Perbandingan Algoritma Naive Bayes Dan KNN Untuk Prediksi Diagnosis Arrhythmia

Sebuah Makalah
Ditujukan sebagai
tugas besar mata kuliah pengantar kecerdasan buatan

oleh:

Nama	***Nim***
1. *Khalifardy Miqdarsah*	*1304211035*
2. *M. Rizky Aulia Gobel*	*1304211002*
3. *Kevin Luzan de Fretes*	*1304211046*

PROGRAM STUDI S1 PJJ TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG

2023

Lembar Pernyatan

Nama	Tugas
1. Khalifardy Miqdarsah	Membuat Program
2. M. Rizky Aulia Gobel	Membuat laporan
3. Kevin Luzan de Fretes	Membuat presentasi dan editing video

Pernyataan:

Tim Kami mengerjakan tugas ini dengan cara yang tidak melanggar aturan perkuliahan dan kode etik akademisi. Jika melakukan plagiarism atau jenis pelanggaran lainnya, maka Tim kami bersedia diberi nilai E untuk Mata Kuliah ini.

Kata Pengantar

Alhamdulillah kami panjatkan atas segala rahmat dan cinta Allah S.W.T dan kepada kanjeng Nabi Muhammad S.A.W atas keindahan rupa dan akhlaknya juga atas cintanya sehingga pada hari ini kehidupan menjadi jauh lebih baik.

Setelah yang utama selanjutnya kami mengucapkan terima kasih kepada orang tua kami yang mengandung maupun yang tidak mengandung kami, terima kasih atas pengorbanannya, kepada guru-guru kami baik yang pernah bertemu atau yang belum bertemu, terima kasih atas kucuran air ilmunya semoga menjadi jariyah yang bisa disetorkan kepada Allah di akhirat nanti. Kepada saudara-saudara dari satu rahim maupun saudara-saudara kemanusiaan terima kasih atas semangatnya. Tak lupa juga kepada pasangan hidup kami baik yang sudah dipertemukan maupun yang masih mencari jalan untuk bertemu terima kasih atas api yang membuat kami tetap bertahan. Dan yang terakhir kepada semesta, terimakasih atas kehidupan yang indah, jogetan-jogetan yang mebuat takjub pikiran dan membuat melafazkan *rabbana ma khalqna hadza batila, ya tuhanku sungguh tiada satupun engkau ciptakan dengan sia-sia.*

Bekasi, 02 Juni 2023

Penulis

Daftar Isi

Daftar Tabel

Daftar Gambar

Bab 1

Pendahuluan

Arrhythmia adalah salah satu penyakit kelainan pada jantung. Penyakit ini menyebabkan detak jantung yang tidak normal atau tidak berarturan. Seringkali orang-orang yang mempunyai kelainan ini tidak ada gejala sebelumnya, bisa tiba-tiba pusing, nyeri dada , pusing hingga pingsan. Untuk itu perlu dilakukan pemeriksaan lebih jauh untuk diagnosis penyakit ini.

Dokter dalam mendiagnosis penyakit ini dibantu oleh alat yang namanya *Electrocardiogram* (EKG), yang mana di EKG terdapat parameter-parameter penting seperti *heart beat,* QRS, T wave , J wave dst. Parameter-parameter tersebut yang nantinya menentukan apakah seseorang memiliki *arrhytimia* atau tidak.

Dalam upaya membantu meringankan tugas dokter dalam menganalisa dan mendiagnosis apakah seorang pasien memiliki *arrhytimia* atau tidak, dirancanglah suatu penelitian metode AI yang mempunyai tingkat akurasi cukup baik dalam mendiagnosis. Hasil dari penelitian ini diharapkan nantinya bisa dijadikan dasar dalam membangun model AI yang baik untuk mendiagnosis *Arrhythmia.*

1.1. Dataset

Pada penelitian ini diberikan suatu data set yang memiliki 279 fitur ditambah satu kolom klasifikasi diagnosis. Klasifikasi berisi 16 hasil diagnosis yang bisa dilhat pada tabel 1.1. (source: *https://archive.ics.uci.edu/ml/datasets/Arrhythmia)*

Diagnosis	**code**
Normal	1
Ischemic changes (Coronary Artery Disease)	2
Old Anterior Myocardial Infarction	3
Old Inferior Myocardial Infarction	4
Sinus tachycardy	5
Sinus bradycardy	6
Ventricular Premature Contraction (PVC)	7
Supraventricular Premature Contraction	8

Left bundle branch block	9
Right bundle branch block	10
1. degree AtrioVentricular block	11
2. degree AV block	12
3. degree AV block	13
Left ventricule hypertrophy	14
Atrial Fibrillation or Flutter	15
Others	16

Table 1.1. *Tabel jenis-jenis diagnosis*

Pada dataset ini setiap hasil diagnosis dikodekan dengan bilangan integer 1-16. Untuk data-data kosong diwakili dengan "?".

Terdapat 7 fitur bertipe boolean yaitu sex, EOR_Rwave, EODD_Rwave, EORPwave, EODD_Pwave, EOR_Twave, EODD_Twave. Sisanya fitur yang bersifat float/numerik.

Untuk total record data atau baris data berjumlah 452 data sehingga dimensi keseluruhan data adalah 452 x 280.

1.2. Alasan Pemilihan metode

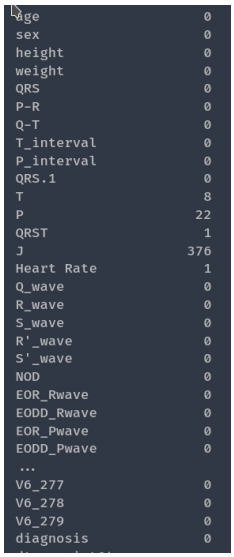
Pemilihan metode knn dan bayes pada penelitian ini di dasari pada batasan metode yang bisa dipakai 2 dari 3 metode yaitu decision tree, knn dan naive bayes. Lalu tidak terpakainya decision tree disebabkan beberapa alasan:

1. Fitur data yang begitu banyak sehingga jika memakai decision tree akan membutuhkan ruang dan waktu yang lebih banyak
2. Kebanyakan data pada data set bersifat kontinyu sedangkan pada decision tree lebih cocok untuk penggunaan data-data berbentuk diskrit
3. Outliers yang cukup banyak pada hampir semua fitur , decision tree sangat rentan terhadap outliers

Mempertimbangkan ketiga hal diataslah maka dalam penelitian ini di rancang menggunakan 2 metode yaitu KNN dan Naive Bayes.

1.3. Explaratory Data Analysys (EDA) dan preprocessing data

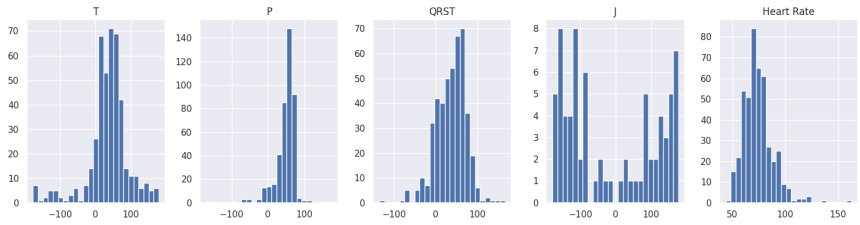
Eksplorasi data dimulai dengan mengganti value “?” dengan null supaya nantinya bisa di selidiki ada berapa kolom data yang kosong. Setelah diganti dilakukan pengecekan terdapat total 408 data kosong yang tersebar pada 5 kolom yaitu kolom T, P, QRST, J dan Heart Rate, untuk detailnya bisa dilihat pada gambar 1.1.



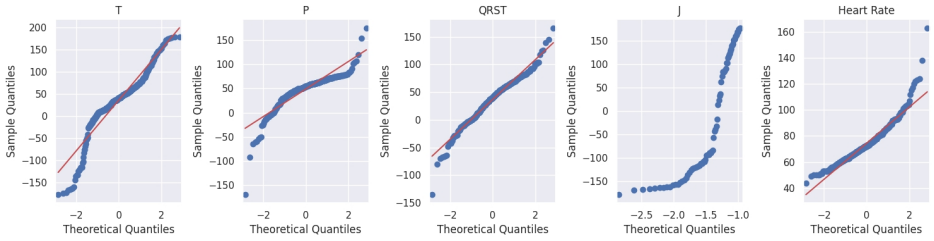
```
age                0
sex                0
height             0
weight             0
QRS                0
P-R                0
Q-T                0
T_interval         0
P_interval         0
QRS.1              0
T                  8
P                 22
QRST               1
J                376
Heart Rate         1
Q_wave             0
R_wave             0
S_wave             0
R'_wave            0
S'_wave            0
NOD                0
EOR_Rwave          0
EODD_Rwave         0
EOR_Pwave          0
EODD_Pwave         0
 ...
V6_277             0
V6_278             0
V6_279             0
diagnosis          0
```

Gambar 1.1. Detail Data Kosong

Selanjutnya untuk mengisi data kosong tersebut dilakukan ploting untuk melihat distribusi data dari ke 5 kolom tersebut apakah berdistribusi normal atau tidak, hasil ploting bisa dilihat pada gambar 1.2. dan gambar 1.3 untuk plotting qq plot



Gambar 1.2. Grafik histogram plot



Gambar 1.3. QQ Plot

Terlihat dari hasil plotingan baik dari sisi histogram dan QQplot. Dari histogram distribus dara relatif mirip dengan distribusi normal sedangkan dari QQ plot terhat relatif distribus mengikuti garis lurus(garis merah) yang artinya distribusi mendekati distribusi normal.

Karena fitur mendekati distribusi normal maka untuk mengisi kolom kosong bisa dengan beberapa cara, bisa dengan mean, median atau dengan memprediksi nilainya dengan regresi linear. Pada penelitian ini metode yang dipakai adalah metode regresi linear. Pemilihan metode pengisian dengan metode linear karaena seperti kolom J yang memiliki lebih dari 50 persen data kosong, jika diisi dengan mean atau median maka nilainya cenderung tidak representatif untuk semua data dan mengakibatkan variansinya kecil.

Secara umum regresi linear yang sederhana hanya mempunyai satu variabel independen X dan satu variabel dependen Y, lalu ada slope b atau koefesien dari X dan a yaitu intercept, secara matematis di tuliskan:

$$\mathbf{Y = a + bX}$$

dimana:

Y : Variabel Independen

X : variabe dependen

a : intercept

b : slope

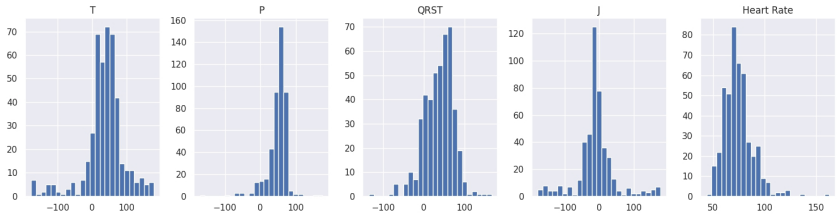
Untuk bisa memprediksi maka terlebih dahulu dicari a dan b dengan mempertimbangkan Y sebagai nilai-nilai pada setiap kolom kosong dan X nilai-nilai pada kolom yang tidak kosong yang memiliki korelasi yang kuat terhadap kolom kolom kosong nilai-nilai korelasi tersebut bisa dilihat pada gambar1.4.

	T	P	QRST	J	Heart Rate
Q-T	NaN	NaN	NaN	NaN	-0.582526
T	1.000000	NaN	NaN	NaN	NaN
P	NaN	1.000000	NaN	NaN	NaN
QRST	NaN	NaN	1.000000	NaN	NaN
J	NaN	NaN	NaN	1.000000	NaN
Heart Rate	NaN	NaN	NaN	NaN	1.000000
Amplitude_PWave	NaN	-0.545918	NaN	NaN	NaN
DII_170	NaN	NaN	NaN	0.533040	NaN
DII_179	NaN	NaN	0.562403	NaN	NaN
DIII_180	NaN	NaN	NaN	0.657752	NaN
DIII_186	NaN	0.721729	NaN	NaN	NaN
DIII_187	0.510392	NaN	NaN	NaN	NaN
DIII_188	NaN	NaN	0.555960	NaN	NaN
DIII_189	NaN	NaN	0.761026	NaN	NaN
AVL_202	NaN	-0.503591	NaN	NaN	NaN
AVL_206	NaN	-0.792279	NaN	NaN	NaN
AVL_208	NaN	-0.535638	-0.510441	NaN	NaN
AVL_209	NaN	-0.504002	-0.636423	NaN	NaN
AVF_210	NaN	NaN	NaN	0.685795	NaN
AVF_217	0.509225	NaN	NaN	NaN	NaN
AVF_218	NaN	NaN	NaN	-0.505021	NaN
AVF_219	NaN	NaN	0.614310	NaN	NaN

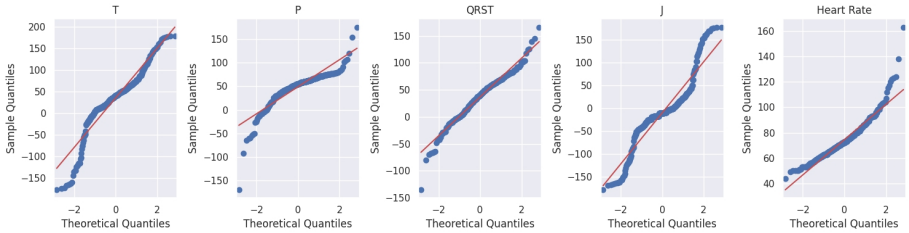
Gambar 1.4. Tabel Korelasi

Nilai korelasi berkisar diantara -1 s/d 1 , nilai negatif artinya korelasi nya berbanding terbalik artinya jika variabel independen naik maka variabel dependen akan turun dan sebaliknya, Sedangkan nilai positif artinya berbanding lurus, jika variabel independen naik maka variabel dependen pun ikut naik m begitu juga sebaliknya. Sedangkan nilai nol artinya tidak ada korelasinya sama sekali. Korelasi semakin mendekat $\pm$ 1 artinya semakin kuat korelasinya , semakin mendekati nol maka semakin tidak berkolerasi.

Berdasarkan gambar 1.4 dan pengertian dari korelasi tadi, maka untuk T berkolerasi cukup kuat dengan DIII_187, P berkolerasi kuat dengan DIII_186, QRST berkolerasi kuat dengan DIII_189, J berkolerasi kuat dengan AVF_210 dan heart_rate berkolerasi quat dengan Q-T. Maka dengan sudah ditemukannya pasangan variabel independen dari masing-masing variabel independen dilakukan regresi untuk mengisi data-data kosong tersebut, hasil plotting setelah data kosong bisa dilihat pada gambar 1.5 dan 1.6.



Gambar 1.5. Histogram Plot setelah di isi



Gambar 1.6. QQplot setelah diisi

Sesudah pengisian dari nilai-nilai kosong, selanjutnya adalah mencari kolom-kolom dengan nilai varians yang kecil dalam hal ini varians < 16. Nilai 16 dipilih karena mengikuti jumlah kelas klasifikasi yang berjumlah 16. Nilai variansi sendiri adalah suatu ukuran statistik yang mengambarkan seberapa besar variasi dalam suatu kumpulan data, jika varians kecil artinya kumpulan data cenderung homogen dan data yang terlalu homogen tidak terlalu mempengaruhi dalam memprediksi. Untuk itu fitur-fitur yang memiliki data-data homogen bisa di drop.

Selain dari kolom-kolom yang memiliki variansi yang kecil, selanjutnya untuk kolom-kolom boolean seperti sex, 'EOR_Rwave', 'EODD_Rwave', 'EOR_Pwave', 'EODD_Pwave', 'EOR_Twave', 'EODD_Twave' akan di drop juga. Hal ini disebakan karena kolom-kolom ini tidak terlalu relevan dalam hal mendiagnosis seseorang memliki *arrythymia* atau tidak.

Setelah menghapus kolom-kolom dengan variansi yang kecil dan yang tidak relevan, data set hanya menyisakan 180 fitur.

Tahap terakhir dari eda dan *pre-processing data* melakukan scalling pada dataset supaya memliki range yang sama. Pada penelitian kali ini metode scalling yang digunakan adalah standarisasi, karena kebanyakan distribusi data mendekati distribusi normal. Hasil standarisasi di eksport menjadi file csv yang baru untuk dilakukan pemrosesan data/ pembentukan model.

Bab 2

Dasar Teori

Pada bab ini akan dijelaskan secara lebih mendetail tentang metode algoritma yang dipilih yaitu KNN dan naive bayes.

2.1. K- Nearest Neighboor (KNN)

Seperti namanya K tetangga terdekat, algoritma ini akan menganggap radius K dari dirinya mempunyai karakteristik atau klasifikasii yang sama. Ini bisa diandaikan pada suatu daerah memiliki 10 RT, dimana dibagi menjadi K km , dimana setiap K km pertama adalah satu RT dan seterusnya. Ide dasar dari KNN adalah secara intuitif jika setiap karakteristik dari data memiliki kedekatan yang mendekati nol seharusnya memiliki klasifikasi yang sama, kedekatan ini yang dismbolkan dengan K.

2.1.1. Jarak

Algoritma KNN sangat bergantung dengan kedekatan artinya sangat bergantung dengan jarak. Umumnya untuk menghitung jarak menggunakan rumus jarak dari euclidean yaitu:

$$\mathbf{r} = \sqrt{(\mathbf{x_2^2} - \mathbf{x_1^2}) + (\mathbf{y_2^2} - \mathbf{y_1^2})}$$

dimana :

r : radius /jarak euclidean

x : fitur 1

y: fitur 2

Selain pengunaan jarak euclidean untuk fitur-fitur yang diskrit biasa digunakan jarak manhattan. Manhattan distance menghitung jarak dengan menjumlahkan semua selisih jarak antar fitur lalu menjunlahkannya, rumus umumnya seperti ini:

$$\mathrm{d} = \sum_{\mathrm{i}=1}^{\mathrm{m}} |\mathrm{x_i} - \mathrm{y_i}|$$

dimana :

r : radius /jarak manhattan

x : fitur 1

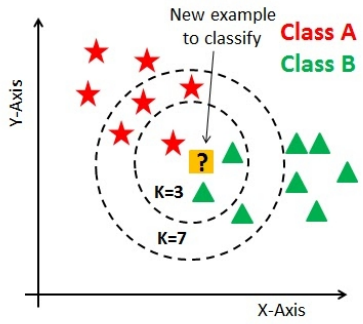
y: fitur 2

Selain dari kedua pengukuran jarak diatas ada beberapa lagi metode pengukuran seperti Minkowski, Hamming dsb. Namun hal itu tidak akan dijelaskan pada penelitian kali ini.

2.1.2. Nillai K

Salah satu problem yang krusial di KNN adalah penentuan nilai K, pemilihan nilai K yang tidak baik akan menyebabkan model yang dibuat berbasis KNN menjadi tidak optimal. Untuk itu diperlukan evaluasi nilai K yang baik supaya penentuan nilai K menjadi optimal.

Nilai K adalah banyaknya tetangga yang diambil berdasarkan jaraknya. Contoh jika K = 1 maka akan diambil 1 tetangga dari tetangga terdekat, begitu juga jika K =2, K=3 sampai K = n. Untuk lebih memudahkan lihat gambar 2.1



Gambar 2 .1. Ilustrasi Knn (source: https://www.http://www.kitainformatika.com/2019/10/hitung-manual-algoritma-k-nearest.html)

Ilustrasi pada gambar 2.1. mengambarkan bagaimana sistem dari knn, kotak kuning adalah entitas yang ingin di prediksi. Jika koordinatnya x,y maka saat k = 3 , 3 tetangga yang diambil pada kasus ini 2 segitiga hijau dan 1 bintang merah, maka klasifikasi kota kuning termasuk kedalam kelas segitiga hijau atau kelas B, begitu seterusnya sampai k= N.

Umumnya nilai k bernilai ganjil untuk menghindari jumlah yang sama pada kelas, namun tidak menutup kemungkinan untuk beberapa kasus k bernilai genap.

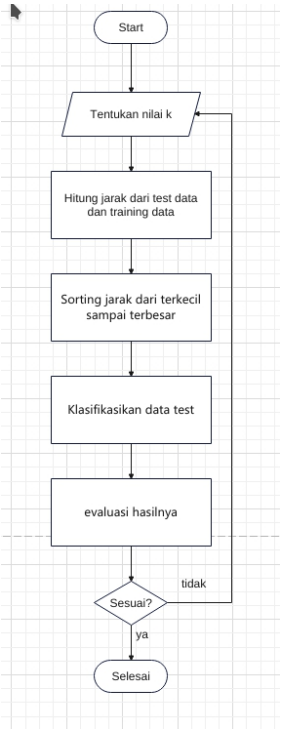
Optimalisasi nilai k dilakukan secara brute force artinya dilakukan dengan mengevaluasi dari k =1 sampai k = N lalu diplotting dan dilihat nilai k yang stabil dan memliki akurasi yang diinginkan.

2.1.3. Algoritma K-NN

Algoritma K-NN relatif lebih mudah untuk dipahami, adapun langkah-langkah nya sebagai berikut:

1. Tentukan Nilai K
2. Hitung Jarak d antara test data dan training data
3. sorting jarak d
4. pilih sebanyak K dari kumpulan jarak
5. klasifikasikan test data berdasarkan kelas terbanyak dari jarak yang sudah dipilih sebanyak K
6. Evaluasi hasilnya , jika sudah sesuai dengan yang diinginkan selesai, jika belum ulangi dari langkah pertama

Visualisasi nya dengan flowchart bisa dilihat pada gambar 2.2 dibawah ini.



Gambar 2.2. Flowchart algoritma KNN

2.1.4. Source Code

Untuk implementasi algoritma knn pada penelitian ini menggunakan python dengan dibantu beberapa library seperti matplotlib dan seaborn untuk ploting grafik, pandas untuk membantu proses pembacaan data dan juga numpy untuk kalkulasi perhitungan .

Sedangkan source code untuk algoritmanya sendiri dibuat pada folder tersendiri dan file tersendiri lalu dibuatkan class, hal ini untuk memudahkan jika dilain hari akan digunakan kembali atau jika ingin diperbaiki. Folder nya bernama **library_project** sedangkan filenya bernama **metode.py.**

2.1.4.1 Source code class Knearest Neighbors

```
class KnearestNeighbors:
    def __init__(self, k=1):
        self.k = k

    def euclidean_distance(self, x1, x2):
        return np.sqrt(np.sum((x1-x2)**2))

    def manhattan_distance(self, x1, x2):
        return np.sum(np.absolute(x1-x2))

    def list_distance(self, tipe, xtrain, xtest, posisi):
        distance = []
        for i in range(len(xtrain)):
            if tipe == "euclidean":
                distance.append(self.euclidean_distance(
                    xtrain.loc[i], xtest.loc[posisi]))
            elif tipe == "manhattan":
                distance.append(self.manhattan_distance(
                    xtrain.loc[i], xtest.loc[posisi]))

        return distance

    def predict(self, tipe, xtrain, xtest, posisi, ytrain, k=None):

        if k is None:
            k = self.k
        distance = self.list_distance(
            tipe, xtrain, xtest, posisi)
        data_ = xtrain.copy()
        data_["distance"] = distance
        data_["label"] = ytrain

        data_ = data_.sort_values(by="distance")
        y_pred = data_[:k].label.mode()

        return y_pred[0]

    def accuracy(self, y_pred, y_test):
        benar = 0
        for i in range(len(y_pred)):
            if y_pred[i] == y_test[i]:
                benar += 1

        return benar/len(y_test)
```

```
    def evaluasi(self, tipe, train, test, label, k=None, ):
        if k is None:
            k = self.k

        xtrain, ytrain = train.drop(label, axis=1), train[label]
        xtest, ytest = test.drop(label, axis=1), test[label]

        ypred = []
        for i in range(len(xtest)):
            ypred.append(self.predict(tipe, xtrain, xtest, i, ytrain, k))

        return self.accuracy(ypred, ytest)

    def get_average_accuracy(self, tipe, list_data, label, k=None, cetak=False):
        if k is None:
            k = self.k

        akurasi = []

        for i in range(len(list_data)):
            train, test = list_data[i]
            akurasi.append(self.evaluasi(tipe, train, test, label, k))

        if cetak:
            print("Untuk k : {} , Rata-rata akurasi:{}".format(k,
                  sum(akurasi)/len(akurasi)))
        else:
            return sum(akurasi)/len(akurasi)

    def plot_evaluasi_k(self, tipe, list_data, label, start, end, step=1):

        average_accuracy = []
        k_list = []
        for i in range(start, end, step):
            k_list.append(i)
            average_accuracy.append(
                self.get_average_accuracy(tipe, list_data, label, i))

        plt.plot(k_list, average_accuracy)
        plt.title("evaluasi K {}-{}".format(start, end-1))
        plt.xlabel("k value")
        plt.ylabel("accuracy")
        plt.show()
```

Gambar 2 .3. Source code

Pada source code diatas adalah sebuah class **KnearestNeighbors** yang berisi method-method yang nantinya digunakan dalam membangun model KNN. Atribut yang dimiliki class ini adalah nilai k dengan jika saat pemanggilan kelas ini nilai k tidak dimasukan akan secara default menganggap k = 1.

Selanjutnya akan di terangkan isi dari method-method yang terdapat dalam kelas tersebut:

1. **def euclidan_distance** berfungsi untuk menghitung jarak dengan metode euclidian dan mengembalikann hasil perhitungan tersebut, inputnya adalah fitur-fitur pada data train dan fitur-fitur pada data test

2. **def manhatan_distance** return dan inputnya sama, hanya saja perhitungan jaraknya berdasarkan metode manhattan..

3. **def list_distance** berfungsi untuk menghitung dan menghipun keseluruhan jarak dari data test dan data train. Input dari fungsi ini adalah tipe untuk memilih metode apa yang ingin digunakan apakah manhattan atau euclidean, xtrain adalah fitur-fitur pada data train, xtest fitur-fitur pada data test dan posisi untuk menandai pada baris keberapa data yang sedang dihitung. method ini mengembalikan kumpulan jarak.

4. **def predict** method yang nantinya akan mengembalikan nilai prediksi dengan inputan tipe, xtrain, xtest, posisi, ytest dan k dengan default None.

5. **def accuracy** method yang mengembalikan nilai akurasi dengan inputan hasil prediksi dan aktual nya

6. **def evaluasi** method yang mengembalikan nilai akurasi sama seperti def **accuracy** hanya saja inputannya berup tipe untuk menentukan penentuan metode perhitungan jarak, data test, data train, string kolom dependen dan k dengan default None

7. **def get_average_accuracy** method yang bisa berfungsi sebagai fungsi atau prosedur, tapi sama sama mengembalikan atau mencetak rata-rata akurasi dengan inputan tipe, kumpulan data, label, k dan boolean berupa apa mau di cetak atau tidak

8. **def_plot_evaluasi_k** method yang berfungsi sebagai procedure untuk menampilkan plotingan akurasi berdasarkan k yang dipilih inputan dari method ini adalah tipe, kumpulan data, label, start untuk dimulai dari k berapa dan end berkahir di k berapa ,lalu step plottnya akan ada dinterval berapa

2.1.4.2. Source code knn

```python
import pandas as pd
import numpy as np
from library_project.dataproses import split_data
from library_project.metode import KnearestNeighbors
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
# %%
# import data
dataku = pd.read_csv("feature_data_arythimia_std.csv")
# %%
# bagi menjadi 4 data dan split menjadi train dan test data
data1 = split_data(dataku[:113], 70, "right", True)
data2 = split_data(dataku[113:226], 70, "left", True)
data3 = split_data(dataku[226:339], 70, "middle", True)
data4 = split_data(dataku[339:], 70, "right", True)

# %%
train, test = data2
# %%
# prediksi akurasi k =15
knn = KnearestNeighbors()
train1, test1 = data3
akurasi_euclid = knn.evaluasi("euclidean", train1, test1, "diagnosis", k=15)
akurasi_manha = knn.evaluasi("manhattan", train1, test1, "diagnosis", k=15)

# %%
# rata-rata akurasi
list_data = [data1, data2, data4]
knn.get_average_accuracy(
    "euclidean", list_data, "diagnosis", 15, True)


# %%
# plot evaluasi akurasi 1-30
knn.plot_evaluasi_k("euclidean", list_data, "diagnosis", 1, 31)

# %%
```

Gambar 2 .4. Source code Knn

Penjelasan pada source code diatas adalah sebagai berikut pada baris 1-8 adalah mengimport library yang diperlukan termasuk library yang dibangun sendiri dalam folder library_project. Setelah itu dengan menggunakan pandas membaca file yang sudah di standarisasi , hasilnya di simpan di dalam variabel dataku. Lalu pada baris ke 14-17 data yang sudah diimport akan dibagi menjadi 4 pasangan data train dan test lalu disimpan di variabel data1-data4. Selanjutnya pada baris ke-23 panggil **class KnearestNeighbors** lalu simpan ke dalam variabel knn. kemudian pada baris ke-25 dan ke-26 adalah code untuk memprediksi sekaligus mengevaluasi berapa akurasi pada k = 15 pada metode pengukuran jarak yang berbeda. Lalu pada baris ke 30-32 adalah code untuk mendapatkan akurasi rata-rata dari keempat pasangan data yang sudah dipisah tadi. Selanjutnya pada baris terakhir adalah code untuk memplotinng akurasi data untuk k = 1 sampai k = 30.

2.2. Naive bayes

Metode Algoritma Naive bayes bersandarkan pada teorema bayes dalam ilmu probabilitas dan statistik. Teorema bayes mengatakan bahwa ketika telah diketahui bahwa suatu kejadian A telah diketahui peluangnya P(A) dan peluang B ketika A

sudah terjadi P(B|A) maka akan bisa dihitung peluang terjadinya A ketika B terjadi P(A|B). Secara matematis dirumuskan :

$$\mathbf{P(A|B)} = \frac{\mathbf{P(B|A)\ P(A)}}{\mathbf{P(B)}}$$

Untuk lebih mudah nya diberikan suatu study case saat wabah covid-19. Peluang seseorang memiliki virus corona di dalam dirinya adalah 0.5, dan peluang seseorang ketika melakukan test dinyatakan positif adalah 0.4, lalu probabilitas seseorang sudah memiliki virus corona melakukan tes dan hasilnya positif adalah 0.8 maka peluang sesorang melakukan test dan hasilnya positif memiliki virus di dalamnya adalah :

$$\mathbf{P(A|B)} = \frac{\mathbf{0.8 * 0.5}}{\mathbf{0.4}} = \mathbf{1}$$

Berdasarkan teorema tersebut naive bayes di bangun. Sendainya kita memiliki suatu data yang memiliki beberapa fitur atau parameter ingin diprediksi berdasarkan fitur-fitur yang dimiliki termasuk pada kelas manakah data tersebut ?. Jika sebelumnya kita sudah memiliki sekumpulan data lengkap dengan fitur dan klasifikasinya maka untuk memprediksi suatu data yang belum di ketahui kelasnya kita bisa memakai rumus teorema bayesian dimana :

$$\mathbf{P(C = c_i|X)} = \frac{\mathbf{P(X|C = c_i)P(C = c_i)}}{\mathbf{P(X)}}$$

dimana:

$P(C = c_i|X) = probabilitas\ kelasnya\ terhadap\ fitur - fiturnya$

$P(X|C = c_i) = Probabilitas\ setiap\ fitur\ terhadap\ kelasnya$

$P(C = c_i) = Probabilitas\ independen\ kelasnya$

$P(X) = Probabilitas\ independen\ setiap\ fiturnya$

Namun karena sangat sulit untuk meghitung probabilitas fitur terhadap kelasnya maka setiap fiturnya dianggap independen satu sama lain sehingga perhitungan nya menjadi:

$$\mathbf{P(C = c_i|X) = P(x_1|C)P(x_2|C)P(x_3|C)\ldots P(X_n|C) * P(C)}$$

karena itulah disebut naive bayes karena dianggap semua fitur saling independen, padahal jika berbicara pada kenyataannya hampir tidak mungkin setiap fitur bersifat saling independen.

Untuk lebih memahami bagaimana metode naive bayes bekerja perhatikan contoh berikut ini. Terdapat sebuah data set berisi record data tentang laporan cuaca setiap harinya dimana klasifikasi kelasnya ada 3 cerah, berawan dan hujan. Terdapat pula 3 fitur yang membersamainya, kelembaban , temperatur dan kondisi angin. Secara lengkap bisa dilihat pada tabel 2.1.

Kelembaban	Temperatur	Kondisi Angin	Cuaca
Wet	hot	tidak berangin	berawan
Wet	hot	berangin	hujan
wet	cold	tidak berangin	berawan
dry	hot	tidak berangin	tidak hujan
dry	cold	berangin	hujan

Table 2.1. Tabel data cuaca

Lalu pertanyaannya adalah jika kita memiliki data kelembaban dry, temperatur hot dan kondisi angin berangin bagaimana prediksinya apakah hujan atau tidak ?.

Untuk menjaawb pertanyaan tersebut pertama-tama kita cari probabilitas dari setiap kelas dan fitur terhadap kelas terlebih dahulu, kita akan menggunakan tabel karena akan lebih mudah. Tabel dari data diatas bisa dilihat pada gambar 2.5.

Kelembaban	Class		
	Berawan	hujan	tidak hujan
wet	1	0.5	0
dry	0	0.5	1

temperatur	Class		
	Berawan	hujan	tidak hujan
hot	0.5	0.5	1
cold	0.5	0.5	0

Kondisi Angin	Class		
	Berawan	hujan	tidak hujan
Berangin	0	1	0
tidak berangin	1	0	1

Class		
Berawan	hujan	tidak hujan
0.4	0.4	0.2

Gambar 2.5. Tabel Probabilitas setiap fitur berdasarkan kelas dan probabilitas independen kelas

Karena terdapat beberapa kondisi dimana probabilitasnya nol, maka dilakukan smothting laplace dengan menambahkan satu pada setiap bagian sehingga probabilitas masing masing akan menjadi seperti pada gambar 2.6.

Kelembaban	Class		
	Berawan	hujan	tidak hujan
wet	0.75	0.5	0.33
dry	0.25	0.5	0.67

temperatur	Class		
	Berawan	hujan	tidak hujan
hot	0.5	0.5	0.67
cold	0.5	0.5	0.33

Kondisi Angin	Class		
	Berawan	hujan	tidak hujan
Berangin	0.25	0.75	0.25
tidak berangin	0.75	0.25	0.75

Class		
Berawan	hujan	tidak hujan
0.4	0.4	0.2

Gambar 2.6. Tabel probabilitas setelah ditambah dengan satu

Setelah adanya smothing laplace kemudian dihitung setiap probabilitas kemungkinan kelas dari fitur data test:

$$P(Hujan|dry, hot, berangin) = P(dry|hujan)P(hot|hujan)P(berangin|hujan) * P(hujan)$$

$$= 0.5 * 0.5 * 0.75 * 0.4 = 0.075 \text{ ..... } (1)$$

$$P(Berawan|dry, hot, berangin) = P(dry|berawan)P(hot|berawan)P(berangin|berawan) * P(berawan)$$

$$= 0.25 * 0.5 * 0.25 * 0.4 = 0.0125 \text{ ..... } (2)$$

$$P(tidak\ hujan|dry, hot, berangin) = P(dry|tidak\ hujan)P(hot|tidak\ hujan)P(berangin|tidak\ hujan) * P(tidak\ hujan)$$

$$= 0.67 * 0.67 * 0.25 * 0.2 = 0.022445 \text{ ..... } (3)$$

Jika dibandingkan hasil dari 1,2 dan 3 maka probabilitas yang tertinggi ada di satu. Artinya data dengan kelembaban dry, temperatur hot dan kondisi angin berangin diklasifikasikan ke kelas **Hujan.**

Selain dari data-data diskrit, naive bayes juga bisa memprediksi fitur-fitur yang memiliki data kontinyu pengerjaannya mirip dengan yang diskrit, yang berbeda hanyalah rumus untuk mencari probabilitasnya.

Untuk data kontinyu secara matematis probabilitasnya dirumuskan :

$$\mathbf{P(X_j|C = c_i) = \frac{e^{-(\frac{(X_j - \mu_{ij})^2}{2\mu_{ij}^2})}}{\mu_{ij}\sqrt{2\pi}}}$$

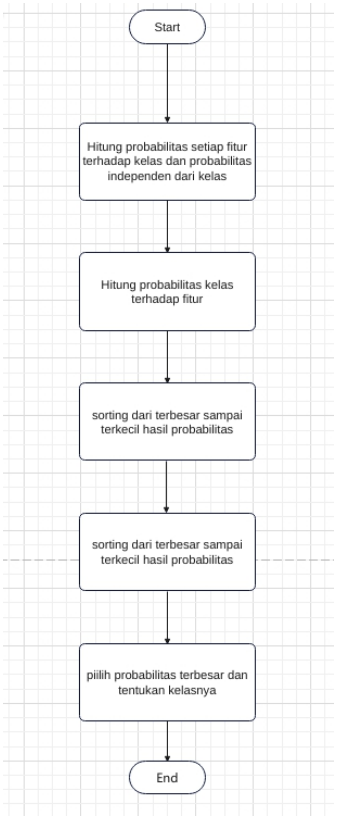
Selanjutnya untuk langkah-langkah setelah menghitung probabilitas, sama dengan proses pada data diskrit.

2.2.1. Algoritma Naive bayes

Algoritma dari naive bayes juga cukup sederhana dan mudah di mengerti, berikut ini adalah langkah-langkah dalam memprediksi suatu data test:

1. Hitung setiap probabilitas fitur terhadap kelas dan hitung probabilitas independen kelas.
2. Hitung probabilitas kelas terhadap fitur
3. sorting dari terbesar sampai terkecil hasil probabilitas
4. piilih probabilitas terbesar dan tentukan kelasnya
5. selesai

Jika di visualisasikan dengan flowchart maka akan seperti gambar 2.7



Gambar 2.7. Flowchart Algoritma Naive Bayes

2.2.2. Source Code

Seperti halnya dengan KNN pada naive bayes juga dibuatkan library tersendiri pada folder library_project dengan membuat sebuah class bernama **NaiveBayes ,** source code seperti yang terlihat pada gambar 2.8., 2.9 dan 2.10.

2.2.2.1 Source Code Class naive bayes

```
class NaiveBayes:
    def __init__(self, data_train, kolom_y, kelas):
        self.data_train = data_train
        self.kolom_y = kolom_y
        self.kelas = kelas

    def collect_mean(self, kolom_target):
        record = dict(self.data_train.groupby(
            self.kolom_y)[kolom_target].count())
        count = {key: record[key]+1 for key in record.keys()}
        total = dict(self.data_train.groupby(self.kolom_y)[kolom_target].sum())

        mean = {key: total[key]/count[key] for key in count.keys()}

        return mean

    def collect_std(self, kolom_target, mean):
        pembilang = 0
        std = {}
        for key in mean.keys():

            miu = mean[key]
            rec = self.data_train[kolom_target].loc[self.data_train[self.kolom_y] == key]
            for i in range(len(rec)):
                value = (rec.iloc[i]-miu)**2
                pembilang += value
            std[key] = math.sqrt(pembilang/(len(rec)+len(rec)))

        return std

    def collect_probablity_y(self):
        record = dict(self.data_train[self.kolom_y].value_counts())
        dictio = {key: record[key]+1 for key in record.keys()}
        for i in dictio.keys():
            dictio[i] = dictio[i]/(len(self.data_train)+len(self.kolom_y))

        return dictio
```

Gambar 2.8. Source code class part 1

Gambar 2.8. Menunjukan sebuah class bernama Naive Bayes dengan parameter input, data_train, kolom_y dan kelas/kategorisasi. Untuk bagian pertama ini terdapat 3 method yaitu collect_mean, collect_std dan collect_probability_y. Adapun ketiga metode tersebut memiliki fungsi:

1. **collect_mean** akan mengembalikan kumpulan mean dari setiap fitur berdasarkan kategori di dalam kolom target.
2. **collect_std** akan mengembalikan kumpulan standard deviasi dari setiap fitur berdasarkan kategori di dalam kolom target.

3. **collect_probability_y** akan mengembalikan probabilitas dari setiap kelas di dalam kolom target.

```
    def calculate_probability(self, mean, std, x):
        pembilang = math.exp(-((x-mean)**2/(2 * std**2)))
        penyebut = std*math.sqrt(2*math.pi)

        return 1/penyebut*pembilang

    def max_prob(self, dictio):
        key = [i for i in dictio.keys()]
        maxi = dictio[key[0]]
        maxi_id = key[0]

        for keys in key[1:]:
            if maxi < dictio[keys]:
                maxi_id = keys
                maxi = dictio[keys]

        return maxi_id

    def prediksi(self, data_test, truth_colom=None):
        result = []

        for i in range(len(data_test)):
            y_prob = self.collect_probablity_y()
            dict_prob = {}
            for kolom in data_test.columns:
                nilai_x = data_test[kolom].iloc[i]
                mean = self.collect_mean(kolom)
                std = self.collect_std(kolom, mean)

                for key in y_prob.keys():

                    y_prob[key] *= self.calculate_probability(
                        mean[key], std[key], nilai_x)
            dict_prob["id"] = i
            dict_prob["probabilitas"] = y_prob
            dict_prob["result"] = self.max_prob(y_prob)
            if type(truth_colom) != type(None):
                dict_prob["truth_data"] = truth_colom.iloc[i]
                print("record_id : {}, prediksi: {}, truth: {}".format(
                    dict_prob["id"], dict_prob["result"], dict_prob["truth_data"]))
            result.append(dict_prob)

        return result

    def akurasi(self, TP, TN, FP, FN):    Rename this parameter "FN" to match the regular expression ^[_a-z][a-z0-9_]*$.
        return (TP+TN)/(TP+TN+FP+FN)

    def presisi(self, TP, FP):    Rename this parameter "FP" to match the regular expression ^[_a-z][a-z0-9_]*$.
        try:
            return TP/(TP+FP)
        except:    Specify an exception class to catch or reraise the exception
            return 0

    def recall(self, TP, FN):    Rename this parameter "FN" to match the regular expression ^[_a-z][a-z0-9_]*$.
        try:
            return TP/(TP+FN)
        except:    Specify an exception class to catch or reraise the exception
            return 0
```

Gambar 2.9. Source Code Naive bayes part 2

Selanjutnya pada source code part2 di gambar 2.9 memiliki 6 method yaitu calculate_probability, max_prob, prediksi, akurasi, presisi, recall. Adapun fungsi masing-masing method adalah sebagai berikut :

1. **calculate_probability** akan mengembalikan nilai probabilitas fitur terhadap kelas, dengan inputan berupa kumpulan mean, kumpulan std dan data pada suatu fitur.

2. **max_prob** akan mengembalikan kategori dari kelas berdasarkan probabilitas tertinggi dari kumpulan probabilitas.

3. **prediksi** akan mengembalikan kumpulan hasil prediksi

4. **akurasi** akan mengembalikan nilai akurasi berdasarkan True positif(TP), True Negatif(TN), False Positif(FP), False Negatif(FN).

5. **presisi** akan mengembalikan nilai presisi berdasarkan True Positif dan False Positif.

6. **recall** akan mengembalikan nilai recall berdasarkan True Positif dan False negatif.

```
def confusionMatrix(self, result):    Rename method "confusionMatrix" to match the regular expression ^[a-z_][a-z0-9_]*$.

    result_matrix = []

    for kelas in self.kelas:

        TP = 0
        TN = 0
        FP = 0
        FN = 0

        dictio = {}
        print(kelas)

        for res in result:
            if res["truth_data"] == kelas and res["truth_data"] == res["result"]:

                TP += 1
            elif res["truth_data"] != kelas and res["truth_data"] == res["result"]:
                TN += 1
            elif res["truth_data"] == kelas and res["truth_data"] != res["result"]:
                FP += 1
            elif res["truth_data"] != kelas and res["truth_data"] != res["result"]:
                FN += 1

        dictio["kelas"] = kelas
        dictio["kumpulan"] = [TP, TN, FP, FN]
        print(dictio["kumpulan"])

        dictio["akurasi"] = self.akurasi(TP, TN, FP, FN)
        dictio["presisi"] = self.presisi(TP, FP)
        dictio["recall"] = self.recall(TP, FN)
        result_matrix.append(dictio)

    return result_matrix

def cetak_hasil(self, data_test, truth_colom=None):
    hasil = self.prediksi(data_test, truth_colom)

    matrix = self.confusionMatrix(hasil)
    print(matrix)

    akurasi = 0
    presisi = 0
    recall = 0

    for data in matrix:
        akurasi += data["akurasi"]

        presisi += data["presisi"]
        recall += data["recall"]

    akurasi = akurasi / len(self.kelas)
    presisi = presisi / len(self.kelas)
    recall = recall / len(self.kelas)

    print("akurasi: {}, presisi: {}, recall: {}".format(
        akurasi, presisi, recall))
```

Gambar 2.10. Source code part 3

Pada source part 3 terdapaat 2 method yaitu confusion_matrix dan cetak_hasil. adapun fungsi dari kedua method tersebut adalah :

1. **confussion_matrix** akan mengembalikan confusiion matrix.

2. **cetak_hasil** sebuah prosedur yang akan mencetak hasil prediksi, akurasi, presisi dan recall.

2.2.2.2.Source Code naive_bayes

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from library_project.metode import NaiveBayes
from library_project.dataproses import split_data
sns.set()

Run Cell | Run Above | Debug Cell
# %%
dataku = pd.read_csv("feature_data_arythimia_std.csv")
Run Cell | Run Above | Debug Cell
# %%
train, validation = split_data(dataku, 70, "right", True)
Run Cell | Run Above | Debug Cell
# %%
kelas = sorted([i for i in train["diagnosis"].unique()])
nb = NaiveBayes(train, "diagnosis", kelas)
Run Cell | Run Above | Debug Cell
# %%
valid, truth_data = validation.drop(
    columns=["diagnosis"]), validation["diagnosis"]
Run Cell | Run Above | Debug Cell
# %%
nb.cetak_hasil(valid, truth_data)
Run Cell | Run Above | Debug Cell
# %%
```

Gambar 2.11. Source code impementasi naive bayes

Pada implementasi naive bayes kode baris 1 - 5 adalah library yang digunakan termasuk library yang dibuat sendiri pada folder library_project. baris ke-7 membaca data yang sudah melewati feature engineering dan disimpan di variable dataku. Baris ke-8 data di split menjadi data train dan data validation/test. baris ke-9 mendapatkan kategori yang ada di kolom diagnosis, baris ke-10 memanggil kelas NaiveBayes dengan input data train, kolom diagnosis dan list kelas. Baris ke-11 mendapatkan kolom truth dan data test yang sudah di drop kolom targetnya. Baris ke-12 akan mencetak hasil prediksi berserta akurasinya.

2.3. Link Github

Link github untuk semua source code :

https://github.com/khalifardy/artihimia_diagnosis

Bab 3

Evaluasi Hasil Dan Diskusi

Pengukuran performansi pada penelitian ini menggunakan 2 metode yaitu :

1. Metode akurasi biasa
2. Metode conffusion matrix

Metode akurasi biasa digunakan pada algoritma KNN dimana rumus umumnya adalah :

$$\textbf{akurasi} = \frac{\textbf{jumlah data yang benar di prediksi}}{\textbf{Jumlah keseluruhan data}}$$

Sedangkan metode conffusion matrix digunakan untuk mengukur performansi algoritma naive bayes. Pada confussion matrix terdapat 4 parameter untuk pengukuran performansi yaitu True Positif (TP) artinya data yang benar diprediksi, True Negatif artinya data yang benar diprediksi namun bukan bagian dari TP, False Positif (FP) artinya data yang prediksinya tidak sesuai dengan aktual yang seharusnya aktualnya ada pada TP dan terakhir False negatif(FN) data yang prediksinya tidak sesuai dari aktual yang seharusnya aktualnya ada pada TN.

Terdapat 3 pengukuran performansi pada conffusion matrix:

1. **Akurasi** , suatu ukuran keseluruhan data yang benar diprediksi dibandingkan dengan total data. Rumusnya :

$$\textbf{akurasi} = \frac{\textbf{TP} + \textbf{TN}}{\textbf{TP} + \textbf{TN} + \textbf{FP} + \textbf{FN}}$$

2. Presisi , suatu ukuran seberapa besar data positif yang benar dibandingkan keseluruhan data positif. Rumusnya:

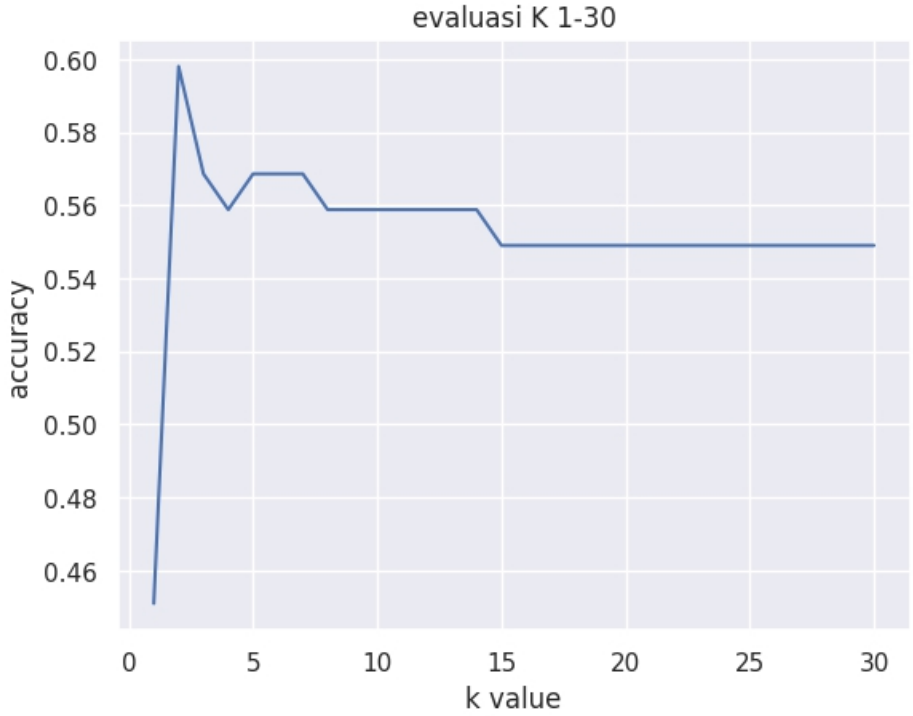
$$\textbf{Presisi} = \frac{\textbf{TP}}{\textbf{TP} + \textbf{FP}}$$

3. **Recall,** suatu ukuran seberapa besar suatu data positif yang benar dibandingkan dengan data positif yang benar di bandingkan data postif yang benar ditambah data negatif yang salah. Rumusnya:

$$\textbf{Recall} = \frac{\textbf{TP}}{\textbf{TP} + \textbf{FN}}$$

3.1. Hasil KNN

Setelah implementasi algoritma knn pada 4 data test dengan k = 15 di dapatkan rata-rata akurasi **0.5490196078431373.** Selanjutnya di lakukan evaluasi kembali dengan memploting rata-rata akurasi dengan k =1 s/d k = 30. Hasilnya bisa dilihat pada grafik di gambar 3.1.



Gambar. 3.2. Grafik evaluasi k

Dari grafik diatas bisa disimpulkan bahwa k mendapatkan nilai akurasi tertingi pada k = 3 yaitu **0.592342342.** Sedangkan kestabilan akurasi tertinggi ada pada k >= 15.

3.2. Hasil Naive Bayes

Hasil performansi dengan menggunakan naive bayes didapatkan akurasi sebesar **0.6176470588235292**, presisi sebesar **0.15783302063789867,** recal sebesar **0.06050998072600134.** Hasil ini di dapatkan dari rata-rata akurasi, rata-rata presisi, rata-rata recall dari 13 kategori di kolom diagnosis.

Dibawah ini diperlihatkan semua confusion matrix dari setiap kategorisasi di kolom diagnosis.

Kelas 1 (normal)			
		Aktual	
Prediksi		+	-
	+	76	46
	-	6	8

Table 3.1. Confusion matrix kelas 1

Dari confusion matrix diatas didapatkan akurasi kelas 1 sebesar **0.617647059**, presisi kelas 1 sebesar **0.904761905**, dan recall kelas 1 sebesar **0.926829268.**

Kelas 2 (Ischemic changes)			
		Aktual	
Prediksi		+	-
	+	3	43
	-	9	81

Table 3.2. Confusion matrix kelas 2

Dari confusion matrix diatas didapatkan akurasi kelas 2 sebesar **0.617647059**, presisi kelas 2 sebesar **0.0357142857**, dan recall kelas 2 sebesar **0.25.**

Kelas 3 (Old Anterior Myocardial Infarction)			
		Aktual	
Prediksi		+	-
	+	2	50
	-	2	82

Table 3.3. Confusion matrix kelas 3

Dari confusion matrix diatas didapatkan akurasi kelas 3 sebesar **0.617647059**, presisi kelas 3 sebesar **0.0238095238**, dan recall kelas 3 sebesar **0.5.**

Kelas 4 (Old Inferior Myocardial Infarction)			
		Aktual	
Prediksi		+	-
	+	0	47
	-	5	84

Table 3.4. Confussion Matrix kelas 4

Dari confusion matrix diatas didapatkan akurasi kelas 4 sebesar **0.617647059**, presisi kelas 4 sebesar **0**, dan recall kelas 4 sebesar **0.**

Kelas 5 (Sinus tachycardy)			
		Aktual	
Prediksi		+	-
	+	0	49
	-	3	84

Table 3.5. Confussion Matrix kelas 5

Dari confusion matrix diatas didapatkan akurasi kelas 5 sebesar **0.617647059**, presisi kelas 5 sebesar **0**, dan recall kelas 5 sebesar **0.**

Kelas 6 (Sinus bradycardy)			
		Aktual	
Prediksi		+	-
	+	0	45
	-	7	84

Table 3.6. Confussion Matrix kelas 6

Dari confusion matrix diatas didapatkan akurasi kelas 6 sebesar **0.617647059**, presisi kelas 6 sebesar **0**, dan recall kelas 6 sebesar **0.**

Kelas 7 (PVC)			
		Aktual	
Prediksi		+	-
	+	0	52
	-	0	84

Table 3.7. Cofussion matrix kelas 7

Dari confusion matrix diatas didapatkan akurasi kelas 7 sebesar **0.617647059**, presisi kelas 7 sebesar **0**, dan recall kelas 7 sebesar **0.**

Kelas 8 (PSC)			
		Aktual	
Prediksi		+	-
	+	0	52
	-	0	84

Table 3.8. Confussion matrix kelas 8

Dari confusion matrix diatas didapatkan akurasi kelas 8 sebesar **0.617647059**, presisi kelas 8 sebesar **0**, dan recall kelas 8 sebesar **0.**

Kelas 9 (Left bundle branch block)			
		Aktual	
Prediksi		+	-
	+	0	48
	-	4	84

Table 3.9. Confussion matrix kelas 9

Dari confusion matrix diatas didapatkan akurasi kelas 9 sebesar **0.617647059**, presisi kelas 9 sebesar **0**, dan recall kelas 9 sebesar **0.**

Kelas 10 (Right bundle branch block)			
		Aktual	
Prediksi		+	-
	+	3	47
	-	5	81

Table 3.10. Confussion matrix kelas 10

Dari confusion matrix diatas didapatkan akurasi kelas 10 sebesar **0.617647059**, presisi kelas 10 sebesar **0.0357142857**, dan recall kelas 10 sebesar **0.375.**

Kelas 14 (Left ventricule hypertrophy)			
		Aktual	
Prediksi		+	-
	+	0	51
	-	1	84

Table 3.11. Confussion matrix kelas 14

Dari confusion matrix diatas didapatkan akurasi kelas 14 sebesar **0.617647059**, presisi kelas 14 sebesar **0**, dan recall kelas 14 sebesar **0.**

Kelas 15 (Left ventricule hypertrophy)			
		Aktual	
Prediksi		+	-
	+	0	49
	-	3	84

Table 3.12. Confussion matrix kelas 15

Dari confusion matrix diatas didapatkan akurasi kelas 15 sebesar **0.617647059**, presisi kelas 15 sebesar **0**, dan recall kelas 15 sebesar **0.**

Kelas 16 (others)			
		Aktual	
Prediksi		+	-
	+	0	45
	-	7	84

Table 3.13. Confussion matrix kelas 15

Dari confusion matrix diatas didapatkan akurasi kelas 15 sebesar **0.617647059**, presisi kelas 15 sebesar **0**, dan recall kelas 15 sebesar **0.**

3.3. Pembahasan

Jika diperbandingkan hasil performansi dari kedua algoritma KNN dan naive bayes dari sisi akurasi naive bayes memilki akurasi sedikit lebih baik dari KNN. Namun jika ditinjau dari sisi kedokteran kedua performansi dari algoritma ini memiliki akurasi yang jauh dari harapan. Karena kedokteran mengharuskan zero toleransi atas kesalahan maka paling tidak akurasi yang dicapai harus diatas **90%.**

Akurasi yang didapat tidak sesuai harapan bisa disebabkan oleh beberapa hal termasuk EDA/preprocessing yang kurang maksimal. Namun jika diasumsikan EDA sudah cukup baik dan dilihat dari segi data train, pada kolom diagnosis variansi data train yang ada kurang baik. Jika di detailkan akan seperti tabel 3.14.

Kelas	jumlah data
1	245
2	44
3	15
4	15
5	13
6	25
7	3
8	2
9	9

10	50
11	0
12	0
13	0
14	4
15	5
16	22

Table 3.14. Tabel Jumlah data

Jika dilihat confusion matrix per kelas naive bayes di jumlah data yang sedikit pada data train tidak ada yang bisa diprediksi secara tepat. Karena memang untuk perhitungan naive bayes akan di kalikan probabilitas independen kelas tersebut, disebabkan oleh probabilitas kategori kelas 1 lebih tinggi dibandingkan dengan yang lainnya maka kebanyakan prediksi yang dihasilkan oleh model akan memprediksi kelas 1.

Selain itu baik KNN maupun Naive bayes tidak bisa memprediksi kategorisasi yang tidak ada pada data train seperti kategori 11,12 dan 13 . Karena basis dari naive bayes adalah probabilitas dan KNN adalah kedekatan, jika tidak ada data pada kejadian sebelumnya maka KNN dan naive bayes tidak bisa memprediksinya.

Bab 4

Kesimpulan dan Saran

4.1. Kesimpulan

Kesimpulan dari penelitian ini adalah Naive Bayes memiliki akurasi yang sedikit lebih baik dari KNN. Namun secara performansi keduanya masih jauh dari akurasi yang ditetapkan di bidang kedokteran yaitu diatas **90 %.** Performansi yang masih jauh dari harapan karena data set yang tersedia kurang bervariasi pada kolom diagnosis, terlalu banyak ketimpangan data antar kelas sehingga menyebabkan model baik model KNN atau naive bayes ambigu dalam menyimpulkan atau memprediksi.

4.2. Saran

Untuk penelitian kedepannya bisa dilakukan preprocessing dengan menggunakan **principal component analysys(PCA)** untuk pemilihan fitur dan menyederhanakan dimensi. Selain itu disarankan juga untuk mencoba metode algoritma lainnya seperti deeplearning untuk memodelkan sistem prediksi yang lebih baik.

Link:

Makalah,Presentasi dan video : tugas_besar

link source code : github

Bab 5

Daftar Pustaka

[1] *Dr. Suyanto, S.T,M.Sc. , Artificial Intelligence searching,reasoning,planning dan learning, 3nd ed., Bandung, Indonesia: Penerbit Informatika, 2021.*

[2] *Primartha Rifkie. , Belajar Machine Learning Teori dan Praktek, 1st ed., Bandung, Indonesia: Penerbit Informatika, 2018.*

[3] *Jubilee Enterprise. , python untuk analisis dan visualisasi data, 1st ed., Jakarta, Indonesia: Elex Media Komputindo, 2023.*

[4] *Farrell Peter, Fuentes Alvaro, Sudhir Kolhhe Ajinkya, Nguyen Quan, Sarver Joseph Alexander, Tsatsos Marios. , The statistics and Calculus With Python, 1st ed., Birmingham, UK: Pact Publishing Ltd, 2020.*

[5] *Walpole, Ronald E; Myers, Raymond H, Myers, Sharon L; Ye,Keying. Probability & statistics for engineer & scientist :MyStatLabUpdate. 9th ed.,Ediburgh,England:Pearson Education Limited, 2016.*