_neighbors': [5, 10],
    'weights': ['uniform', 'distance'],
    'algorithm': ['auto', 'ball_tree'],
    'leaf_size': [30, 60],
    'p': [2],
    'metric': ['minkowski'],
    'metric_params': [None],
    'n_jobs': [None]
}

# grid hyperparameter untuk Decision Tree Classifier

dct_param_grid = {
    'criterion': ['gini', 'entropy'], # Function to measure the quality of a split
    'splitter': ['best', 'random'], # Strategy used to choose the split at each node
    'max_depth': [None, 10], # Hanya 2 nilai untuk max_depth
    'min_samples_split': [2, 5], # Hanya 2 nilai untuk min_samples_split
    'min_samples_leaf': [1, 2], # Hanya 2 nilai untuk min_samples_leaf
    'max_features': [None, 'sqrt'], # Hanya 2 nilai untuk max_features
    'class_weight': [None, 'balanced'] # Hanya 2 nilai untuk class_weight
}
```

✓ 0.0s

Create instance for every model (baseline, kNN, DecisionTreeClassifier)

```
# Baseline model untuk klasifikasi
# Strategi 'most_frequent' untuk selalu memprediksi kelas yang paling sering muncul
instance_baseline = DummyClassifier()

# Instance KNN
instance_knn = KNeighborsClassifier(n_jobs=-1)

# Instance Decision Tree Classifier (tidak mendukung n_jobs)
instance_dt = DecisionTreeClassifier()
```

[29] ✓ 0.0s

Python

Create a Function for Training the model

```
def train_model(model, param_grid, X_train, y_train):  
    """  
    Melakukan pelatihan model dengan hyperparameter tuning menggunakan GridSearchCV.  
  
    Parameters:  
    model : estimator  
    | Instance model yang akan dilatih.  
    param_grid : dict  
    | Daftar hyperparameter untuk tuning.  
    X_train : DataFrame atau ndarray  
    | Data latih.  
    y_train : Series atau ndarray  
    | Target data latih.  
  
    Returns:  
    grid_search : GridSearchCV  
    | Instance GridSearchCV yang telah dilatih.  
    """  
    # Membuat instance GridSearchCV  
    grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, n_jobs=-1, verbose=3)  
  
    # Melakukan pelatihan model  
    grid_search.fit(X_train, y_train)  
  
    # Mengembalikan instance GridSearchCV yang telah dilatih  
    return grid_search
```

[30]

✓ 0.0s

Python

Calling the Function for every model (baseline, kNN, DTC)

```
# For baseline model  
#  
baseline_grid_search = train_model(model=instance_baseline,  
    param_grid=base_param_grid,  
    X_train=X_train_prep,  
    y_train=y_train_prep)
```

[31]

✓ 1.4s

... Fitting 5 folds for each of 1 candidates, totalling 5 fits

```

# For kNN Classifier model
#
baseline_grid_knn = train_model(model=instance_knn,
                                param_grid=knn_param_grid,
                                X_train=X_train_prep,
                                y_train=y_train_prep)

```

[32] ✓ 10.1s

... Fitting 5 folds for each of 16 candidates, totalling 80 fits
c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\neighbors\cla
return self._fit(X, y)

```

# For DCT Classifier model
#
baseline_grid_dct = train_model(model=instance_dt,
                                param_grid=dct_param_grid,
                                X_train=X_train_prep,
                                y_train=y_train_prep)

```

[33] ✓ 13.4s

... Fitting 5 folds for each of 128 candidates, totalling 640 fits

Then Saving the models after fitting

```

# Call the serialize function to dump the models into .pkl files

serialize_data(data=baseline_grid_search, path='../models/baseline_model.pkl')
serialize_data(data=knn_grid_search, path='../models/kNN_model.pkl')
serialize_data(data=dct_grid_search, path='../models/dct_model.pkl')

```

[12] ✓ 0.7s

... ['../models/dct_model.pkl']

2. [50 poin] Tuning dan Evaluasi

a. Muat library yang dibutuhkan

- i. Muat Utils dari src
- ii. Muat numpy, seaborn, dan pandas
- iii. Muat classification report untuk metrics

```
from utils import load_data
from utils import split_input_output
from utils import split_train_test
from utils import serialize_data
from utils import deserialize_data

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import RandomOverSampler
from sklearn.metrics import classification_report
```

36] ✓ 0.0s

b. Lakukan threshold tuning

- i. Muat data validation
- ii. Muat model yang telah Anda latih
- iii. Buat fungsi untuk melakukan threshold tuning
 1. Fungsi ini memiliki parameter untuk menerima data:
 - a. Model yang telah dilatih
 - b. Data validasi
 - c. Data target validasi
 2. Lakukan prediksi yang menghasilkan probabilitas dari pada data validasi menggunakan model yang telah Anda latih
 3. Buat perulangan dari minimal 100 nilai dari 0 hingga 1
 - a. Lakukan thresholding pada probabilitas hasil prediksi tersebut
 - b. Lakukan perhitungan metrics menggunakan classification report
 - c. Ambil bagian metrics (recall, f1-score, atau precision) yang diinginkan oleh user pada interview di bagian meet 1
 - d. Buat struktur data dictionary yang berisi:
 - i. Nama model (e.g. logistic regression, random forest, etc.)
 - ii. Nilai threshold yang digunakan dalam perhitungan metrics saat ini

- iii. Metrics yang digunakan (e.g. macro avg recall, f1-score, accuracy, etc.)
 - e. Simpan struktur data tersebut untuk diplotting nanti
- 4. Plotting hasil dari metrics vs threshold yang telah dilakukan
- 5. Kembalikan daftar nilai threshold
- 6. Buat docstring untuk fungsi ini
- iv. Lakukan threshold tuning pada setiap model yang telah Anda latih dengan menggunakan fungsi yang telah Anda buat dan dengan menggunakan data validation
- v. Simpan daftar nilai threshold dari semua model yang telah Anda lakukan threshold tuning
- vi. Buat fungsi untuk memilih threshold terbaik dan model terbaik berdasarkan metrics yang digunakan
 - 1. Fungsi ini memiliki parameter:
 - a. Daftar threshold dari semua model yang telah Anda lakukan threshold tuning (jika ada n model, maka seharusnya ada n elemen dari daftar threshold)
 - b. Path untuk menyimpan informasi hasil terbaik nantinya
 - 2. Gabungkan semua daftar threshold menjadi satu dataframe
 - 3. Sort by metrics, descending
 - 4. Ambil index pertama dari daftar threshold
 - 5. Simpan threshold terpilih tersebut sesuai dengan path dan dalam bentuk JSON
 - 6. Kembalikan threshold terpilih tersebut
 - 7. Buat docstring untuk fungsi ini
- vii. Jalankan fungsi pemilihan threshold terbaik

2. Threshold Tuning (Data Validation)

```
[28] %pwd
✓ 0.0s
... 'D:\\Documents\\ai_engineer_pacmann\\ai_engineer_pacmann\\CANDRA_MLPROCESS\\src'
```

```
# Call the deserialize function to save the dataset into variable
X_valid_model = deserialize_data(path='../data/processed/X_valid_prep.pkl')

# Menggunakan data y_valid yang original (yang tidak dilakukan preprocessing atau feature Engineering)
# Karena data target haruslah original
##
y_valid_model = deserialize_data(path='../data/interim/y_valid.pkl')
[30] ✓ 0.0s
```



```

def threshold_tuning(trained_model, X_valid, y_valid, metric='f1-score', thresholds=100):
    """
    Melakukan tuning threshold untuk model yang telah dilatih.

    Parameters:
    -----
    trained_model : objek model yang telah dilatih
        Model yang akan digunakan untuk prediksi dan threshold tuning.
    X_valid : array-like
        Data validasi untuk dilakukan prediksi.
    y_valid : array-like
        Data target validasi yang sebenarnya.
    metric : str, opsional, default: 'f1-score'
        Metrics yang akan digunakan untuk evaluasi. Pilihan: 'recall', 'precision', 'f1-score', 'accuracy', dll.
    thresholds : int, default: 100
        Jumlah threshold yang akan dieksplorasi dari 0 hingga 1.

    Returns:
    -----
    list
        Daftar nilai threshold yang dieksplorasi.
    """
    # Prediksi probabilitas dari data validasi
    y_pred_proba = trained_model.predict_proba(X_valid)[:, 1]

    # Inisialisasi daftar untuk menyimpan hasil metrics
    thresholds_list = np.linspace(0, 1, thresholds)
    metrics_values = []

    # Lakukan perulangan untuk setiap nilai threshold
    for threshold in thresholds_list:
        # Lakukan thresholding probabilitas
        y_pred_threshold = (y_pred_proba > threshold).astype(int)

        # Hitung metrics menggunakan classification report
        report = classification_report(y_valid, y_pred_threshold, output_dict=True)

        # Ambil nilai metrics yang diinginkan
        if metric in report['macro avg']:
            metric_value = report['macro avg'][metric]
        else:
            raise ValueError(f"Metric '{metric}' tidak ditemukan dalam laporan klasifikasi.")

        # Simpan hasil metrics
        metrics_values.append({
            'model_name': type(trained_model).__name__,
            'threshold': threshold,
            'metric_value': metric_value,
        })

    return metrics_values

```

[31]

✓ 0.0s

Python

Melakukan Threshold Tuning pada model baseline, kNN, DCT

```
# Melakukan threshold tuning dengan model baseline
baseline_thresholds_data = threshold_tuning(trained_model=baseline_grid_search,
                                             X_valid=X_valid_model,
                                             y_valid=y_valid_model,
                                             metric='f1-score')

[33] ✓ 0.5s Python
```

... c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined: No labeled samples in true set

... c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined: No labeled samples in true set

... c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined: No labeled samples in true set

... c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined: No labeled samples in true set

... c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined: No labeled samples in true set

... c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined: No labeled samples in true set

```
# Melakukan threshold tuning dengan model k-NN
knn_thresholds_data = threshold_tuning(trained_model=knn_grid_search,
                                       X_valid=X_valid_model,
                                       y_valid=y_valid_model,
                                       metric='f1-score')

[34] ✓ 0.8s Python
```

... c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined: No labeled samples in true set

... c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined: No labeled samples in true set

... c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined: No labeled samples in true set

... c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined: No labeled samples in true set

```
# Melakukan threshold tuning dengan model DCT
dct_thresholds_data = threshold_tuning(trained_model=dct_grid_search,
                                       X_valid=X_valid_model,
                                       y_valid=y_valid_model,
                                       metric='f1-score')

[35] ✓ 0.6s Python
```

... c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined: No labeled samples in true set

... c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined: No labeled samples in true set

... c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined: No labeled samples in true set

... c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision is ill-defined: No labeled samples in true set

Membuat Fungsi untuk Thresholds Terbaik

```
import json

def select_best_threshold(thresholds_data, save_path):
    """
    Memilih threshold terbaik dan model terbaik berdasarkan metrics yang digunakan.

    Parameters:
    -----
    thresholds_data : list
        Daftar hasil threshold tuning dari semua model.
    save_path : str
        Path untuk menyimpan informasi hasil terbaik.

    Returns:
    -----
    float
        Threshold terpilih.
    """
    # Konversi daftar threshold ke dalam DataFrame
    df_thresholds = pd.DataFrame(thresholds_data)

    # Urutkan berdasarkan metrics, descending
    df_thresholds_sorted = df_thresholds.sort_values(by='metric_value', ascending=False)

    # Ambil threshold terpilih (nilai tertinggi)
    best_threshold = df_thresholds_sorted.iloc[0]['threshold']

    # Simpan threshold terpilih ke dalam file JSON
    with open(save_path, 'w') as f:
        json.dump({'best_threshold': best_threshold}, f)

    return best_threshold
```

The screenshot shows a Jupyter Notebook interface. On the left, a file explorer displays a directory structure for 'AI_ENGINEER_PACMANN' with subdirectories 'data', 'processed', 'raw', and 'models'. The 'models' directory contains files like 'baseline_best_threshold.json', 'baseline_model.pkl', 'dct_best_threshold.json', 'dct_model.pkl', 'knn_best_threshold.json', and 'knn_model.pkl'. The main notebook area shows a code cell with the following content:

```
# Menyimpan hasil threshold tuning dalam file JSON

# Baseline
baseline_best_threshold = select_best_threshold(thresholds_data = baseline_thresholds_data,
                                                save_path= '../models/baseline_best_threshold.json')

# k-NN
knn_best_threshold = select_best_threshold(thresholds_data = knn_thresholds_data,
                                          save_path= '../models/knn_best_threshold.json')

# dct
dct_best_threshold = select_best_threshold(thresholds_data = dct_thresholds_data,
                                          save_path= '../models/dct_best_threshold.json')
```

c. Lakukan evaluasi model

- i. Muat data test
- ii. Buat fungsi untuk melakukan evaluasi model
 1. Fungsi ini memiliki parameter:
 - a. Model yang telah dilatih
 - b. Data threshold yang telah terpilih
 - c. Data test
 - d. Data target test

2. Lakukan prediksi probabilitas untuk data latih tersebut dengan menggunakan model yang terbaik berdasarkan fungsi pemilihan threshold terbaik pada poin b
 3. Lakukan thresholding dengan menggunakan threshold yang telah dipilih dari fungsi pemilihan threshold terbaik pada poin b
 4. Print hasil classification reportnya
- iii. Lakukan evaluasi model menggunakan fungsi tersebut

~ c. Evaluasion model with Test set (X_test y_test)

```
# Call the deserialize function to save the dataset into variable

X_test_model = deserialize_data(path='../data/processed/X_test_prep.pkl')

y_test_model = deserialize_data(path='../data/interim/y_test.pkl')
```

[47]

✓ 0.6s

▷ ~

```
def evaluate_model(model, best_threshold, X_test, y_test):
    """
    Evaluasi model dengan menggunakan threshold terbaik yang telah dipilih.

    Parameters:
    - model: Trained model
    - best_threshold: Selected best threshold
    - X_test: Test data
    - y_test: Test target data

    Returns:
    - classification report
    """
    # Prediksi probabilitas untuk data test
    y_prob = model.predict_proba(X_test)[:, 1]

    # Lakukan thresholding
    y_pred = (y_prob >= best_threshold).astype(int)

    # Print hasil classification report
    print(classification_report(y_test, y_pred))
```

[49]

✓ 0.0s

Python

```
# Melakukan evaluasi baseline model
baseline_evaluate = evaluate_model(model= baseline_grid_search,
                                   best_threshold = baseline_best_threshold,
                                   X_test = X_test_model,
                                   y_test = y_test_model)
```

[51] ✓ 0.0s

Python

```
...
precision recall f1-score support
0 0.78 1.00 0.88 2548
1 0.00 0.00 0.00 711

accuracy 0.78 3259
macro avg 0.39 0.50 0.44 3259
weighted avg 0.61 0.78 0.69 3259
```

c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision-Recall scores were not calculated because the predicted labels were empty. The scores will be 0.0 unless data is supplied.

c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision-Recall scores were not calculated because the predicted labels were empty. The scores will be 0.0 unless data is supplied.

c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: UndefinedMetricWarning: Precision-Recall scores were not calculated because the predicted labels were empty. The scores will be 0.0 unless data is supplied.

```
# Melakukan evaluasi k-NN model
knn_evaluate = evaluate_model(model= knn_grid_search,
                               best_threshold = knn_best_threshold,
                               X_test = X_test_model,
                               y_test = y_test_model)
```

[52] ✓ 0.1s

```
...
precision recall f1-score support
0 0.89 0.87 0.88 2548
1 0.58 0.63 0.60 711

accuracy 0.82 3259
macro avg 0.74 0.75 0.74 3259
weighted avg 0.83 0.82 0.82 3259
```

```
# Melakukan evaluasi DTC model
dct_evaluate = evaluate_model(model= dct_grid_search,
                                best_threshold = dct_best_threshold,
                                X_test = X_test_model,
                                y_test = y_test_model)
```

[53] ✓ 0.0s

```
...
      precision    recall  f1-score   support

      0         0.00      0.00      0.00      2548
      1         0.22      1.00      0.36       711

 accuracy          0.22      3259
 macro avg         0.11      0.50      0.18      3259
 weighted avg      0.05      0.22      0.08      3259
```

c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: Under

_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: Under

_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

c:\Users\lolcandra\miniconda3\envs\mlproces-pacmann\lib\site-packages\sklearn\metrics_classification.py:1517: Under

_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))