LAMPIRAN

1. Surat Izin Penelitian Ke Perusahaan Ailesh

Bandar Lampung, 13 April 2025

Nomor : Penelitian.002/DMJ/DFIK/ Dit. Adm.Akd/Reg/IV - 2025
Lampiran : -
Perihal : Permohonan Izin Penelitian

Kepada Yth,
Bapak Cahaya Prautama Direktur Operasional Ailesh
Di –
Antasena 2, Gg. II RT.03/RW.18, Tegal Weru, Sariharjo, Kec. Ngaglik, Kab. Sleman, Daerah Istimewa Yogyakarta

Dengan hormat,

Sehubungan dengan peraturan Akademik Institut Informatika dan Bisnis (IIB) Darmajaya bahwa mahasiswa/i Strata Satu (S1) yang akan menyelesaikan studinya diwajibkan untuk memiliki pengalaman kerja dengan melaksanakan Penelitian dan membuat laporan yang waktunya disesuaikan dengan kalender Institut Informatika dan Bisnis (IIB) Darmajaya.

Untuk itu kami mohon kerja sama Bapak/Ibu agar kiranya dapat menerima mahasiswa/i untuk melakukan Penelitian, yang pelaksanaannya dimulai dari tanggal **18 April 2025 s.d 18 Mei 2025** (selama satu bulan)

Adapun mahasiswa/i tersebut adalah :

Nama	**: Hani Utari**
NPM	**: 2111050005**
Jurusan	**: S1 Sistem Informasi**
Jenjang	**: Strata Satu (S1)**

Demikian permohonan ini dibuat, atas perhatian dan kerjasama yang baik kami ucapkan terimakasih.

Dekan Fakultas Ilmu Komputer,

Dr. Muhammad Said Hasibuan, M.Kom
NIK. 01220905

Tembusan:
1. Program Studi S1 Sistem Informasi
2. Arsip.

Jalan Z.A. Pagar Alam, No.93, Labuhan Ratu, Bandar Lampung, Lampung
www.darmajaya.ac.id
info@darmajaya.ac.id
0721-787214
0721-700261

2. Code Analisis Data Penelitian Menggunakan Bahasa Pemrograman Python

```
# import library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
# import file
df = pd.read_csv("/content/ailesh.csv")

df.head()
df.shape
df.info()

# remove value
df = df[df.value != '#REF!']

# convert dtype
df['value'] = df['value'].astype(float)

# prompt: cek missing value

df.isnull().sum()

# Drop the 'persentase' column due to many missing values
df = df.drop('persentase', axis=1)

# Verify the column has been removed
df.columns

# prompt: cek data duplikat yang mempunyai nilai sama setiap
baris dengan kolom yang lain

# check for rows where all columns have the same value
duplicate_rows = df[df.apply(lambda row: row.nunique() == 1,
axis=1)]

print("Rows with duplicate values across all columns:")
```

```
duplicate_rows

#creat df consisting of only emission to air
columns =
['industry','sector','location','unit_process','input_output','ca
tegory','subcategory','material','material_rename','value','unit'
]
df_air = df[df['subcategory']=='Emission to Air'][columns]


# emisi masing2 industri
round(df_air.groupby('industry')['value'].sum().reset_index(), 2)

# emission ouput by sector
df_og = df_air[df_air['industry'] == 'Oil & Gas Distribution']
sector_og =
df_og.groupby('sector')['value'].sum().sort_values(ascending =
False).reset_index().head()
chart_sector_og = sector_og.sort_values(by = 'value', ascending =
True)
sector_og

# vizualization
plt.figure(figsize = (10,4))
plt.barh(chart_sector_og['sector'], # X
        chart_sector_og['value']) # Y
plt.title('Oil & Gas Emission by Sector')
plt.xlabel('Sector')
plt.ylabel('Emission Output')
plt.grid(axis = 'y', linestyle = '--', alpha = 0.6)
plt.show()

# 5 highest unit process
df_og = df_air[df_air['industry'] == 'Oil & Gas Distribution']
unit_process_og =
df_og.groupby('unit_process')['value'].sum().sort_values(ascendin
g = False).reset_index().head()
chart1 = unit_process_og.sort_values(by = 'value', ascending =
True)
unit_process_og
```

```
plt.figure(figsize = (10,4))
plt.barh(chart1['unit_process'], # X
        chart1['value']) # Y
plt.title('Oil & Gas Emission by Unit Process')
plt.xlabel('Unit Process')
plt.ylabel('CO2 Output')
plt.grid(axis = 'y', linestyle = '--', alpha = 0.6)
plt.show()

# breakdown use stage
df_og_usestage = df_og[df_og['unit_process'] == 'Use Stage']
usestage_og =
df_og_usestage.groupby('material_rename')['value'].sum().sort_val
ues(ascending = False).reset_index().head()
chart2 = usestage_og.sort_values(by = 'value', ascending = True)
usestage_og

# create chart
plt.figure(figsize = (10,4))
plt.barh(chart2['material_rename'], # X
        chart2['value']) # Y
plt.title('Use Stage')
plt.xlabel('Emission')
plt.ylabel('Output')
plt.grid(axis = 'y', linestyle = '--', alpha = 0.6)
plt.show()

# emission ouput by sector
df_pp = df_air[df_air['industry'] == 'Power Plant']
sector_pp =
df_pp.groupby('sector')['value'].sum().sort_values(ascending =
False).reset_index().head()
chart_sector_pp = sector_pp.sort_values(by = 'value', ascending =
True)
sector_pp
# vizualization
plt.figure(figsize = (10,4))
plt.barh(chart_sector_pp['sector'], # X
        chart_sector_pp['value']) # Y
plt.title('Power Plant Emission by Sector')
plt.xlabel('Sector')
```

```
plt.ylabel('Emission Output')
plt.grid(axis = 'y', linestyle = '--', alpha = 0.6)
plt.show()

# 5 highest unit process
df_pp = df_air[df_air['industry'] == 'Power Plant']
unit_process_pp =
df_pp.groupby('unit_process')['value'].sum().sort_values(ascendin
g = False).reset_index().head()
chart3 = unit_process_pp.sort_values(by = 'value', ascending =
True)
unit_process_pp

plt.figure(figsize = (10,4))
plt.barh(chart3['unit_process'], # X
        chart3['value']) # Y
plt.title('Power Plant Emission by Unit Process')
plt.xlabel('Unit Process')
plt.ylabel('Output')
plt.grid(axis = 'y', linestyle = '--', alpha = 0.6)
plt.show()

# breakdown HRSG
df_pp_hrsg = df_pp[df_pp['unit_process'] == 'Hrsg']
hrsg_pp =
df_pp_hrsg.groupby('material_rename')['value'].sum().sort_values(
ascending = False).reset_index().head()
chart4 = hrsg_pp.sort_values(by = 'value', ascending = True)
hrsg_pp


plt.figure(figsize = (10,4))
plt.barh(chart4['material_rename'], # X
        chart4['value']) # Y
plt.title('Hrsg')
plt.xlabel('Emission')
plt.ylabel('Output')
plt.grid(axis = 'y', linestyle = '--', alpha = 0.6)
plt.show()

# check locations with use stage
```

```
df_og_usestage['location'].unique()
# list location
# seperate location because difference in data recording method
location_og1 = ['Adi Sucipto', 'Ahmad Yani', 'Adi Sumarno']
location_og2 = ['Boyolali', 'Makassar', 'Cilacap', 'Rewulu',
'Bitung']

# list subcategory
subcategory_og = ['Material', 'Energy', 'Diesel', 'Chemical']

# first df filter location, input, and unit process
df_input_og1 = df[(df['location'].isin(location_og1)) &
(df['input_output'] == 'Input') & (df['unit_process'] == 'Use
Stage')]

# second df filter location, input, and subcategory
df_input_og2 = df[(df['location'].isin(location_og2)) &
(df['input_output'] == 'Input') &
(df['subcategory'].isin(subcategory_og))]

# concatenate both df
df_input_og = pd.concat([df_input_og1, df_input_og2], axis=0)

# groupby subcategory
df_subcategory_og =
df_input_og.groupby('subcategory')['value'].sum().sort_values(asc
ending = False).reset_index().head()
df_subcategory_og

# not included material
exclude = ['Feedwater', 'Unorganic Consumption', 'Organic
Consumption']

for i in subcategory_og:
  # filter df
  df_material_og = df_input_og[(df_input_og['subcategory'] == i)
& (~df_input_og['material_rename'].isin(exclude))]

  # df_material_og
```

```
  df_material_og =
df_material_og.groupby('material_rename')['value'].sum().sort_val
ues(ascending = False)

  # print
  print(i)
  print(df_material_og)

  # pie chart
  plt.figure(figsize = (5,5))
  plt.pie(df_material_og,
        labels = df_material_og.index,
        autopct = '%1.1f%%',
        startangle = 90,
        pctdistance = 0.5)
  plt.title(f'{i}')
  plt.legend(df_material_og.index, loc = 'best', bbox_to_anchor =
(1.0,1.0))
  plt.show()

# Siapkan daftar untuk menampung data
summary_data = []

# Loop setiap subkategori
for i in subcategory_og:
    df_material_og = df_input_og[
        (df_input_og['subcategory'] == i) &
        (~df_input_og['material_rename'].isin(exclude))
    ]

    # Grouping dan agregasi
    df_summary =
df_material_og.groupby('material_rename')['value'].sum().reset_in
dex()
    df_summary['subcategory'] = i  # Tambahkan nama subkategori
    summary_data.append(df_summary)

# Gabungkan semua hasil jadi satu dataframe
df_final = pd.concat(summary_data, ignore_index=True)

# Simpan ke CSV
```

```
df_final.to_csv('input_use_stage_summary.csv', index=False)

# check locations with HRSG
df_pp_hrsg['location'].unique()

# list location
location_pp = ['IP Grati', 'IP Cilegon']

# list subcategory
subcategory_pp = ['Material', 'Chemical', 'Fuel']

# filter
df_input_pp = df[(df['location'].isin(location_pp)) &
(df['input_output'] == 'Input') &
              (df['unit_process'] == 'Hrsg') &
(df['subcategory'].isin(subcategory_pp))]

# groupby subcategory
df_subcategory_pp =
df_input_pp.groupby('subcategory')['value'].sum().sort_values(asc
ending = False).reset_index().head()
df_subcategory_pp
for i in subcategory_pp:
  # filter df
  df_material_pp = df_input_pp[df_input_pp['subcategory'] == i]

  # df_material_pp
  df_material_pp =
df_material_pp.groupby('material_rename')['value'].sum().sort_val
ues(ascending = False)

  # print
  print(i)
  print(df_material_pp)

  # pie chart
  plt.figure(figsize = (5,5))
  plt.pie(df_material_pp,
        # labels = df_material_pp.index,
        autopct = '%1.1f%%',
        startangle = 90,
```

```
        pctdistance = 0.5)
  plt.title(f'{i}')
  plt.legend(df_material_pp.index, loc = 'best', bbox_to_anchor =
(1.0,1.0))
  plt.show()

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA


df = pd.read_csv ('/content/cleaned_emisi_udara_air.csv')


features = ['sector','location', 'unit_process',
'material_rename', 'value']

import pandas as pd
from sklearn.preprocessing import LabelEncoder
# Mengidentifikasi kolom kategorikal
categorical_cols =
df.select_dtypes(include=['object']).columns.tolist()

# Melakukan label encoding pada kolom kategorikal
label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le  # Menyimpan encoder jika diperlukan
untuk decoding di masa mendatang

# Menampilkan informasi setelah label encoding
print("\nInformasi Dataset Setelah Label Encoding:")
print(df.info())
```

```
# Menampilkan beberapa baris pertama setelah label encoding
print("\nBeberapa Baris Pertama Setelah Label Encoding:")
print(df.head())

# Normalisasi data
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)

# Menentukan jumlah cluster optimal menggunakan metode Elbow
inertia = []
K = range(1, 11)
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(df_scaled)
    inertia.append(kmeans.inertia_)

# Visualisasi Elbow Method
plt.figure(figsize=(6, 3))
plt.plot(K, inertia, 'bo-')
plt.title('Metode Elbow untuk Menentukan Jumlah Cluster')
plt.xlabel('Jumlah Cluster (k)')
plt.ylabel('Inertia')
plt.show()

# Clustering menggunakan K-Means (misalnya, 3 cluster)
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(df_scaled)

# Menambahkan hasil clustering ke data asli
df['Cluster'] = clusters

# Analisis hasil clustering
# Visualisasi PCA untuk 2D plot
pca = PCA(n_components=2)
df_pca = pca.fit_transform(df_scaled)

plt.figure(figsize=(8, 5))
sns.scatterplot(x=df_pca[:, 0], y=df_pca[:, 1],
hue=df['Cluster'], palette='Set1', s=100)
plt.title('Visualisasi Clustering dengan PCA')
```

```
plt.xlabel('Komponen Utama 1')
plt.ylabel('Komponen Utama 2')
plt.legend(title='Cluster')
plt.show()

# Menampilkan hasil clustering dan rata-rata fitur per cluster
cluster_avg = df.groupby('Cluster').mean(numeric_only=True)


# Menampilkan rata-rata nilai untuk tiap cluster
print("\nRata-rata Fitur per Cluster:")
print(cluster_avg)

# Calculate cluster summary
cluster_summary = df.groupby('Cluster').mean(numeric_only=True)

# Mapping cluster numerik ke label deskriptif
cluster_labels = {
    0: 'High air emissions',    # Cluster 0 mewakili tingkat
emisi sedang
    1: 'Low air emission',    # Cluster 1 mewakili tingkat emisi
rendah
    2: 'Moderate air emissions'     # Cluster 2 mewakili tingkat
emisi tinggi
}

# Menambahkan kolom baru 'Cluster_Label' dengan mengubah angka
menjadi label
df['Cluster_Label'] = df['Cluster'].map(cluster_labels)

print(df)
```