

LAMPIRAN

Lampiran 1: URL *Dataset*, ECLAT *Algorithm*, SHAP – XAI, Streamlit *Application*

Link dataset dimensi koil dan impedansi transformator di PT. X - Kaggle

<https://www.kaggle.com/datasets/yegiajoehan/rekap-coil-dataset/>

Link ECLAT algorithm untuk 9 parameter - Kaggle

<https://www.kaggle.com/code/yegiajoehan/eclat-ta-2024>

Link SHAP – XAI algorithm untuk model XGBoost - Kaggle

<https://www.kaggle.com/code/yegiajoehan/coil-impedance-prediction-ta-2024>

Link Streamlit Application Prediksi Koil Impedansi x SHAP - Streamlit

<https://coil-impedance-ta2024.streamlit.app>

Link Streamlit Application Code - Github

<https://github.com/yoeltan/coil-impedance-prediction-main>

Link database Streamlit Application – Google Sheet

<https://docs.google.com/spreadsheets/d/1Kyqfi27LpWL51dn3Hfr6wNVyi1YrnB4hgTqARJNMV0s/edit?usp=sharing>

Link ARM ECLAT Rules 9 Parameter – Google Sheet

https://docs.google.com/spreadsheets/d/1X7ATicAgme0iRBz6S2nqdtoBD6riINZ1/edit?usp=drive_link&ouid=108554928609129954325&rtpof=true&sd=true

Lampiran 2: ECLAT *Algorithm* pada Kaggle – 8 Parameter

```
# Mengimpor pustaka yang diperlukan untuk analisis data
import numpy as np
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.preprocessing import TransactionEncoder

# Membaca data dari file Excel pada sheet 'association_fixwidth_imp'
data1 = pd.read_excel('/kaggle/input/rekap-coil-dataset/rekap_coil_v4.xlsx',
'association_fixwidth_imp')

# Menghapus kolom-kolom yang tidak diperlukan ('TID LV', 'TOD LV', 'TID HV', 'TOD HV')
data1.drop(columns=['TID LV', 'TOD LV', 'TID HV', 'TOD HV'], inplace=True)

# Mengonversi DataFrame menjadi numpy array
arr1 = data1.to_numpy() # data dari sheet 'association_fixwidth_imp'
data = arr1

# Menampilkan DataFrame untuk melihat data yang dimuat
data1

# Menginisialisasi TransactionEncoder
te = TransactionEncoder()

# Mengubah data transaksi menjadi format yang bisa digunakan oleh apriori
te_ary = te.fit(data).transform(data)

# Membuat DataFrame dari data transaksi yang telah diubah
df = pd.DataFrame(te_ary, columns=te.columns_)

# Menampilkan DataFrame untuk melihat data yang telah diubah
df

# Menggunakan algoritma apriori untuk menemukan frequent itemsets dengan minimum
support 0.01
```

```

frequent_itemsets = apriori(df, min_support=0.01, use_colnames=True)
# Menampilkan frequent itemsets untuk melihat data yang telah dibuat
frequent_itemsets

# Menghasilkan aturan asosiasi dari frequent itemsets dengan metric 'confidence' dan minimum
threshold 1
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=1)

# Mengurutkan aturan berdasarkan nilai confidence dan lift secara descending
rules = rules.sort_values(['confidence', 'lift'], ascending=[False, False])

# Memilih kolom yang relevan dari aturan asosiasi
rules_confidence = rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']]

# Membuat DataFrame dari aturan asosiasi
df = pd.DataFrame(rules_confidence, columns=['antecedents', 'consequents', 'support',
'confidence', 'lift'])

# Mengonversi kolom 'antecedents' menjadi string dan menambahkan kata "Jika"
df['antecedents'] = df['antecedents'].apply(lambda x: ', '.join(x))

# Mengonversi kolom 'consequents' menjadi string dan menambahkan kata "Maka"
df['consequents'] = df['consequents'].apply(lambda x: ', '.join(x))

# Menambahkan "jika" pada kolom 'antecedents' dan "maka" pada kolom 'consequents'
df['antecedents'] = "Jika " + df['antecedents']
df['consequents'] = "Maka " + df['consequents']

# Menyimpan DataFrame ke dalam file Excel dengan nama 'eclat_rules_Confidence.xlsx'
excel_file = "eclat_rules_Confidence.xlsx"
df[['antecedents', 'consequents', 'support', 'confidence', 'lift']].to_excel(excel_file, index=False)
# Menampilkan aturan asosiasi dengan confidence tertinggi
rules_confidence

```

Lampiran 3: ECLAT *Algorithm* pada Kaggle – 9 Parameter

```
# Mengimpor pustaka yang diperlukan untuk analisis data
import numpy as np
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.preprocessing import TransactionEncoder

# Membaca data dari file Excel pada sheet 'association_fixwidth_imp'
data1 = pd.read_excel('/kaggle/input/rekap-coil-dataset/rekap_coil_v4.xlsx',
'association_fixwidth_imp')

# Menghapus kolom-kolom yang tidak diperlukan ('TID LV', 'TOD LV', 'TID HV', 'TOD HV')
data1.drop(columns=['TOD LV','TID HV','TOD HV'], inplace=True)

# Mengonversi DataFrame menjadi numpy array
arr1 = data1.to_numpy() # data dari sheet 'association_fixwidth_imp'
data = arr1

# Menampilkan DataFrame untuk melihat data yang dimuat
data1

# Menginisialisasi TransactionEncoder
te = TransactionEncoder()

# Mengubah data transaksi menjadi format yang bisa digunakan oleh apriori
te_ary = te.fit(data).transform(data)

# Membuat DataFrame dari data transaksi yang telah diubah
df = pd.DataFrame(te_ary, columns=te.columns_)

# Menampilkan DataFrame untuk melihat data yang telah diubah
df

# Menggunakan algoritma apriori untuk menemukan frequent itemsets dengan minimum
support 0.01
```

```

frequent_itemsets = apriori(df, min_support=0.01, use_colnames=True)
# Menampilkan frequent itemsets untuk melihat data yang telah dibuat
frequent_itemsets

# Menghasilkan aturan asosiasi dari frequent itemsets dengan metric 'confidence' dan minimum
threshold 1
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=1)

# Mengurutkan aturan berdasarkan nilai confidence dan lift secara descending
rules = rules.sort_values(['confidence', 'lift'], ascending=[False, False])

# Memilih kolom yang relevan dari aturan asosiasi
rules_confidence = rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']]

# Membuat DataFrame dari aturan asosiasi
df = pd.DataFrame(rules_confidence, columns=['antecedents', 'consequents', 'support',
'confidence', 'lift'])

# Mengonversi kolom 'antecedents' menjadi string dan menambahkan kata "Jika"
df['antecedents'] = df['antecedents'].apply(lambda x: ', '.join(x))

# Mengonversi kolom 'consequents' menjadi string dan menambahkan kata "Maka"
df['consequents'] = df['consequents'].apply(lambda x: ', '.join(x))

# Menambahkan "jika" pada kolom 'antecedents' dan "maka" pada kolom 'consequents'
df['antecedents'] = "Jika " + df['antecedents']
df['consequents'] = "Maka " + df['consequents']

# Menyimpan DataFrame ke dalam file Excel dengan nama 'eclat_rules_Confidence.xlsx'
excel_file = "eclat_rules_Confidence.xlsx"
df[['antecedents', 'consequents', 'support', 'confidence', 'lift']].to_excel(excel_file, index=False)
# Menampilkan aturan asosiasi dengan confidence tertinggi
rules_confidence.

```

Lampiran 4: SHAP *Algorithm* pada Kaggle

```
# Mengimpor pustaka yang diperlukan untuk visualisasi SHAP
import shap
import pandas as pd
import matplotlib.pyplot as plt
from pandas_profiling import ProfileReport

# Menginisialisasi SHAP visualisasi
shap.initjs()

# Fungsi untuk membuat nilai SHAP dan penjelasan
def create_shapval(explainer, test_data):
    # Membuat DataFrame dari nilai SHAP dengan kolom yang diberi label
    shap_values = pd.DataFrame(explainer.shap_values(test_data), columns=[str(col+' Shap') for
col in test_data.columns])
    shap_values.index = test_data.index

    # Membuat objek SHAP Explanation
    explanation = shap.Explanation(shap_values.values, data=test_data, base_values=Y.mean(),
feature_names=test_data.columns)
    print("Explanation Length :", len(explanation))
    return shap_values, explanation

# Menggunakan model XGBoost untuk membuat TreeExplainer
xgb_explainer = shap.TreeExplainer(xgb)

# Menghasilkan nilai SHAP dan penjelasan menggunakan fungsi create_shapval
xgb_shap, xgb_explanation = create_shapval(xgb_explainer, X)

# Menyimpan data input dan nilai SHAP ke dalam file Excel
X.to_excel("X.xlsx")
xgb_shap.to_excel("xgb_shap.xlsx")
```

```

# Membuat laporan profil dari nilai SHAP menggunakan ProfileReport
ProfileReport(xgb_shap)

# Fungsi untuk membuat plot SHAP
def shap_plot(plot_type, shap_val, feature_data, explainer=None, explanation=None,
index1=None, index2=None, show=True):

    # Fungsi untuk memeriksa keberadaan argumen
    def safe(to_check):
        if to_check is not None:
            return True
        else:
            print("ERROR: Use the right argument on", to_check)
            return False

    # Plot tipe summary
    if plot_type == 'summary':
        shap.summary_plot(shap_val,                                     features=feature_data,
feature_names=feature_data.columns, show=show)

    # Plot tipe decision
    elif plot_type == 'decision':
        return False if not safe(explainer) else shap.decision_plot(explainer.expected_value,
shap_val, features=feature_data)

    # Plot tipe bar
    elif plot_type == 'bar':
        shap.bar_plot(shap_val, feature_data.values[index1], feature_data.columns)

    # Plot tipe waterfall
    elif plot_type == 'waterfall':
        shap.plots.waterfall(explanation,          max_display=len(feature_data.columns))      #
max_display=data.columns

# Membuat plot summary SHAP dan menyimpannya sebagai gambar
summary_fig = shap_plot('summary', xgb_shap.values, X, show=False)

```

```
plt.savefig('summaryAll_fig.png')
```

```
# Membuat plot decision SHAP
```

```
shap_plot('decision', xgb_shap.values, X, explainer=xgb_explainer)
```

```
# Membuat plot bar SHAP untuk satu sampel data
```

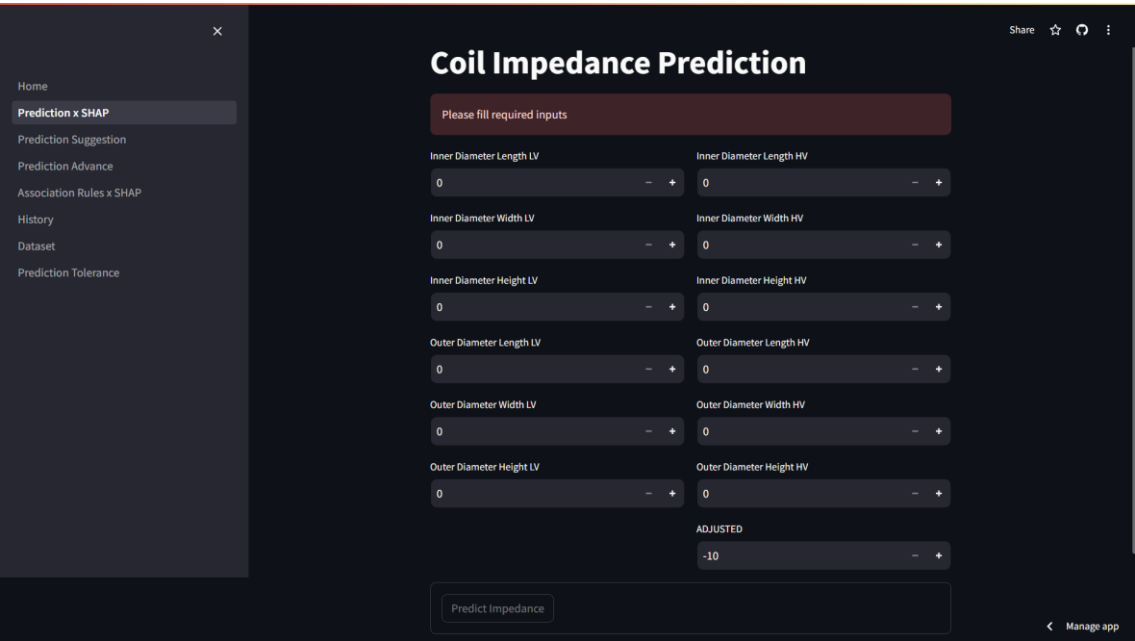
```
shap_plot('bar', xgb_shap.values[0], X, index1=0)
```

```
# Membuat plot waterfall SHAP untuk satu sampel data dan menyimpannya sebagai gambar
```

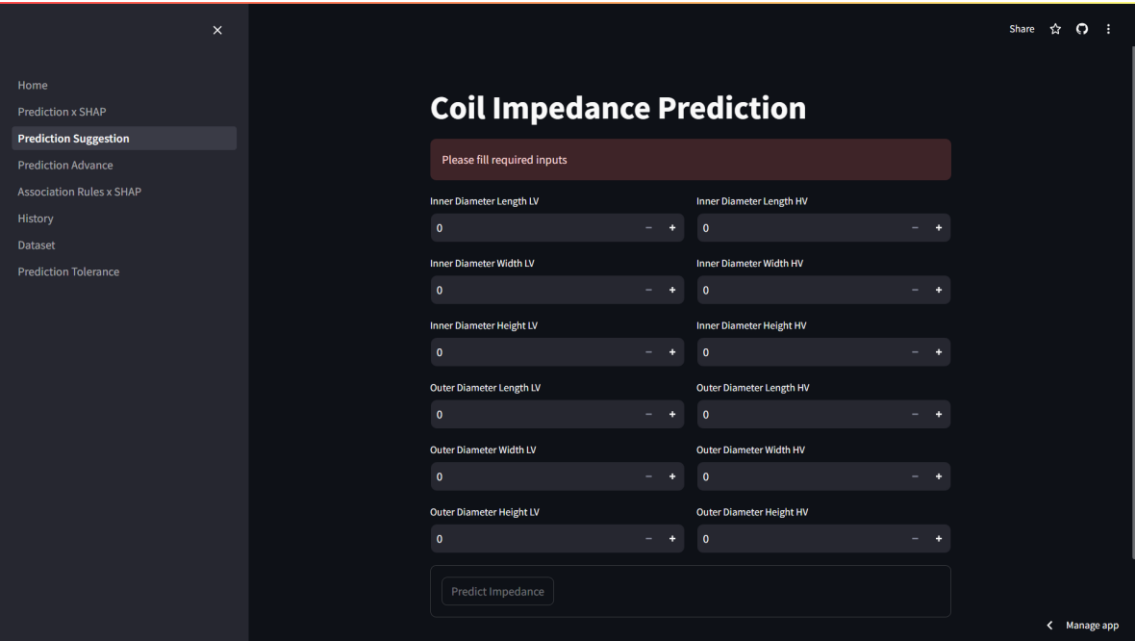
```
shap_plot('waterfall', None, X, explanation=xgb_explanation[0])
```

```
plt.savefig('shap_plot.png')
```

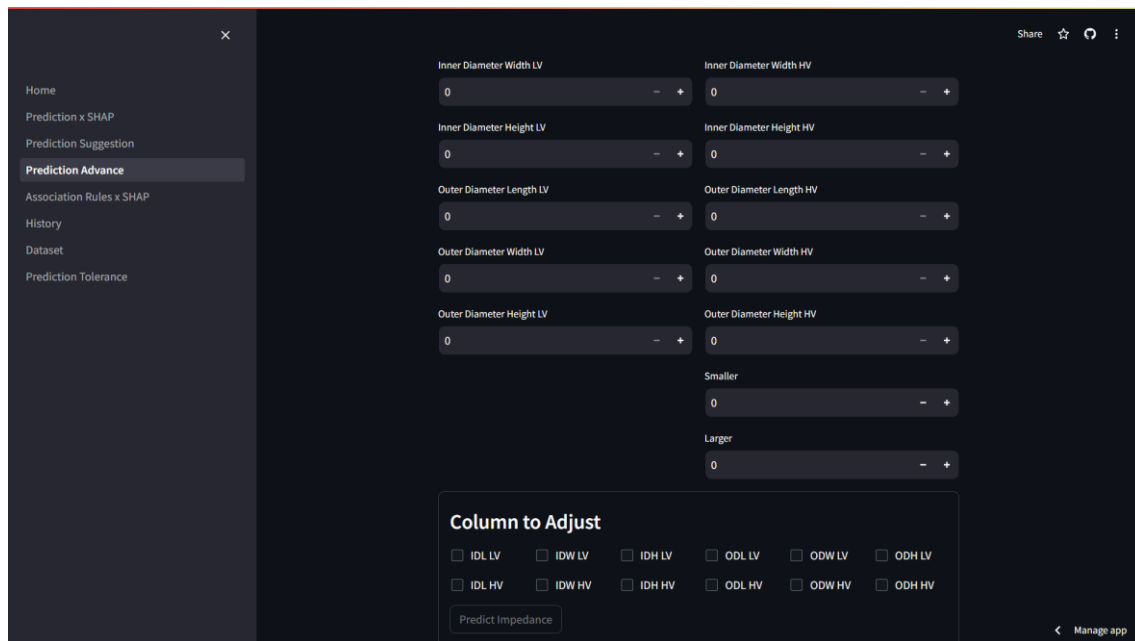
Lampiran 5: *User Interface Streamlit Application*



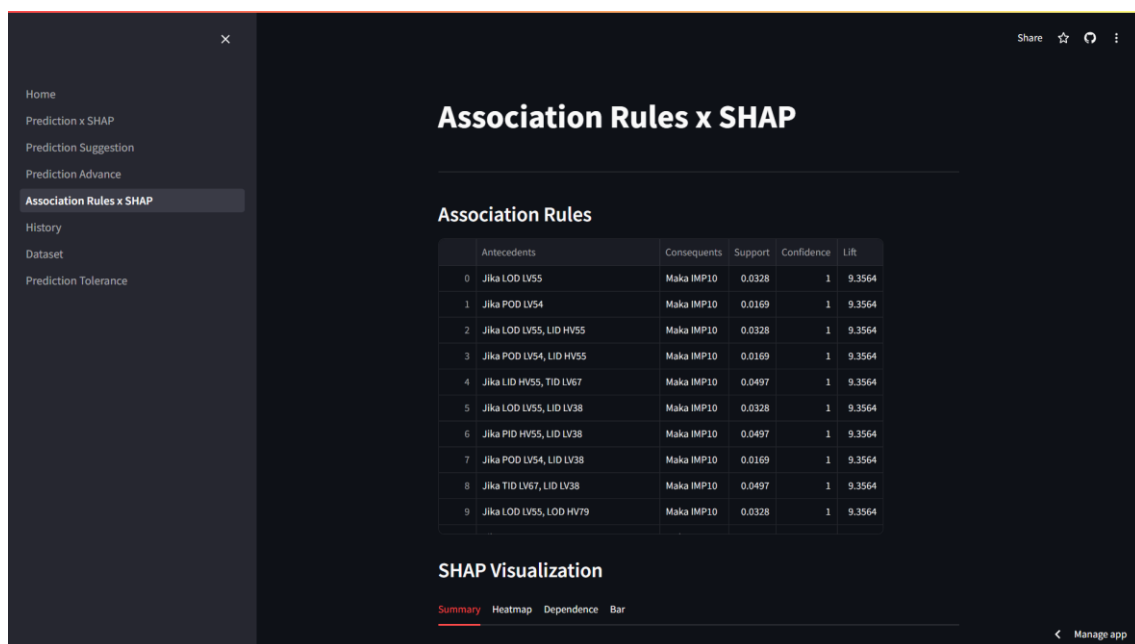
User Interface Prediction x SHAP



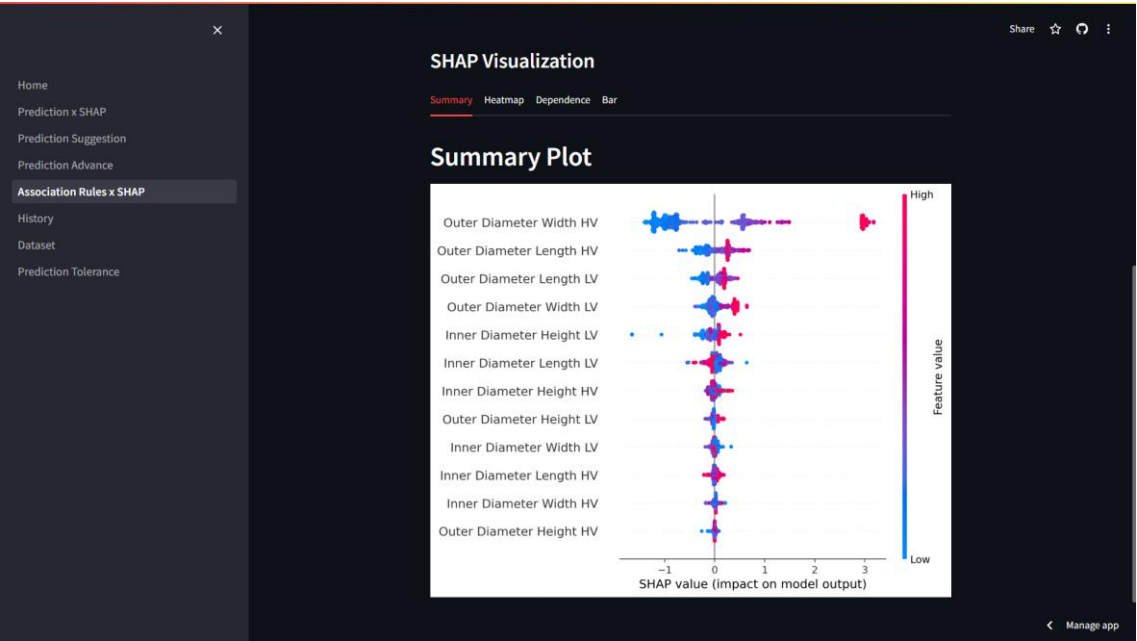
User Interface Prediction Suggestion



User Interface Prediction Advance



User Interface Association Rules x SHAP



User Interface Association Rules x SHAP

Prediction History

	Predict Time	Inner Diameter Length LV	Inner Diameter Width LV	Inner Diameter Height LV	Outer Diameter Length LV
0	2024-06-07 15:08:56	217	163	340	
1	2024-06-10 6:47:44	217	163	340	
2	2024-06-10 6:49:33	217	163	340	
3	2024-06-10 7:36:38	217	163	340	
4	2024-06-11 0:18:46	99	99	99	
5	2024-06-11 0:31:43	217	163	340	
6	2024-06-12 1:26:13	217	163	340	
7	2024-06-14 11:25:21	217	163	340	

User Interface History

×

Share ☆ ↺ ⋮

Home

Prediction x SHAP

Prediction Suggestion

Prediction Advance

Association Rules x SHAP

History

Dataset

Prediction Tolerance

Dataset Impedance

	Inner Diameter Length LV	Inner Diameter Width LV	Inner Diameter Height LV	Outer Diameter Length LV
0	187	152	220	197
1	152	133	320	254
2	133	253	343	325
3	104	103	286	150
4	226	183	345	330
5	236	183	340	333
6	226	183	345	331
7	210	163	550	319
8	152	134	320	255
9	217	163	340	324

Dataset Tolerance

	Inner Diameter Length LV	Inner Diameter Width LV	Inner Diameter Height LV	Outer Diameter Length LV
0	152	133	320	254
1	351	346	330	451
2	303	223	380	430
3	269	193	430	382

←

Manage app

User Interface Dataset

Lampiran 6: Lembar Pengesahan Proposal TA

PROGRAM STUDI TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS KRISTEN PETRA

USULAN TUGAS AKHIR

Nama : Yegia Joehan Eltan
NRP : C11200037
Bidang Studi : Teknik Tenaga Listrik
Judul Tugas Akhir : Penggunaan *Machine Learning* untuk Memprediksi Parameter yang Mempengaruhi Penyimpangan Impedansi dalam Proses Pembuatan Transformator di PT. Bambang Djaja
Pembimbing I : Ir. Julius Sentosa Setiadji, M.T.
Pembimbing II : Dr.Ing. Indar Sugiarto, M.Sc.
Dilaksanakan : Semester Genap / Tahun 2023/2024

Surabaya, 1 Desember 2023

Yang mengusulkan,


Yegia Joehan Eltan

Menyetujui:

Pembimbing I

Ir. Julius Sentosa Setiadji, M.T.

Pembimbing II

Dr. Ing. Indar Sugiarto, M.Sc.

Mengetahui :

Koordinator Tugas Akhir

Emmy
Hosea

Ir. Emmy Hosea, M.Eng.Sc.

Catatan:
MK WAJIB: SKS : 142 LEAP: A
- PENGANTAR LINGKUNGAN IPK : 3,47 EPT : 507
- PENGANTAR MANAJEMEN