Tugas Besar 2 IF3170 Inteligensi Buatan
Implementasi Algoritma KNN dan Naive-Bayes Untuk Membuat Model

Diajukan untuk memenuhi tugas mata kuliah IF3170 Inteligensi Buatan

Disusun oleh Kelompok :

Varraz Hazandra Abrar	13521020
Shelma Salsabila	13521115
Asyifa Nurul Shafira	13521125
Ferindya Aulia Berlianty	13521161

PROGRAM STUDI SARJANA TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2023

BAB I

Penjelasan Implementasi Program

1. Implementasi Algoritma KNN

Algoritma K-Nearest Neighbor merupakan algoritma *supervised learning* dimana hasil dari *instance* yang baru diklasifikasikan berdasarkan mayoritas dari kategori k-tetangga terdekat. Algoritma ini menggunakan *neighborhood classification* sebagai nilai prediksi dari nilai *instance* yang baru. Dalam menentukan nilai k, jika diketahui jumlah klasifikasi genap maka menggunakan nilai k ganjil, sebaliknya jika diketahui jumlah klasifikasi ganjil maka menggunakan nilai k genap.

Adapun dalam tugas ini, implementasi algoritma KNN yang dilakukan adalah sebagai berikut.

1. Pengklasifikasian data pada setiap kolom dibagi menjadi 4 klasifikasi untuk nilai numerik dengan ketentuan berikut.
    a. Klasifikasi 1, yang termasuk ke dalam klasifikasi ini adalah nilai yang berada di antara minimum sampai dengan kurang dari kuartil 1
    b. Klasifikasi 2, yang termasuk ke dalam klasifikasi ini adalah nilai yang berada di antara kuartil 1 sampai dengan kurang dari kuartil 2
    c. Klasifikasi 3, yang termasuk ke dalam klasifikasi ini adalah nilai yang berada di antara kuartil 2 sampai dengan kurang dari kuartil 3
    d. Klasifikasi 4, yang termasuk ke dalam klasifikasi ini adalah nilai yang berada di antara kuartil 3 sampai sama dengan nilai maksimum

    Proses pengklasifikasian ini dilakukan dengan memanggil kode `classifyData(data, bins1, labels1)`. Parameter data merupakan hasil pembacaan data frame terhadap suatu kolom yang nilainya ingin diklasifikasikan. Untuk parameter bins merupakan nilai batas-batas klasifikasi yang artinya disini adalah nilai minimum, nilai maksimum, kuartil 1 (Q1), kuartil 2 (Q2), dan kuartil 3 (Q3). Parameter labels adalah nama-nama klasifikasinya. Aturan secara umum yang penulis gunakan jika suatu nilai pada suatu kolom masuk ke dalam klasifikasi 1, maka nama *title* klasifikasi yang diberikan adalah nama kolom1 dan seterusnya.

2. Setelah data-data diklasifikasikan, langkah selanjutnya yaitu melakukan perhitungan *euclidean distance* untuk kolom ram dengan menggunakan fungsi **`euclideanDistance(value, value_ram)`**. Parameter value merupakan hasil dari input terhadap kolom ram, sedangkan parameter value_ram merupakan hasil dari pembacaan terhadap kolom ram.
3. Setelah dihitung *euclidean distance* pada kolom ram, selanjutnya dilakukan validasi terhadap input dengan menggunakan fungsi **`changeInputValidation(values, df)`**. Parameter values merupakan sebuah list dan parameter df merupakan dataframe. Fungsi ini melakukan perubahan terhadap nilai-nilai dalam list values berdasarkan kondisi yang didapatkan dari kolom df. Jika tipe data bukan boolean dan bukan kolom ram atau price_range, maka values akan diubah menjadi string.
4. Langkah selanjutnya yaitu membuat model dengan menggunakan fungsi **`makeArrayModel(df)`**. Parameter df adalah dataframe. Fungsi ini akan mengonversi setiap baris data dalam df menjadi array, kemudian mengembalikan array yang berisi semua baris dari df.
5. Setelah membuat model, langkah selanjutnya yaitu membandingkan nilai-nilai antara array yang diberikan dengan sebuah value yang dilakukan oleh fungsi **`countDifferent(arrays, value)`**. Fungsi ini akan mengembalikan tiga list yaitu counts, ram_differents, dan price_ranges.
6. Selanjutnya, dilakukan penulisan data ke dalam sebuah file di lokasi yang ditentukan oleh file_path yang dilakukan oleh fungsi **`writeModelKNN(array1, array2, array3, file_path)`**. Fungsi ini akan membuat sebuah file teks dengan tiga kolom yang memuat data dari tiga array yang diberikan.

   Adapun untuk kode secara lebih rinci adalah sebagai berikut,

```
import pandas as pd
import math
import numpy as np

# Read the data
def readFileWithoutOutliers(file_path):
    df = pd.read_csv(file_path)
    boolean_column = ['blue', 'dual_sim', 'four_g',
'three_g', 'touch_screen', 'wifi']
```

```
    for column in boolean_column:
        df[column] = df[column].astype(bool)
    Q1 = df['fc'].quantile(0.25)
    Q3 = df['fc'].quantile(0.75)
    IQR = Q3 -Q1
    upper_bound = Q3 + 1.5 * IQR
    lower_bound = Q1 - 1.5 * IQR

    df_no_outliers = df[(df['fc'] > lower_bound) &
(df['fc'] < upper_bound)]
    return df_no_outliers

def classifyData(data, bins1, labels1):
    classified_data = pd.cut(data, bins=bins1,
labels=[str(label) for label in labels1])
    return classified_data
def changeBecameClassifyData(file_path):
    df = readFileWithoutOutliers(file_path)
    df["battery_power"] = classifyData(df["battery_power"],
[500, 864, 1218, 1600, 1998], ['battery_power1',
'battery_power2', 'battery_power3', 'battery_power4'])
    df["clock_speed"] = classifyData(df["clock_speed"],
[0.4, 0.7, 1.5, 2.2, 3], ["clock_speed1", "clock_speed2",
"clock_speed3", "clock_speed4"])
    df["fc"] = classifyData(df["fc"], [-0.1, 1, 3, 7, 15],
["fc1", "fc2", "fc3", "fc4"])
    df["int_memory"] = classifyData(df["int_memory"], [1.9,
16, 32, 48, 64], ["int_memory1", "int_memory2",
"int_memory3", "int_memory4"])
    df["m_dep"] = classifyData(df["m_dep"], [0, 0.2, 0.5,
0.8, 1], ["m_dep1", "m_dep2", "m_dep3", "m_dep4"])
    df["mobile_wt"] = classifyData(df["mobile_wt"], [79,
107.5, 139, 169, 200], ["mobile_wt1", "mobile_wt2",
"mobile_wt3", "mobile_wt4"])
    df["n_cores"] = classifyData(df["n_cores"], [0.9, 2, 4,
7, 8], ["n_cores1", "n_cores2", "n_cores3", "n_cores4"])
    df["pc"] = classifyData(df["pc"], [-0.1, 5, 10, 15,
20], ["pc1", "pc2", "pc3", "pc4"])
    df["px_height"] = classifyData(df["px_height"], [-0.1,
273, 560, 946, 1960], ["px_height1", "px_height2",
"px_height3", "px_height4"])
    df["px_width"] = classifyData(df["px_width"], [499.5,
878, 1247, 1623, 1998], ["px_width1", "px_width2",
"px_width3", "px_width4"])
    df["sc_h"] = classifyData(df["sc_h"], [4.9, 9, 12, 16,
19], ["sc_h1", "sc_h2", "sc_h3", "sc_h4"])
    df["sc_w"] = classifyData(df["sc_w"], [-0.1, 2, 5, 9,
18], ["sc_w1", "sc_w2", "sc_w3", "sc_w4"])
    df["talk_time"] = classifyData(df["talk_time"], [1.9,
6, 11, 16, 20], ["talk_time1", "talk_time2", "talk_time3",
```

```
"talk_time4"])
    return df
def classify(namaKolom, value, df):
    if df[namaKolom].dtype != bool:
        Q1 = df[namaKolom].min()
        Q2 = df[namaKolom].quantile(0.25)
        Q3 = df[namaKolom].quantile(0.5)
        Q4 = df[namaKolom].quantile(0.75)
        Q5 = df[namaKolom].max()
        if(value >= Q1 and value <= Q2):
            return 1
        elif(value > Q2 and value <= Q3):
            return 2
        elif(value > Q3 and value <= Q4):
            return 3
        elif(value > Q4 and value <= Q5):
            return 4
    else:
        if value == True:
            return 1
        else:
            return 2

#Hanya untuk kolom RAM
def euclideanDistance(value, value_ram):
    hasil = math.sqrt((value_ram - value)**2)
    return hasil
def changeInputValidation(values, df):
    i = 0
    for kolom in df.columns:
        if kolom != "ram" and df[kolom].dtype != bool and
kolom != "price_range":
            values[i] = f'{kolom}{classify(kolom,
values[i], df)}'
        elif df[kolom].dtype == bool:
            if values[i] == 0:
                values[i] = False
            else:
                values[i] = True

        i += 1
    return values
def makeArrayModel(df):
    array_hasil = []
    for index, row in df.iterrows():
        array_hasil.append(row.values)
    array_hasil = np.array(array_hasil)
    return array_hasil
def countDifferent(arrays, value):
    counts = []
    ram_differents = []
```

```
    price_ranges = []
    count = 0
    ram_different = 0
    for array in arrays:
        for i in range(len(array)-1):
            if i != 13:
                if(array[i] != value[i]):
                    count += 1
            else:
                ram_different = euclideanDistance(array[i],
value[i])
        counts.append(count)
        ram_differents.append(ram_different)
        count = 0
        ram_different = 0
        price_ranges.append(array[len(array)-1])
    return counts, ram_differents, price_ranges

def writeModelKNN(array1, array2, array3, file_path):
    data_list = list(zip(array1, array2, array3))
    with open(file_path, 'w') as file:
        file.write("Count\tRAM_Difference\tPrice_Range\n")
        for data in data_list:
            file.write('\t'.join(map(str, data)) + '\n')
def makeDataframeSorted(file_name, pd, kolom1, kolom2, k):
    data_list = []
    with open(file_name, 'r') as file:
        lines = file.readlines()
    for line in lines[1:]:
        values = line.strip().split()
        data_dict = {
            'Count': int(values[0]),
            'RAM_Difference': float(values[1]),
            'Price_Range': int(values[2])
        }
        data_list.append(data_dict)
    data = pd.DataFrame(data_list)
    sorted_data = data.sort_values(by=[kolom1, kolom2])[:k]
    return sorted_data

def finalResultKNN(values):
    df = readFileWithoutOutliers("../data/data_train.csv")
    df1 =
changeBecameClassifyData("../data/data_train.csv")
    arrays = makeArrayModel(df1)
    input = changeInputValidation(values, df)
    count, ram, price_ranges = countDifferent(arrays,
input)
    writeModelKNN(count, ram, price_ranges, "hasilknn.txt")
    sorted_data = makeDataframeSorted("hasilknn.txt", pd,
"RAM_Difference", "Count", 28)
```

```
    return sorted_data["Price_Range"]

print(finalResultKNN([775,0,1.0,0,3,0,46,0.7,159,2,16,862,1
864,568,17,15,11,1,1,1]))
```

```
df2 =
readFileWithoutOutliers("../data/data_validation.csv")
df3 =
readFileWithoutOutliers("../data/data_validation.csv")
df2 = df2.drop(columns=['price_range'])
results = []
numpy_array = df2.values
arrays = makeArrayModel(df)
for array in numpy_array:
    input = changeInputValidation(array, df)
    count, ram, price_ranges = countDifferent(arrays,
input)
    writeModelKNN(count, ram, price_ranges, "hasilknn.txt")
    sorted_data = makeDataframeSorted("hasilknn.txt", pd,
"RAM_Difference", "Count", 30)
    x = sorted_data["Price_Range"].mode().iloc[0]
    results.append(x)
print("Hello")
print(results)
price_range_comparison = results ==
df3['price_range'].to_numpy()

count = 0
for array in price_range_comparison:
    if(array == True):
        count += 1
print(count)
print(count/len(price_range_comparison))
```

2. Implementasi Algoritma Naive-Bayes

Algoritma Naive Bayes merupakan salah satu algoritma yang terdapat pada teknik klasifikasi (Classification). Algoritma ini merupakan pengklasifikasian dengan metode probabilitas dan statistik yang ditemukan oleh Thomas Bayes, yaitu dengan memprediksi peluang di masa depan berdasarkan pengalaman di masa sebelumnya. Klasifikasi Naive Bayes diasumsikan bahwa ada atau tidak ciri tertentu dari sebuah class tidak ada hubunganya dengan ciri dari class lainnya.

Adapun pada tugas ini implementasi Naive Bayes yang dilakukan adalah sebagai berikut.

1. Pengklasifikasian setiap data di setiap kolom menjadi 4 klasifikasi untuk nilai numerik dengan ketentuan berikut.
    a. Klasifikasi 1, yang termasuk kedalam klasifikasi ini adalah nilai diantara minimum sampai dengan kurang dari kuartil 1
    b. Klasifikasi 2, yang termasuk kedalam klasifikasi ini adalah nilai diantara kuartil 1 sampai dengan kurang dari kuartil 2
    c. Klasifikasi 3, yang termasuk kedalam klasifikasi ini adalah nilai diantara kuartil 2 sampai dengan kurang dari kuartil 3
    d. Klasifikasi 4, yang termasuk kedalam klasifikasi ini adalah nilai diantara kuartil 3 sampai dengan sama dengan nilai maksimum

    Proses pengklasifikasian ini dilakukan dengan memanggil kode **`classifyData(data, bins1, labels1).`** Parameter data adalah hasil pembacaan data frame terhadap suatu kolom yang nilainya ingin diklasifikasikan. Kemudian bins adalah nilai batas-batas klasifikasi artinya disini adalah nilai minimum, maksimum, Q1, Q2, dan Q3. Dan labels adalah nama-nama klasifikasinya. Aturan general yang penulis gunakan jika suatu nilai pada suatu kolom masuk kedalam klasifikasi 1 maka nama *title* klasifikasi yang diberikan adalah nama kolom1 dan seterusnya.
2. Setelah data-data diklasifikasikan maka selanjutnya adalah membuat model. Proses pembuatan model ini ditulis kedalam file .txt dan dibagi menjadi 3 bagian. Pada bagian atas ditulis model untuk data numeric, kemudian model untuk tipe data non numeric dan model untuk nilai kolom target. Pembuatannya memanggil 3 fungsi yaitu
    a. Fungsi **`writeMultipleClassifyDataToTxt(file_path, kolom_klasifikasi,data_frame)`**

        Fungsi ini digunakan untuk membuat model kolom numeric. Fungsi ini memiliki 3 parameter yakni, file_path adalah alamat dari model dalam bentuk .txt akan disimpan. Adapun parameter kedua adalah kolom_klasifikasi. Kolom klasifikasi ini berbentuk array yang di

dalamnya terkandung nama kolom, Kemudian bins adalah nilai batas-batas klasifikasi artinya disini adalah nilai minimum, maksimum, Q1, Q2, dan Q3. Dan labels adalah nama-nama klasifikasinya. Adapun parameter yang terakhir adalah data_frame yaitu hasil proses read file csv. Fungsi ini memanggil **`countClassifyData(namaKolomClassify, data,bins,labels),`** intinya fungsi ini memanggil fungsi untuk klasifikasi. Kemudian nanti setiap nilai klasifikasi ini akan dihitung jumlahnya berdasarkan keterhubungannya dengan kolom target serta dibagi dengan banyaknya nilai kolom target. Agar mempermudah sebagai contoh sebagai berikut. Misalkan kolom battery-power dia akan diklasifikasikan kedalam battery-power1, battery-power2, battery-power3, battery-power4. Kemudian misalnya kolom klasifikasi yang diambil adalah battery-power1 maka dia akan dihitung jumlah battery-power1 yang nilai kolom targetnya adalah 0 kemudian dibagi dengan jumlah kolom target 0 dan seterusnya dilakukan terhadap semua kolom target.

b. Fungsi **`writeMultipleClassifyDataBooleanToTxt(file_path, kolom_klasifikasi,data_frame)`**

Fungsi ini digunakan untuk membuat model kolom numeric. Fungsi ini memiliki 3 parameter yakni, file_path adalah alamat dari model dalam bentuk .txt akan disimpan. Adapun parameter kedua adalah kolom_klasifikasi. Kolom klasifikasi ini berbentuk array yang di dalamnya terkandung nama kolom, dan labels adalah nama-nama klasifikasinya. Adapun parameter yang terakhir adalah data_frame yaitu hasil proses read file csv. Fungsi ini memanggil fungsi **`countClassifyDataBoolean(namaKolomClassify,data),`** intinya fungsi ini memanggil fungsi untuk klasifikasi. Kemudian nanti setiap nilai klasifikasi ini akan dihitung jumlahnya berdasarkan keterhubungannya dengan kolom target serta dibagi dengan banyaknya nilai kolom target. Seperti layaknya kolom numeric perbedaanya adalah pada model ini hanya ada 2 klasifikasi yaitu untuk nilai True dan False. Jadi tidak ada fungsi klasifikasi layaknya kolom numeric.

c. Fungsi **`writePvj(file_path, namaKolom, data_frame, arrayNilai)`**

Fungsi ini digunakan untuk membuat model kolom target. Fungsi ini memiliki 3 parameter yakni, file_path adalah alamat dari model dalam bentuk .txt akan disimpan. Adapun parameter kedua adalah namaKolom disini namaKolom yang digunakan adalah price_range. Data_frame hasil pembacaan file dari csv dan arrayNilai merupakan nilai-nilai dari kolom target itu. Pada fungsi ini nanti akan dituliskan modelnya dengan mencari jumlah nilai itu di kolom target serta membaginya dengan semua jumlah data yang ada di kolom target.

3. Fungsi klasifikasi untuk data inputan

Seperti yang telah diketahui bahwa input yang akan diberikan adalah kolom-kolom yang berhubungan dengan price_range kemudian nanti akan diprediksi price_range nya berapa. Namun, seperti yang sudah dijelaskan sebelumnya bahwa kita mengklasifikasikan datanya. Sehingga data yang baru yang mau diprediksi price_range nya juga harus diklasifikasikan. Pada kode program dapat diklasifikasikan dengan menggunakan fungsi **`classify(namaKolom,value,df).`** Value disini bentuknya array yang berisi nilai-nilai setiap kolom kecuali price_range. Kemudian nanti setiap nilai di array value akan dimasukan ke dalam suatu klasifikasi.

4. Fungsi main

Nantinya setelah ketemu setiap kolom klasifikasinya berapa nilai probabilitasnya akan dicari di model. Setelah itu dikalikan dengan nilai probabailitas target. Untuk semua target dihitung kemudian dicari yang paling besar dan jawabannya adalah masuk kedalam price_range itu. Adapun fungsi main adalah **`naiveBayesMain(df,target,values,lokasiFile, targetColumn).`** Fungsi ini memanggil dua fungsi lain yaitu, **`getValue(target,lokasiFile,classify_result)`** mencari probabilitas di model yang bukan kolom target. Fungsi kedua yang dipanggil adalah, **`getValueTarget(target,lokasiFile)`** untuk mencari probabilitas di model yang merupakan kolom target.

Adapun untuk kode secara lebih rinci adalah sebagai berikut,

1. Kode untuk membuat model

```
import pandas as pd

#Fungsi-fungsi
def readFileWithoutOutliers():
    df = pd.read_csv("../data/data_train.csv")
    kolom_boolean = ['blue', 'dual_sim', 'four_g',
'three_g', 'touch_screen', 'wifi']
    for kolom in kolom_boolean:
        df[kolom] = df[kolom].astype(bool)
    Q1 = df['fc'].quantile(0.25)
    Q3 = df['fc'].quantile(0.75)
    IQR = Q3 - Q1
    upper_bound = Q3 + 1.5 * IQR
    lower_bound = Q1 - 1.5 * IQR
    df_no_outliers = df[(df['fc'] > lower_bound) &
(df['fc'] < upper_bound)]
    return df_no_outliers

def classifyData(data, bins1, labels1):
    classified_data = pd.cut(data, bins=bins1,
labels=[str(label) for label in labels1])
    return classified_data
def countClassifyData(namaKolomClassify, data, bins,
labels):
    df[namaKolomClassify] =  classifyData(data, bins,
labels)
    result0 = df[df['price_range'] ==
0][namaKolomClassify].value_counts(sort=False)
    result1 = df[df['price_range'] ==
1][namaKolomClassify].value_counts(sort=False)
    result2 = df[df['price_range'] ==
2][namaKolomClassify].value_counts(sort=False)
    result3 = df[df['price_range'] ==
3][namaKolomClassify].value_counts(sort=False)
    result0 = result0/352
```

```
    result1 = result1/344
    result2 = result2/341
    result3 = result3/334
    return result0, result1, result2, result3

def countClassifyDataBoolean(namaKolomClassify, data):
    df[namaKolomClassify] = data.replace({True, False})
    result0 = df[df['price_range'] ==
0][namaKolomClassify].value_counts(sort=False)
    result1 = df[df['price_range'] ==
1][namaKolomClassify].value_counts(sort=False)
    result2 = df[df['price_range'] ==
2][namaKolomClassify].value_counts(sort=False)
    result3 = df[df['price_range'] ==
3][namaKolomClassify].value_counts(sort=False)
    result0 = result0/352
    result1 = result1/344
    result2 = result2/341
    result3 = result3/334
    return result0, result1, result2, result3


def writeMultipleClassifyDataToTxt(file_path,
kolom_klasifikasi, data_frame):
    with open(file_path, 'w') as file:
        for namaKolomClassify, bins, labels in
kolom_klasifikasi:
            data = data_frame[namaKolomClassify]
            df[namaKolomClassify] = classifyData(data,
bins, labels)
            result0, result1, result2, result3 =
countClassifyData(f"{namaKolomClassify}_klasifikasi",
data, bins, labels)
            file.write("\t".join([labels[0]] +
list(map(str, result0.sort_index().values))) + "\n")
            file.write("\t".join([labels[1]] +
list(map(str, result1.sort_index().values))) + "\n")
            file.write("\t".join([labels[2]] +
list(map(str, result2.sort_index().values))) + "\n")
            file.write("\t".join([labels[3]] +
list(map(str, result3.sort_index().values))) + "\n")


def writeMultipleClassifyDataBooleanToTxt(file_path,
```

```
kolom_klasifikasi, data_frame):
    with open(file_path, 'a') as file:
        for namaKolomClassify, labels in
kolom_klasifikasi:
            data = data_frame[namaKolomClassify]
            result0, result1, result2, result3 =
countClassifyDataBoolean(namaKolomClassify, data)
            file.write("\t".join([labels[0]] +
list(map(str, result0.sort_index().values))) + "\n")
            file.write("\t".join([labels[1]] +
list(map(str, result1.sort_index().values))) + "\n")
            file.write("\t".join([labels[2]] +
list(map(str, result2.sort_index().values))) + "\n")
            file.write("\t".join([labels[3]] +
list(map(str, result3.sort_index().values))) + "\n")

#write Pvj
def writePvj(file_path, namaKolom, data_frame,
arrayNilai):
    i = 1
    with open(file_path, 'a') as file:
        for array in arrayNilai:
            count_array =
len(data_frame[data_frame[namaKolom] == array])/len(df)

file.write(f'{namaKolom}{i}\t{count_array}\n')
            i+=1

df = readFileWithoutOutliers()
kolom_klasifikasi = [
    ('battery_power', [501, 864, 1218, 1600, 1998],
['battery_power1', 'battery_power2', 'battery_power3',
'battery_power4']),
    ("clock_speed", [0.5, 0.7, 1.5, 2.2, 3],
["clock_speed1", "clock_speed2", "clock_speed3",
"clock_speed4"]),
    ("fc", [0, 1, 3, 7, 15], ["fc1", "fc2", "fc3",
"fc4"]),
    ("int_memory", [2, 16, 32, 48, 64], ["int_memory1",
"int_memory2", "int_memory3", "int_memory4"]),
    ("m_dep", [0.1, 0.2, 0.5, 0.8, 1], ["m_dep1",
"m_dep2", "m_dep3", "m_dep4"]),
    ("mobile_wt", [80, 107.5, 139, 169, 200],
["mobile_wt1", "mobile_wt2", "mobile_wt3",
```

```
"mobile_wt4"]),
    ("n_cores", [1, 2, 4, 7, 8], ["n_cores1", "n_cores2",
"n_cores3", "n_cores4"]),
    ("pc", [0, 5, 10, 15, 20], ["pc1", "pc2", "pc3",
"pc4"]),
    ("px_height", [0, 273, 560, 946, 1960],
["px_height1", "px_height2", "px_height3",
"px_height4"]),
    ("px_width", [500, 878, 1247, 1623, 1998],
["px_width1", "px_width2", "px_width3", "px_width4"]),
    ("ram", [256, 1210.5, 2107, 3036.5, 3998], ["ram1",
"ram2", "ram3", "ram4"]),
    ("sc_h", [5, 9, 12, 16, 19], ["sc_h1", "sc_h2",
"sc_h3", "sc_h4"]),
    ("sc_w", [0, 2, 5, 9, 18], ["sc_w1", "sc_w2",
"sc_w3", "sc_w4"]),
    ("talk_time", [2, 6, 11, 16, 20], ["talk_time1",
"talk_time2", "talk_time3", "talk_time4"])

]
kolom_klasifikasi_boolean = [
    ('blue', ['blue1', 'blue2', 'blue3', 'blue4']),
    ("dual_sim", ['dual_sim1', 'dual_sim2', 'dual_sim3',
'dual_sim4']),
    ("four_g", ['four_g1', 'four_g2', 'four_g3',
'four_g4']),
    ("three_g", ['three_g1', 'three_g2', 'three_g3',
'three_g4']),
    ("touch_screen", ['touch_screen1', 'touch_screen2',
'touch_screen3', 'touch_screen4']),
    ("wifi", ['wifi1', 'wifi2', 'wifi3', 'wifi4'])
]
writeMultipleClassifyDataToTxt('hasil.txt',
kolom_klasifikasi, df)
writeMultipleClassifyDataBooleanToTxt('hasil.txt',
kolom_klasifikasi_boolean, df)
writePvj('hasil.txt', 'price_range', df, [0,1,2,3])
```

2. Kode untuk menentukan prediksi

```
df1 = readFileWithoutOutliers()
def classify(namaKolom, value, df):
    if df[namaKolom].dtype != bool:
```

```
        Q1 = df[namaKolom].min()
        Q2 = df[namaKolom].quantile(0.25)
        Q3 = df[namaKolom].quantile(0.5)
        Q4 = df[namaKolom].quantile(0.75)
        Q5 = df[namaKolom].max()
        if(value >= Q1 and value < Q2):
            return 1
        elif(value >= Q2 and value < Q3):
            return 2
        elif(value >= Q3 and value < Q4):
            return 3
        elif(value >= Q4 and value <= Q5):
            return 4
    else:
        if value == True:
            return 1
        else:
            return 2

def getValue(target, lokasiFile, classify_result):
    file_path = lokasiFile
    result = []

    with open(file_path, 'r') as file:
        for line in file:
            tokens = line.strip().split('\t')
            classify_name = tokens[0]

            if classify_name == target:
                # Menambahkan nilai ke dalam list
                result = [float(value) for value in
tokens[1:]]
    if classify_result:  # Memastikan classify_result
tidak kosong
        index = int(classify_result) - 1  # Karena indeks
dimulai dari 0
        return result[index] if 0 <= index < len(result)
else None
def getValueTarget(target, lokasiFile):
    battery_power1_value = None
    with open(lokasiFile, 'r') as file:
        for line in file:
            tokens = line.strip().split('\t')
            feature_name = tokens[0]
```

```
            if feature_name == target:
                battery_power1_value = float(tokens[1])
                Break

def naiveBayesMain(df, target, values, lokasiFile,
targetColumn):
    hasil = 1  # Mulai dengan 1 agar perkalian berjalan
dengan benar
    i = 0

    for kolom in df.columns:
        if kolom != targetColumn:
            classify_result = classify(kolom, values[i],
df)
            value = getValue(f'{kolom}{target}',
lokasiFile, classify_result)

            # Pastikan value tidak None sebelum dikalikan
            if value is not None:
                hasil *= value
                i += 1
            else:
                print(f"Warning: Nilai untuk kolom
{kolom} tidak ditemukan.")
        else:
            value = getValueTarget(f'{kolom}{target}',
lokasiFile)
            if value is not None:
                hasil *= value
    # Tambahkan penanganan jika hasil masih 1
    if hasil == 1:
        print("Warning: Semua nilai kolom tidak
ditemukan. Mohon periksa input dan data.")

    return hasil

def result(hasil1, hasil2, hasil3, hasil4):
    if hasil1 > hasil2 and hasil1 > hasil3 and hasil1 >
hasil4:
        return "biaya rendah"
    elif hasil2 > hasil1 and hasil2 > hasil3 and hasil2 >
hasil4:
        return "biaya sedang"
    elif hasil3 > hasil2 and hasil3 > hasil2 and hasil3 >
```

```
hasil4:
        return "biaya tinggi"
    elif hasil4 > hasil1 and hasil4 > hasil2 and hasil4 >
hasil3:
        return "biaya sangat tinggi"
values =
[1866,0,1.4,0,0,0,30,0.5,182,3,0,108,1781,3834,16,11,8,0,
0,0]
hasil1 = naiveBayesMain(df1, 1, values, "hasil.txt",
"price_range")
hasil2 = naiveBayesMain(df1, 2, values, "hasil.txt",
"price_range")
hasil3 = naiveBayesMain(df1, 3, values, "hasil.txt",
"price_range")
hasil4 = naiveBayesMain(df1, 4, values, "hasil.txt",
"price_range")
print(result(hasil1,hasil2,hasil3,hasil4))
```

3. Implementasi dengan menggunakan *scikit learn*

Scikit Learn merupakan *library* yang menyediakan beragam algoritma dalam *machine learning*. Scikit Learn dapat digunakan untuk pengklasifikasian data ke dalam kelompok dan memprediksi nilai/label pada kolom target di suatu dataset. Berikut langkah-langkah implementasi dengan menggunakan Scikit Learn.

a. Implementasi Scikit Learn pada Naive Bayes

1. Pembacaan dataset dan memisahkan kolom target dan fitur-fitur dari dataset. Dataset yang digunakan adalah data_train.csv dengan kolom target “price_range”. Fitur-fitur yang dimaksud berupa kolom-kolom pada dataset kecuali kolom target. Fitur digunakan untuk memprediksi nilai pada kolom target.
2. *Split* dataset menjadi data latih dan data uji. Data latih pada dataset yang digunakan adalah “X_train” dan “y_train”. Data ujinya adalah “X_test”. Ada Data latih digunakan untuk melatih model, sedangkan data uji digunakan untuk mengevaluasi model.
3. Mendefinisikan *preprocessor* dengan menggunakan fungsi **`ColumnTransformer()`** untuk melakukan *preprocessing* pada data numerik.

4. Mendefinisikan *pipeline* menggunakan **`pipeline()`**. *Pipeline* terdiri dari dua tahap, yaitu melakukan *preprocessing* dengan *preprocessor* yang sudah didefinisikan sebelumnya dan klasifikasi menggunakan fungsi **`GaussianNB()`** yang diambil dari *library* "sklearn" yang merupakan Scikit Learn itu sendiri.
5. Melatih model dengan menggunakan fungsi **`fit(X_train, y_train)`**.
6. Membuat data frame (x_pred) untuk melakukan prediksi dengan fungsi **`predict(x_pred).`** Fungsi predict berasal dari *pipeline*. Hasil prediksi disimpan dalam array "predicted_price_range". Mengganti nilai pada kolom "price_range" di dataset "x_pred" dengan hasil prediksi "predicted_price_range". Semua langkah tersebut dilakukan di dalam fungsi **`finalValueKNNSKICITNaive(values, Pipeline=Pipeline)`**.
7. Memberikan hasil prediksi dari dataset yang diberikan dengan fungsi **`finalResultSkicitNaive(values)`** yang memanggil fungsi **`finalValueKNNSKICITNaive(values, Pipeline=Pipeline)`**.

Berikut kode dari implementasi Naive Bayes dengan Scikit Learn.

1. Impor *library* yang diperlukan, pembacaan dataset, dan pemisahan label dan fitur.

```
import pandas as pd
import numpy as np
from jcopml.pipeline import num_pipe, cat_pipe
from jcopml.plot import plot_confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.naive_bayes import GaussianNB

df = pd.read_csv('../data/data_train.csv')
X = df.drop(columns=['price_range'])
y = df['price_range']
```

2. Membagi dataset menjadi data uji dan data latih.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
```

```
random_state=42)
X_train.shape, X_test.shape,y_train.shape, y_test.shape
```

3. Mendefinisikan *preprocessor*.

```
preprocessor = ColumnTransformer([
    ('numeric', num_pipe(), ['battery_power', 'clock_speed', 'fc',
'int_memory', 'm_dep', 'mobile_wt', 'n_cores', 'pc', 'px_height',
'px_width', 'ram', 'sc_h', 'sc_w', 'talk_time', 'three_g',
'touch_screen', 'wifi']),
])
```

4. Mendefinisikan *pipeline*.

```
Pipeline = Pipeline([
    ('prep', preprocessor),
    ('algo', GaussianNB())
])
```

5. Melatih model.

```
Pipeline.fit(X_train, y_train)
```

6. Membuat data frame dan melakukan prediksi

```
def finalValueKNNSKICITNaive(values, Pipeline=Pipeline):
    columns = ['battery_power', 'blue', 'clock_speed', 'dual_sim', 'fc',
'four_g', 'int_memory', 'm_dep', 'mobile_wt',
               'n_cores', 'pc', 'px_height', 'px_width', 'ram', 'sc_h',
'sc_w', 'talk_time', 'three_g', 'touch_screen',
               'wifi']

    X_pred = pd.DataFrame([values], columns=columns)

    predicted_price_range = Pipeline.predict(X_pred)
```

```
    X_pred['price_range'] = predicted_price_range
    return (X_pred["price_range"][0])
```

7. Hasil prediksi dengan data train

```
def finalResultSkicitNaive(values):
    df = pd.read_csv('../data/data_train.csv')
    x = finalValueKNNSKICITNaive(values, Pipeline=Pipeline)
    return x
```

b. Implementasi KNN dengan scikit Learn

1. Pembacaan dataset dan memisahkan kolom target dan fitur-fitur dari dataset. Dataset yang digunakan adalah data_train.csv dengan kolom target "price_range". Fitur-fitur yang dimaksud berupa kolom-kolom pada dataset kecuali kolom target. Fitur digunakan untuk memprediksi nilai pada kolom target.
2. *Split* dataset menjadi data latih dan data uji. Data latih pada dataset yang digunakan adalah X_train dan y_train. Data ujinya adalah X_test dan y_test. Data latih digunakan untuk melatih model, sedangkan data uji digunakan untuk mengevaluasi model.
3. Mendefinisikan *preprocessor* dengan menggunakan fungsi **ColumnTransformer()** untuk melakukan *preprocessing* pada data numerik.
4. Mendefinisikan *pipeline* menggunakan **pipeline()**. *Pipeline* terdiri dari dua tahap, yaitu melakukan *preprocessing* dengan *preprocessor* yang sudah didefinisikan sebelumnya dan melakukan klasifikasi dengan fungsi **KNeighborClassifier()** yang diambil dari *library* "sklearn" yang merupakan Scikit Learn itu sendiri.
5. Melatih model dengan menggunakan fungsi **fit(X_train, y_train)**.
6. Membuat data frame (x_pred) untuk melakukan prediksi dengan fungsi **predict(x_pred).** Fungsi predict berasal dari *pipeline*. Hasil prediksi disimpan dalam array "predicted_price_range". Mengganti nilai pada kolom "price_range" di dataset "x_pred" dengan hasil prediksi "predicted_price_range". Semua langkah tersebut dilakukan di dalam fungsi **finalValueKNNSKICITNaive(values, Pipeline=Pipeline)**.

7. Memberikan hasil prediksi dari dataset yang diberikan dengan fungsi **finalResultSkicitNaive(values)** yang memanggil fungsi **finalValueKNNSKICITNaive(values, Pipeline=Pipeline)**.

Berikut kode dari implementasi KNN dengan Scikit Learn.

1. Import *library* yang diperlukan, pembacaan dataset, dan pemisahan label dan fitur.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from jcopml.pipeline import num_pipe, cat_pipe
from jcopml.plot import plot_confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import metrics

# Read the data
df = pd.read_csv('../data/data_train.csv')
X = df.drop(columns=['price_range'])
y = df['price_range']
```

2. Membagi dataset menjadi data uji dan data latih.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
X_train.shape, X_test.shape,y_train.shape, y_test.shape
```

3. Mendefinisikan *preprocessor*.

```
preprocessor = ColumnTransformer([
    ('numeric', num_pipe(), ['battery_power', 'clock_speed', 'fc',
'int_memory', 'm_dep', 'mobile_wt', 'n_cores', 'pc', 'px_height',
'px_width', 'ram', 'sc_h', 'sc_w', 'talk_time', 'three_g',
'touch_screen', 'wifi']),
```

```
])
```

4. Mendefinisikan *pipeline*.

```
Pipeline = Pipeline([
    ('prep', preprocessor),
    ('algo', KNeighborsClassifier())
])
```

5. Melatih model.

```
Pipeline.fit(X_train, y_train)
```

6. Membuat data frame dan melakukan prediksi.

```
# values = [563,1,0.5,1,2,1,41,0.9,145,5,6,1263,1716,2603,11,2,9,1,1,0]
def finalValueKNNSKICIT(values, Pipeline=Pipeline):
    # Membuat DataFrame dari nilai-nilai tersebut
    columns = ['battery_power', 'blue', 'clock_speed', 'dual_sim', 'fc',
'four_g', 'int_memory', 'm_dep', 'mobile_wt',
               'n_cores', 'pc', 'px_height', 'px_width', 'ram', 'sc_h',
'sc_w', 'talk_time', 'three_g', 'touch_screen',
               'wifi']

    X_pred = pd.DataFrame([values], columns=columns)

    # # Menampilkan DataFrame sebelum prediksi
    # print("DataFrame Sebelum Prediksi:")
    # # print(X_pred)

    # Melakukan prediksi menggunakan Pipeline
    predicted_price_range = Pipeline.predict(X_pred)

    # Menggantikan nilai 'price_range' dengan hasil prediksi
    X_pred['price_range'] = predicted_price_range
    return (X_pred["price_range"][0])
# print(finalValueKNNSKICIT(values))
```

7. Hasil prediksi dengan data train

```
def finalResultSkicit(values):
    df = pd.read_csv('../data/data_train.csv')
    x = finalValueKNNSKICIT(values, Pipeline=Pipeline)
    return x
```

BAB II

Test Case

A. Perbandingan algoritma KNN *scikit-learn* dengan algoritma KNN *Manuality*

Instance	*Correct Class*	*Prediction*	
1	+	-	FN
2	+	+	TP
3	+	+	TP
4	+	+	TP
5	-	+	FP
6	+	+	TP
7	+	+	TP
8	+	+	TP
9	+	+	TP
10	+	-	FN
11	+	+	TP
12	+	+	TP
13	+	+	TP

14	+	+	TP
15	+	+	TP
16	+	+	TP
17	+	+	TP
18	+	+	TP
19	-	-	TN
20	+	+	TP

Perhitungan *Precision, Recall, Accuracy*

Precision	***Recall***	***Accuracy***
Tp / Tp + Fp = 16 / 16 + 1 = 16 / 17 = 0.94117	*Tp / Tp + Fn* = 16 / 16 + 2 = 16 / 18 = 0.8888	*Tp + Tn / Tp + Fp + Tn + Fn* = 16 + 1 / 16 + 1 + 1 + 2 = 17 / 20 = 0.85

B. Perbandingan algoritma Naive-Bayes *scikit-learn* dengan algoritma Naive-Bayes *Manuality*

Instance	*Correct Class*	*Prediction*	
1	+	-	FN
2	+	+	TP
3	+	+	TP
4	+	+	TP
5	-	-	TN
6	+	+	TP
7	-	+	FP
8	+	+	TP

9	+	+	TP
10	+	-	FN
11	+	+	TP
12	+	+	TP
13	+	+	TP
14	+	+	TP
15	+	+	TP
16	+	+	TP
17	+	+	TP
18	+	+	TP
19	-	-	TN
20	+	+	TP

Perhitungan *Precision, Recall, Accuracy*

Precision	***Recall***	***Accuracy***
Tp / Tp + Fp = 15 / 15 + 1 = 15 / 16 = 0.9375	*Tp / Tp + Fn* = 15 / 15 + 2 = 15 / 17 = 0.8824	*Tp + Tn / Tp + Fp + Tn + Fn* = 15 + 2 / 15 + 1 + 2 + 2 = 17 / 20 = 0.85

Insight yang didapat adalah antara dua algoritma ini memiliki akurasi yang sama tidak sempurna karena beberapa hal. Kami telah melakukan pengujian terhadap dataset data validation dan diperoleh hanya sekitar 74% yang benar. Hal ini dis=dorong dari beberapa faktor terutama proses pengklasifikasian data.

B. *Screenshot Test Case*

```
Hallo WECOME TO MAIN
Berikan Inputan:
Masukkan nilai battery_power : 563
Masukkan nilai blue : 1
Masukkan nilai clock_speed : 0.5
Masukkan nilai dual_sim : 1
Masukkan nilai fc : 2
Masukkan nilai four_g : 1
Masukkan nilai int_memory : 41
Masukkan nilai m_dep : 0.9
Masukkan nilai mobile_wt : 145
Masukkan nilai n_cores : 5
Masukkan nilai pc : 6
Masukkan nilai px_height : 1263
Masukkan nilai px_width : 1716
Masukkan nilai ram : 2603
Masukkan nilai sc_h : 11
Masukkan nilai sc_w : 2
Masukkan nilai talk_time : 9
Masukkan nilai three_g : 1
Masukkan nilai touch_screen : 1
Masukkan nilai wifi : 0
This main will give you all of prediction:
_____NAIVE BAYES SKICIT LEARN_____
Biaya Tinggi
_____NAIVE BAYES MANUALITY_____
Biaya Tinggi
_____KNN SKICIT LEARN_____
Biaya Tinggi
_____KNN MANUALITY_____
Biaya Tinggi
```

Saran dan Kesimpulan

1. Saran

Lebih membaca dokumentasi dari bahasa yang digunakan karena ada beberapa *library* dalam *numpy* yang bisa digunakan untuk mempermudah namun karena kelalaian kami terhadap itu jadi tidak digunakan.

2. Kesimpulan

Dari tugas besar yang telah kami buat, kami telah berhasil membuat model untuk algoritma KNN dan Naive Bayes baik menggunakan pustaka maupun tidak. Namun ada beberapa kelemahan dalam model yang dibuat secara manual, yakni akurasinya kurang baik yaitu di angka 74 untuk KNN dan 77 untuk Naive Bayes. Ada beberapa hal yang melatarbelakangi ketidakakuratan ini. Yang pertama, kami tidak terlalu melakukan pembersihan data di awal, yang dilakukan hanya menghilangkan data *outliers* saja. Hal lain adalah adanya pengklasifikasian pada data yang cukup sedikit, awalnya kami membuat ini untuk mempermudah dalam proses penghitungan dan pembuatan model. Hanya saja ternyata ini menyebabkan ketidakakuratan.

Pembagian Tugas

Tugas	NIM
KNN	13521115
Naive Bayes	13521125, 13521161
Implementasi KNN dengan Scikit Learn	13521125, 13521161
Implementasi Naive Bayes dengan Scikit Learn	13521020, 13521115

Repository

https://github.com/shelmasalsa17/Tubes2_Intelegensi-_Buatan.git