 **Naïve Bayes K-NN Simple Regresi Linier**



**Naïve Bayes**

import pandas as pd

import numpy as np

import os

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score,precision\_score,recall\_score, f1\_score, classification\_report

df=pd.read\_excel('respon.xlsx','nb')

print(df)

print()

le = LabelEncoder()

print(f'field Kecepatan Angin SEBELUM dilakukan label encoder\n {df["Kecepatan Angin"]}')

print()

df["Kecepatan Angin"]=le.fit\_transform(df["Kecepatan Angin"])

df["Suhu"]=le.fit\_transform(df["Suhu"])

df["Jenis Cuaca"]=le.fit\_transform(df["Jenis Cuaca"])

print(f'field Kecepatan Angin SETELAH dilakukan label encoder\n {df["Kecepatan Angin"]}')

print()

print(f'SEMUA field SETELAH dilakukan label encoder\n {df}')

# X kapital bukan class label (bukan Jenis Cuaca)

X = df.iloc[:,0:2].values

Y = df.iloc[:,2:3].values

print('\n',X)

print('\n',Y)

X\_train, X\_test, y\_train, y\_test=train\_test\_split(X,df['Jenis Cuaca'],test\_size=0.2,random\_state=10)

klasifikasi=MultinomialNB().fit(X\_train,y\_train)

prediksi = klasifikasi.predict(X\_test)

print('\n',f'Tingkat akurasi : ', accuracy\_score(y\_test,prediksi))

print()

print(classification\_report(y\_test,prediksi,zero\_division=0))

print()

# rendah0, sedang1, tinggi2

# dingin0, normal1, panas2

# cerah0, hujan1, mendung2

# tinggi panas? 2,2

cuaca=[[2, 2]]

prediksi=klasifikasi.predict(cuaca)

prediksi\_proba=klasifikasi.predict\_proba(cuaca)

print(f'Hasil prediksi proba cuaca: ',prediksi\_proba)

print()

for cari in prediksi:

    if cari==2:

        print('Hasil prediksi cuaca = MENDUNG')

    elif cari==1:

        print('Hasil prediksi cuaca = HUJAN')

    else:

        print('Hasil prediksi cuaca = CERAH')

print()

print(f'Hasil prediksi cuaca: ',prediksi)

print()

**K-NN**

import pandas as pd

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, classification\_report

def k\_nn\_prediction(X\_train, y\_train, X\_test, y\_test, n\_neighbors):

    # print('-------' \* 9)

    print('\nK = ', n\_neighbors)

    knn\_classifier = KNeighborsClassifier(n\_neighbors)

    knn\_classifier.fit(X\_train, y\_train)

    # Melakukan prediksi pada data uji

    predictions = knn\_classifier.predict(X\_test)

    # Evaluasi model

    accuracy = accuracy\_score(y\_test, predictions)

    classification\_rep = classification\_report(y\_test, predictions, zero\_division=0)

    print(f'Tingkat akurasi : {accuracy}')

    print('\n', classification\_rep)

    # Contoh prediksi untuk data baru

    new\_data = [[5, 7]]

    prediction = knn\_classifier.predict(new\_data)

    # Mapping kelas hasil prediksi ke kategori aslinya

    result\_mapping = {2: 'TAMAN', 1: 'PANTAI', 0: 'GUNUNG'}

    result\_category = result\_mapping.get(prediction[0], 'Kategori tidak dikenal')

    print(f'Hasil prediksi wisata dengan K={n\_neighbors}: {result\_category}')

    print('-------' \* 9)

# Contoh penggunaan fungsi

df = pd.read\_excel('respon.xlsx', 'knn')

print(df)

print('-------' \* 9)

le = LabelEncoder()

# Menggunakan LabelEncoder untuk mengubah data kategorikal menjadi numerik

df["Jarak\_dari\_Pusat\_Kota"] = le.fit\_transform(df["Jarak\_dari\_Pusat\_Kota"])

df["Tingkat\_Kepopuleran"] = le.fit\_transform(df["Tingkat\_Kepopuleran"])

df["Jenis\_Tempat\_Wisata"] = le.fit\_transform(df["Jenis\_Tempat\_Wisata"])

X = df.iloc[:, 0:2].values

y = df["Jenis\_Tempat\_Wisata"].values

# Membagi dataset menjadi data latih dan data uji

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=10)

n\_neighbors1 = 3

n\_neighbors2 = 5

k\_nn\_prediction(X\_train, y\_train, X\_test, y\_test, n\_neighbors1)

k\_nn\_prediction(X\_train, y\_train, X\_test, y\_test, n\_neighbors2)

**Atau bisa juga diberi inputan pilihan knn setelah membagi dataset menjadi data latih**# Minta pengguna untuk memasukkan nilai n\_neighbors

valid\_values = [1, 3, 5, 7]

while True:

    try:

        n\_neighbors\_input = int(input("Masukkan nilai n\_neighbors (1, 3, 5, 7): "))

        if n\_neighbors\_input in valid\_values:

            break

        else:

            print("Masukkan nilai yang benar (1, 3, 5, atau 7).")

    except ValueError:

        print("Masukkan nilai yang benar (angka positif).")

# Panggil fungsi k\_nn\_prediction dengan nilai n\_neighbors yang dimasukkan

k\_nn\_prediction(X\_train, y\_train, X\_test, y\_test, n\_neighbors\_input)

**Simple Regresi Linier**

import pandas as pd

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error,r2\_score

import math

data=pd.read\_excel('respon.xlsx','simple-reglin')

print(data,'\n')

# ambil variabel x dan y pada data

X\_train=data.iloc[:,:1].values

print(f'variabel X : \n {X\_train}','\n')

y\_train=data.iloc[:,1].values

print(f'variabel Y : \n {y\_train}','\n')

# terapkan Linier regresi

reg=LinearRegression()

reg.fit(X\_train,y\_train)

# mencari variabel 'coefficient' $ 'Intercept'

print(f'Coefficient : {str(reg.coef\_)}','\n')

print(f'Intercept : {str(reg.intercept\_)}','\n')

pred=reg.predict(X\_train)

print(f'Hasil prediksi : \n {pred}','\n')

print('Coefficient of Determination : %.2f' % r2\_score(y\_train,pred),'\n')

print('Mean Square Error : %.2f' % mean\_squared\_error(y\_train,pred),'\n')

print('Root Mean Square Error : %.2f' % math.sqrt(mean\_squared\_error(y\_train,pred)),'\n')

# harga tiket = X = variabel memengaruhi

# jumlah penontoh = Y = variabel dipengaruhi

# Mencari nilai Y berdasarkan nilai X

cek =[[30]]

pred=reg.predict(cek)

print(f'Hasil Prediksi X{cek} = Y{pred}','\n')

# Mencari nilai X berdasarkan nilai Y

cek\_y = [750]

pred\_x = (cek\_y - reg.intercept\_) / reg.coef\_

print(f'Hasil prediksi Y{cek\_y} = X{pred\_x}', '\n')

**pip list +> melihat list**

**pip install pandas =>**

**pip install openpyxl => membaca file excel**

**pip install matplotlib => visualisasi data**

**pip install numpy => visualisasi data**

**pip install -U scikit-learn**

**pip install streamlit**

**Naïve Bayes (Streamlit) run: streamlit run namafile.py**

import streamlit as st

from PIL import Image

import pandas as pd

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score,precision\_score,recall\_score, f1\_score, classification\_report

img=Image.open('halaman.jpg')

st.image(img,width=700)

# Membuat header web

st.write('''

         # APLIKASI KELULUSAN MAHASISWA

         ## Berbasis Web Menggunakan Streamlit

''')

# streamlit run naive-bayes-2.py

# side bar

img2=Image.open('header.png')

st.sidebar.image(img2)

st.sidebar.header('Inputan atribut :')

# inputan

# nama=st.sidebar.text\_input('Masukan Nama : ',placeholder='Ketikan Nama dengan benar')

kecepatan\_angin=st.sidebar.selectbox('Pilih Kecepatan Angin',('0-Rendah','1-Sedang','2-Tinggi'),placeholder='pilih Kecepatan Angin',index=None)

suhu=st.sidebar.selectbox('Pilih Suhu',('0-Dingin','1-normal','2-Panas'),placeholder='pilih Suhu',index=None)

# Membuat tombol

tombol ='t'

if 'clicked' not in st.session\_state:

    st.session\_state.clicked=False

def click\_button():

    st.session\_state.clicked=True

st.sidebar.button('PREDIKSI',type='secondary',on\_click=click\_button)

# st.sidebar.button("Prediksi",type="primary", on\_click=click\_button)

if st.session\_state.clicked:

    tombol='y'

# algoritma naive bayes

def hitung\_naivebayes(atribut):

    df=pd.read\_excel('respon.xlsx','nb')

    le = LabelEncoder()

    df["Kecepatan Angin"]=le.fit\_transform(df["Kecepatan Angin"])

    df["Suhu"]=le.fit\_transform(df["Suhu"])

    X = df.iloc[:,0:2].values

    Y = df.iloc[:,2:3].values

    X\_train, X\_test, y\_train, y\_test=train\_test\_split(X,df['Jenis Cuaca'],test\_size=0.2,random\_state=10)

    klasifikasi=MultinomialNB().fit(X\_train,y\_train)

    # prediksi = klasifikasi.predict(X\_test)

    prediksi = klasifikasi.predict(atribut)

    st.write('nilai prediksi ',prediksi)

    if prediksi == 0:

        st.write("hasil prediksi Cuaca Cerah")

    elif prediksi == 1:

        st.write("hasil prediksi Cuaca Hujan")

    elif prediksi == 2:

        st.write("hasil prediksi Cuaca Mendung")

    nilai\_prob=klasifikasi.predict\_proba(atribut)

    st.write("#### Nilai probabilitas cuaca 0-cerah, 1-hujan, 2-mendung: ",nilai\_prob)

if tombol =='y':

    # cek tombol

    # st.write("Tombol diklik")

    # membuat data inputan dari user

    data\_input={

        'kecepatan\_angin': kecepatan\_angin,

        'suhu': suhu,

        }

    # st.write("Nama mahasiswa: ",nama)

    st.write(data\_input)

    inputan=pd.DataFrame(data\_input, index=[0])

    st.write(inputan)

    data\_input\_konversi={

        'kecepatan\_angin': int(kecepatan\_angin[0:1]),

        'suhu': int(suhu[0:1]),

        }

    st.write(data\_input\_konversi)

    atribut=pd.DataFrame(data\_input\_konversi, index=[0])

    hitung\_naivebayes(atribut)