Tugas Mata Kuliah

Machine Learning Praktikum – Data Transformation



Oleh :

Ferry Triwantono – 082111633094

**FAKULTAS SAINS DAN TEKNOLOGI**

**PROGRAM STUDI SISTEM INFORMASI**

**UNIVERSITAS AIRLANGGA**

**2023**

**Data Transformation**

1. Matlab

Syntax :

sales = readtable("vgsales.csv");

sales\_var = sales(:,{'Year', 'NA\_Sales', 'EU\_Sales', 'JP\_Sales', 'Global\_Sales'});

*% Mendeteksi missing value*

if any(ismissing(sales\_var))

    disp('Berikut adalah nilai yang missing');

    disp(sales\_var(any(ismissing(sales\_var),2),:));

else

    disp('Tidak terdapat nilai yang missing');

end

*% Mengganti missing value dengan menghapusnya*

missingVal = rmmissing(sales\_var);

if any(ismissing(sales\_var))

    disp('Berikut hasil missing value yang sudah diperbaiki:');

    disp(missingVal);

else

    disp('Tidak terdapat missing value');

end

outlier = detOutlier(table2array(missingVal));

if ~isempty(outlier)

    disp('Berikut adalah outlier');

    disp(outlier);

else

    disp('Tidak terdapat outlier');

end

*% Mengganti outlier pada variabel input menggunakan mean*

if ~isempty(outlier)

    disp('Penggantian outlier');

    disp(filloutliers(sales\_var, 'nearest', 'mean'));

else

    disp('Tidak ada outlier')

end

*% Melakukan normalisasi pada variabel input*

normalized = normalize(sales\_var,"zscore");

disp('normalized')

*% Function yang dapat mendeteksi outlier dengan Quartiles*

function *outlier* = detOutlier(*df*)

    q1 = quantile(df, 0.25);

    q3 = quantile(df, 0.75);

    iqr = q3 - q1;

    outlier = df((df < (q1 - 1.5 \* iqr)) | (df > (q3 + 1.5 \* iqr)));

end

1. Python

*Syntax :*

*# DATA PREPROCESSING*

from sklearn import preprocessing

import pandas as pd

import numpy as np

*# Memuat dataset*

sales = pd.read\_csv('D:\Coolyeah\Mata Kuliah\SMT 4\Machine Learning Praktikum\Week\_4\Tugas\vgsales.csv')

*# Menentukan variabel input dan variabel output.*

inputVar = pd.DataFrame(sales.loc[:,['NA\_Sales', 'EU\_Sales', 'JP\_Sales', 'Year']])

outputVar = pd.DataFrame(sales['Global\_Sales'])

print("==============MISSING VALUE===============")

*# Mendeteksi missing value pada variabel input*

if inputVar.isnull().values.any():

    print("Berikut nilai-nilai yang hilang:")

    print(inputVar[inputVar.isnull().any(*axis*=1)])

else:

    print("Tidak terdapat missing value")

*# Mengganti missing value dengan menghapusnya*

isMissingIn = inputVar.dropna()

if inputVar.isnull().values.any():

    print("Berikut hasil missing value yang sudah diperbaiki:")

    print(isMissingIn)

else:

    print("Tidak terdapat missing value")

*# Mendeteksi missing value pada variabel output*

if outputVar.isnull().values.any():

    print("Berikut nilai-nilai yang hilang:")

    print(outputVar[inputVar.isnull().any(*axis*=1)])

else:

    print("Tidak terdapat missing value")

*# Mengganti missing value dengan menghapusnya*

isMissingOut = outputVar.dropna()

if outputVar.isnull().values.any():

    print("Berikut hasil missing value yang sudah diperbaiki:")

    print(isMissingOut)

else:

    print("Tidak terdapat missing value")

print("==============OUTLIER===============")

*# Function yang dapat mendeteksi outlier dengan Quartiles*

def detOutlier(*df*):

    q1 = *df*.quantile(0.25)

    q3 = *df*.quantile(0.75)

    iqr = q3 - q1

    outlier = *df*[((*df* < (q1 - 1.5 \* iqr)) | (*df* > (q3 + 1.5 \* iqr)))]

    return outlier

*# Deteksi outlier pada variabel input*

inOutlierShow = detOutlier(isMissingIn)

if not inOutlierShow.empty:

    print("Berikut outlier nya")

    print(inOutlierShow)

else:

    print("Tidak ada outlier")

*# Deteksi variabel output*

outOutlierShow = detOutlier(isMissingOut)

if not outOutlierShow.empty:

    print("Berikut outlier nya")

    print(outOutlierShow)

else:

    print("Tidak ada outlier")

*# Mengganti outlier pada variabel input dengan nilai mean*

def replaceOutlier(*df*):

    q1 = *df*.quantile(0.25)

    q3 = *df*.quantile(0.75)

    iqr = q3 - q1

    upper\_bound = q3 + 1.5 \* iqr

    lower\_bound = q1 - 1.5 \* iqr

    mean = *df*[(*df* > lower\_bound) & (*df* < upper\_bound)].mean()

*df*[*df* > upper\_bound] = mean

*df*[*df* < lower\_bound] = mean

    return *df*

*# Membuat dataframe untuk menampung hasil penggantian outlier*

inDataReplacedOutliers = pd.DataFrame()

*# melakukan looping pada setiap kolom*

for col in inputVar.columns:

*# mengganti outlier dengan mean di setiap kolom*

    colReplacedOutliers = replaceOutlier(inputVar[col])

*# menampung hasilnya*

    inDataReplacedOutliers[col] = colReplacedOutliers

*# menampilkan hasil penanganan outlier*

print(inDataReplacedOutliers)

*# Mengganti outlier pada variabel output dengan nilai mean*

def replaceOutlier(*df*):

    q1 = *df*.quantile(0.25)

    q3 = *df*.quantile(0.75)

    iqr = q3 - q1

    upper\_bound = q3 + 1.5 \* iqr

    lower\_bound = q1 - 1.5 \* iqr

    mean = *df*[(*df* > lower\_bound) & (*df* < upper\_bound)].mean()

*df*[*df* > upper\_bound] = mean

*df*[*df* < lower\_bound] = mean

    return *df*

*# Membuat dataframe untuk menampung hasil penggantian outlier*

outDataReplacedOutliers = pd.DataFrame()

*# melakukan looping pada setiap kolom*

for col in outputVar.columns:

*# mengganti outlier dengan mean di setiap kolom*

    colReplacedOutliers = replaceOutlier(outputVar[col])

*# menampung hasilnya*

    outDataReplacedOutliers[col] = colReplacedOutliers

*# menampilkan hasil penanganan outlier*

print(outDataReplacedOutliers)

*# NORMALISASI DATA*

print("==============NORMALISASI===============")

*# variabel input*

*# menggunakan min-max*

minMaxNormIn = (inputVar - inputVar.min())/(inputVar.max()-inputVar.min())

print("Menampilkan variabel input yang sudah di normalisasi menggunakan nilai min-max")

print(minMaxNormIn)

*# menggunakan z-score*

zScoreIn = (inputVar - inputVar.mean())/(inputVar.std())

print("Menampilkan variabel input yang sudah di normalisasi menggunakan z-score")

print(zScoreIn)

*# menggunakan function dari scikit learn*

minMaxScaler = preprocessing.MinMaxScaler()

inScaledData = minMaxScaler.fit\_transform(inputVar)

inNormalized = pd.DataFrame(inScaledData)

print("Variabel input yang sudah di normalisasi:")

print(inNormalized)

*# variabel output*

*# menggunakan min-max*

minMaxNormOut = (outputVar-outputVar.min())/(outputVar.max()-outputVar.min())

print("Menampilkan variabel output yang sudah di normalisasi menggunakan nilai min-max")

print(minMaxNormOut)

*# menggunakan z-score*

zScoreOut = (outputVar - outputVar.mean())/(outputVar.std())

print("Menampilkan variabel input yang sudah di normalisasi menggunakan z-score")

print(zScoreOut)

*# menggunakan function dari scikit learn*

minMaxScaler = preprocessing.MinMaxScaler()

outScaledData = minMaxScaler.fit\_transform(outputVar)

outNormalized = pd.DataFrame(outScaledData)

print("Variabel output yang sudah di normalisasi:")

print(outNormalized)