LAMPIRAN

```
from dbfread import DBF
from pandas import DataFrame
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

dbf = DBF('18_sak_202208_15.dbf')
frame = DataFrame(iter(dbf))

## Missing Value
from jcopml.plot import plot_missing_value
plot_missing_value(frame)
missing_values_count = frame.isnull().sum()
print(missing_values_count)
kolom_kosong = frame.columns[frame.isna().any()].tolist()
fitur = frame.columns
print(fitur)
for feature in fitur:
    print(feature)
```

```
sf                                                                    =
frame[['KLAS','K4','K6','R4','R6A','R6D','R6E','R8A','R8B','R8C','R8D','R8E',
'R8F',
        'R9A','R9B','R9C','R10','R38A','R38B','R42A','R47B']]
sf.R6E.value_counts()

# Imputation
plot_missing_value(sf)
sf[['R6E','R9B','R9C','R10','R47B']]                                  =
sf[['R6E','R9B','R9C','R10','R47B']].fillna(2)
sf['R42A'] = sf['R42A'].fillna(10)
def disabilitas(row):
    if (row['R8A'] == 4 and row['R8C'] == 4 and row['R8E'] == 4) and
      (row['R8B'] == 8 and row['R8D'] == 8 and row['R8F'] == 8):
      return 2
    else:
      return 1
sf['R8'] = sf.apply(disabilitas, axis=1)
idxR6E = sf.columns.get_loc('R6E')
sf.insert(idxR6E + 1, 'R8', sf.pop('R8'))

# Fungsi untuk melakukan binning umur
def binning_umur(umur):
```

```
    if umur <= 5:
        return 1
    elif 5 < umur <= 11:
        return 2
    elif 11 < umur <= 16:
        return 3
    elif 16 < umur <= 25:
        return 4
    elif 25 < umur <= 35:
        return 5
    elif 35 < umur <= 45:
        return 6
    elif 45 < umur <= 55:
        return 7
    elif 55 < umur <= 65:
        return 8
    else:
        return 9

# Menambahkan kolom baru K5 ke dalam dataset
sf['K6'] = sf['K6'].apply(binning_umur)
sf['TARGET'] = np.where(
                    (sf['R9A'] == 1) |
```

```
                (sf['R9B'] == 1) |
                (sf['R9C'] == 1) |
                (sf['R10'] == 1), 1, 2)
hapus =['R8A','R8B','R8C','R8D','R8E','R8F','R9A','R9B','R9C','R10']
sf = sf.drop(hapus, axis=1)
X = sf.drop(columns = 'TARGET')
y = sf.TARGET

from sklearn.svm import SVC
# Inisialisasi model SVM
model = SVC(kernel='linear', C=1.0, random_state=42)
model.fit(X,y)
model.score(X,y)

## Dataset Splitting
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y,
random_state=42)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder
numerical_pipeline = Pipeline([
```

```
    ('scaler',MinMaxScaler())
])
categorical_pipeline = Pipeline([
    ('onehot',OneHotEncoder())
])
from sklearn.compose import ColumnTransformer
preprocessor = ColumnTransformer([
    ('categoric',                                          categorical_pipeline,
['KLAS','K4','K6','R4','R6A','R6D','R6E','R8','R38A','R38B','R42A','R47B']),
])
from sklearn.svm import SVC
pipeline = Pipeline([
    ('prep', preprocessor),
    ('algo', SVC(max_iter=500))
])
pipeline.fit(X_train, y_train)

### Grid  Search
import pandas as pd
from sklearn.model_selection import GridSearchCV
from jcopml.tuning import grid_search_params as gsp
model = GridSearchCV(pipeline, gsp.svm_params, cv=4, scoring='f1',
n_jobs=-1, verbose=1)
```

```
model.fit(X_train, y_train)

pd.DataFrame(model.cv_results_).sort_values("rank_test_score")

model.best_params_

model.score(X_train, y_train), model.best_score_, model.score(X_test, y_test)


import matplotlib.pyplot as plt


usf = usf

klasifikasi_column = 'KLAS'

target_column = 'TARGET'


# Membuat grafik batang

fig, ax = plt.subplots(figsize=(8, 6))

usf.groupby(klasifikasi_column)[target_column].value_counts().unstack().plot

(kind='bar', color=['green', 'red'], ax=ax)


# Menambahkan nilai pada setiap batang

for p in ax.patches:

    width, height = p.get_width(), p.get_height()

    x, y = p.get_xy()

    ax.text(x + width/2, y + height/2, f'{height}', ha='center', va='center',

fontsize=10, color='black')
```

```
# Mengganti label pada sumbu x dan membuatnya horizontal
ax.set_xticklabels(['Perkotaan', 'Perdesaan'], rotation=0)

# Menambahkan judul dan label pada sumbu
plt.title('Pengaruh Klasifikasi Perkotaan/Perdesaan terhadap pengangguran')
plt.xlabel('Klasifikasi')
plt.ylabel('Jumlah')
plt.legend(title='Status', labels=['Bekerja', 'Pengangguran'])
plt.grid(True)
plt.show()
```