PROJEK UJIAN AKHIR SEMESTER

PENGANTAR DATA SAINS

PEMODELAN DAN KLASIFIKASI READMISI PASIEN DIABETES DENGAN ARTIFICIAL NEURAL NETWORKS

Muhammad Rainul Hakim

22/493467/PA/21193

Erlin Shofiana

22/493520/PA/21196

Elgita Kisti Cahyani

22/499206/PA/21536

DOSEN PENGAMPU

Drs.Danardono, MPH., Ph.D

Mohammad Fahruli Wahyujati, S.Si., M.Si.

PROGRAM STUDI STATISTIKA

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2024

ABSTRAK

Diabetes Melitus (DM) adalah salah satu penyakit kronis dengan tingkat readmisi pasien ke rumah sakit yang cukup tinggi. Tingginya tingkat readmisi dapat menjadi indikator kualitas perawatan yang tidak optimal. Penelitian ini bertujuan untuk mengidentifikasi faktor-faktor yang berkontribusi terhadap tingginya tingkat readmisi. Artificial Neural Network (ANN) dengan Multilayer Perceptron (MLP) digunakan untuk mengklasifikasikan readmisi pasien diabetes. Hasil menunjukkan akurasi 80% dengan presisi, recall, dan f1-score yang seimbang. Analisis ini menemukan bahwa faktor penting seperti perubahan pengobatan, dosis obat generik, jenis kelamin, dan umur sangat berpengaruh dalam menentukan readmisi pasien. Temuan ini dapat meningkatkan keselamatan dan kualitas hidup pasien diabetes dengan menyediakan panduan perawatan hiperglikemia yang lebih terstruktur dan konsisten.

Key words: *Diabetes Melitus (DM), Readmisi, Artificial Neural Network (ANN), Multilayer Perceptron (MLP)*

1. LATAR BELAKANG

Diabetes melitus adalah salah satu penyakit kronis yang paling umum di seluruh dunia dan memerlukan pengelolaan ketat untuk mencegah komplikasi serius. Tingginya tingkat readmisi pasien diabetes ke rumah sakit menjadi tantangan besar, tidak hanya membebani sistem kesehatan dengan biaya tinggi tetapi juga menunjukkan kualitas perawatan yang mungkin tidak optimal. Menurut International Diabetes Federation (IDF), sekitar 537 juta orang dewasa hidup dengan diabetes pada tahun 2021, dan angka ini diperkirakan akan meningkat. Di Amerika Serikat, biaya tahunan untuk perawatan diabetes mencapai USD 327 miliar. Pasien diabetes memiliki tingkat readmisi rumah sakit yang lebih tinggi, mencapai sekitar 20% dalam 30 hari setelah keluar. Faktor-faktor seperti pengelolaan glukosa yang buruk, kondisi komorbiditas, dan kurangnya edukasi pasien berkontribusi pada readmisi ini. Intervensi yang dapat membantu mengurangi readmisi termasuk program manajemen penyakit yang komprehensif, koordinasi perawatan yang lebih baik, dan penggunaan teknologi untuk pemantauan dan komunikasi pasien. Upaya berkelanjutan sangat penting untuk meningkatkan manajemen diabetes dan mengurangi readmisi, yang pada akhirnya meningkatkan kualitas hidup pasien dan mengurangi beban biaya sistem kesehatan

Penelitian ini berfokus pada analisis klasifikasi pasien diabetes yang melakukan readmisi dan yang tidak, dengan kategori readmisi ditentukan jika pasien kembali berkunjung ke rumah sakit dalam waktu kurang dari 30 hari setelah keluar. Dalam penelitian ini data bersumber dari *Health Facts database (Cerner Corporation, Kansas City, MO)*, yang merupakan *data warehouse* nasional yang mengumpulkan catatan klinis di seluruh rumah sakit di Amerika Serikat.

Tujuan utama dari penelitian ini adalah untuk mengidentifikasi faktor-faktor yang berkontribusi terhadap tingginya tingkat readmisi pada pasien diabetes. Dengan memahami faktor-faktor ini, diharapkan dapat dikembangkan strategi intervensi yang lebih efektif untuk mengurangi readmisi dan meningkatkan kualitas perawatan pasien diabetes.

2. METODE PENELITIAN

2.1 Data

Data dalam penelitian ini bersumber dari *Health Facts database* (*Cerner Corporation, Kansas City, MO*), yang merupakan *data warehouse* nasional yang mengumpulkan catatan klinis di seluruh rumah sakit di Amerika Serikat. Basis data ini mencakup data pertemuan (darurat, rawat jalan, dan rawat inap), spesialisasi penyedia layanan, demografi, diagnosis, dan prosedur rumah sakit yang didokumentasikan oleh kode ICD-9-CM, data laboratorium, data farmasi, angka kematian di rumah sakit, dan karakteristik rumah sakit. Semua data diidentifikasi sesuai dengan Undang-Undang Portabilitas dan Akuntabilitas Asuransi Kesehatan tahun 1996 sebelum diberikan kepada penyelidik.

Data yang digunakan dalam penelitian ini merupakan ekstrak yang mencakup 10 tahun (1999-2008) perawatan klinis di 130 rumah sakit dan jaringan layanan terintegrasi di seluruh Amerika Serikat: data ini mencakup berbagai wilayah geografis, yaitu Midwest (18 rumah sakit), Northeast (58 rumah sakit), South (28 rumah sakit), dan West (16 rumah sakit). Sebagian besar rumah sakit memiliki jumlah tempat tidur antara 100 dan 499, dengan beberapa rumah sakit memiliki jumlah tempat tidur kurang dari 100 dan beberapa lainnya memiliki lebih dari 500 tempat tidur.

Data yang digunakan terdiri dari 101.766 pertemuan unik (kunjungan) dan 50 fitur untuk setiap pertemuannya. Data ini mencakup rawat inap dan rawat jalan, termasuk unit gawat darurat, untuk kelompok pasien yang sama.

2.2 Pre-Processing Data

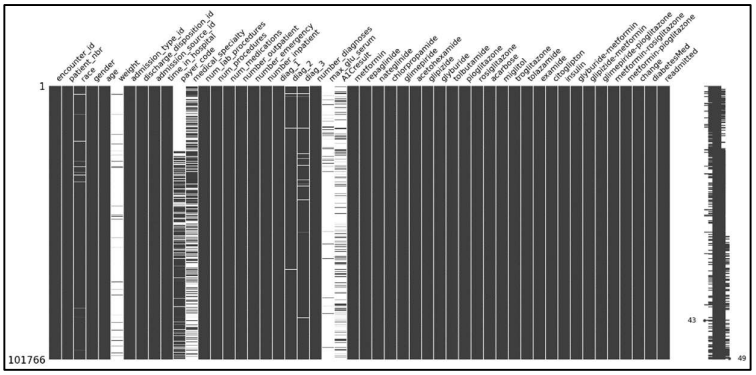
Berikut adalah langkah-langkah preprocessing yang dilakukan pada dataset diabetes:

1. Mengganti nilai kategori kolom ‘readmitted’

   Kolom ‘readmitted’ memiliki tiga kategori: ‘>30’,’<30’ dan ‘NO’. Untuk menyederhanakan analisis, nilai ‘>30’ diubah menjadi ‘NO’ dan ‘<30’ diubah menjadi ‘YES’.
2. Mengganti tanda ‘?’ dengan nilai ‘NA’

   Beberapa kolom dalam dataset memiliki nilai ‘?’ untuk menandakan data yang hilang. Nilai ‘?’ ini diganti dengan NA (Not Available) untuk memudahkan identifikasi dan penanganan missing values.
3. Menghitung dan memvisualisasikan Missing Value

Menghitung persentase missing values untuk setiap kolom dan memvisualisasikannya menggunakan library missingno. Kolom yang memiliki missing value yaitu race, weight, payer code, medical specialty, diagnosis (1, 2 dan 3), max_glu_serum, dan A1Cresult. Berikut visualisasi missing value tersebut:



Gambar 2.1 Visualisasi *Missing Value*

4. Menghapus kolom dengan Missing Value yang tinggi

   Kolom yang memiliki lebih dari 35% missing values dihapus dari dataset.

5. Imputasi Missing Value pada kolom kategorikal

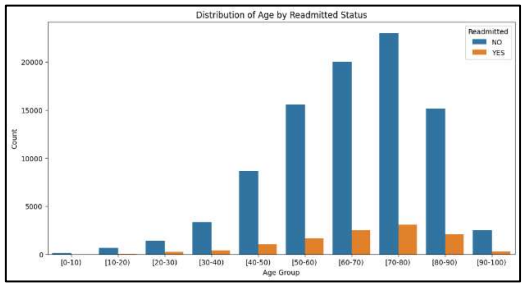
   Missing values pada kolom kategorikal diimputasi menggunakan modus (nilai yang paling sering muncul) dari setiap kategori pada kolom referensi tertentu. Fungsi fill_na_by_mode digunakan untuk mengisi nilai yang hilang berdasarkan kategori referensi.

6. Menyederhanakan kategori diagnosis dan usia

   Kategori diagnosis pada kolom diag_1, diag_2, dan diag_3 disederhanakan menjadi kategori yang lebih umum menggunakan fungsi categorize_diagnosis. Kolom asli kemudian dihapus dan digantikan oleh kolom baru simplified_diag_1, simplified_diag_2, dan simplified_diag_3.

   Kolom age disederhanakan menjadi dua kategori: 'Dibawah 50' dan 'Diatas 50'. Kolom asli age dihapus dan digantikan oleh kolom baru age_group.

   Adapun hasil visualisasi age group sebagai berikut:



Gambar 2.2 Visualisasi Persebaran Umur Pasien Diabetes

Data menunjukkan bahwa kelompok usia yang lebih tua (di atas 50 tahun) memiliki jumlah pasien yang lebih tinggi dan lebih cenderung kembali dirawat inap setelah perawatan awal. Ini mungkin menunjukkan bahwa pasien yang lebih tua memerlukan perhatian lebih dalam perawatan diabetes dan manajemen kesehatan untuk mengurangi kemungkinan rawat inap kembali.

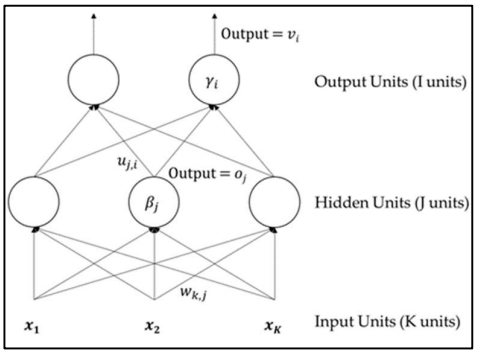
7. Menghapus kolom yang tidak relevan
   Kolom encounter_id dan patient_nbr dihapus karena tidak relevan untuk analisis.
8. Label Encoding pada kolom kategorikal dan melakukan pemisahan fitur dan target
   Kolom-kolom kategorikal diencode menjadi nilai numerik menggunakan LabelEncoder dari sklearn. Selanjutnya Dataset dipisahkan menjadi fitur (X) dan target (y).
9. Melakukan Balancing data dengan SMOTE
   Oversampling dilakukan menggunakan SMOTE (Synthetic Minority Over-sampling Technique) untuk menangani ketidakseimbangan kelas.
10. Memisahkan data train dan test
    Data dibagi menjadi data pelatihan (train) dan data pengujian (test).

2.3 Metode *Artificial Neural Network*

Neural Network adalah salah satu teknik *machine learning* yang sering dikenal sebagai *deep learning*. Metode ini melatih komputer untuk memproses data menggunakan algoritma yang terinspirasi dari cara kerja jaringan saraf di otak manusia. Algoritma *Neural Network* bekerja dengan menerima, memproses, dan mengirimkan informasi melalui jaringan neuron yang saling terhubung. Jaringan ini bersifat adaptif sehingga komputer akan belajar untuk memperbaiki kesalahan yang terjadi. Dalam *Neural Network*, data diproses melalui berbagai lapisan neuron buatan. Setiap lapisan bertugas mengenali pola dan hubungan dalam data tersebut. Proses ini memungkinkan *Neural Network* untuk menyelesaikan berbagai tugas kompleks dan melibatkan data dalam jumlah besar.

Artificial Neural Network (ANN) adalah salah satu cabang dari *Neural Network* yang khusus digunakan untuk data terstruktur, seperti data numerik dan kategori. Salah satu arsitektur ANN yang sering digunakan adalah *Multilayer Perceptron* (MLP) atau *Feedforward Neural Network*. MLP biasanya terdiri dari tiga lapisan utama, yaitu lapisan *input*, lapisan *hidden*, dan lapisan *output*. Prosesnya dimulai dengan lapisan *input* yang menerima data tanpa transformasi, kemudian data ini langsung diteruskan ke lapisan *hidden*. Di lapisan *hidden*, data

diproses dan fungsi aktivasi diterapkan pada setiap neuron sebelum hasilnya diteruskan ke lapisan berikutnya. Lapisan *output* adalah lapisan terakhir yang bertanggung jawab untuk menghasilkan prediksi akhir. Selain itu, ada juga lapisan tambahan yang disebut *dropout layer*, yang berfungsi mematikan neuron secara acak untuk menghindari *overfitting*. Jaringan ini biasanya bersifat *fully connected*, artinya setiap neuron di satu lapisan terhubung dengan semua neuron di lapisan berikutnya.

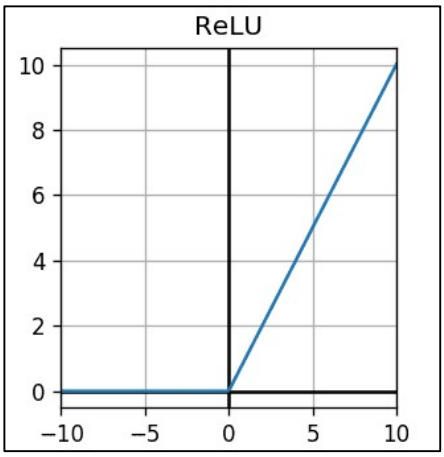


Gambar 2.3 Algoritma *Artificial Neural Network*

Backpropagation adalah algoritma yang biasanya digunakan untuk melatih MLP. Algoritma ini bekerja dengan mengubah bobot sinapsis (*synapse weight*) secara bertahap dari lapisan *output* ke lapisan *input* berdasarkan perbedaan antara *output* yang dihasilkan dengan *output* yang diharapkan (*desired output*). Proses ini memungkinkan koreksi bobot sinapsis dari lapisan *output* ke lapisan *hidden*. Kemudian *error* yang terjadi diteruskan kembali ke lapisan sebelumnya. *Backpropagation* menggunakan prinsip *gradient descent* untuk meminimalkan *loss*, dengan cara menghitung *gradien loss function* terhadap bobot sinapsis dan kemudian memperbarui bobot-bobot tersebut secara iteratif.

Fungsi aktivasi dalam ANN memiliki fungsi untuk mengaktifkan atau tidak mengaktifkan neuron. Salah satu fungsi yang populer digunakan adalah Fungsi Aktivasi *Rectified Linear Unit* (ReLU). Fungsi ini memiliki bentuk matematis yang dapat mengubah input negatif menjadi nol dan mempertahankan input positif seperti aslinya. Berikut merupakan persamaan dari fungsi ReLU.

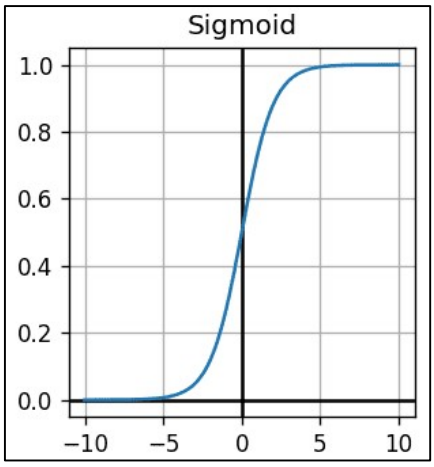
$$y = f(x) = max(0, x)$$



Gambar 2.4 Fungsi ReLU

Selain itu, fungsi aktivasi yang juga umum digunakan adalah Fungsi Aktivasi Sigmoid Biner. Fungsi ini memiliki nilai pada range 0 sampai 1. Berikut merupakan persamaan dari fungsi sigmoid.

$$y = f(x) = \frac{1}{1 + e^{-x}}$$



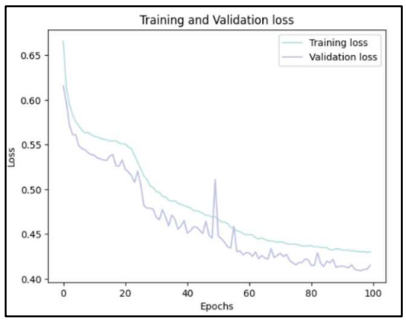
Gambar 2.5 Fungsi Sigmoid

1. HASIL DAN DISKUSI

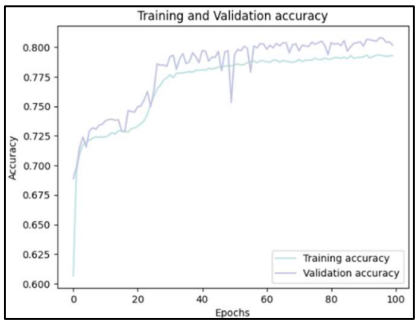
3. 1. Arsitektur *Artificial Neural Network*

Dalam melakukan analisis klasifikasi pasien diabetes yang melakukan readmisi, digunakan arsitektur *Artificial Neural Network* yang terdiri dari beberapa lapisan dengan fungsi dan karakteristik spesifik. Lapisan pertama adalah *input layer* yang memiliki 32 neuron dan menggunakan fungsi aktivasi ReLU untuk memproses 42 fitur input, membantu menangkap hubungan non-linear dalam data. *Hidden layer* pertama dan kedua, masing-masing dengan 32 dan 16 neuron, juga menggunakan ReLU untuk meningkatkan kemampuan model dalam menangkap kompleksitas data. Setelah itu, terdapat *dropout layer* pertama dengan rasio 0.25, yang secara acak menonaktifkan 25% neuron selama pelatihan untuk mencegah overfitting. *Hidden layer* ketiga memiliki 8 neuron dengan ReLU, diikuti oleh *dropout layer* kedua dengan rasio 0.5 untuk lebih lanjut mengurangi overfitting dengan menonaktifkan 50% neuron secara acak. Terakhir, *output layer* terdiri dari satu neuron dengan fungsi aktivasi sigmoid yang mengubah output menjadi probabilitas. Hal tersebut karena arsitektur ANN ini digunakan untuk klasifikasi biner. Model ini dikompilasi dengan *optimizer* Adam dan menggunakan *binary crossentropy* sebagai fungsi *loss*, serta akurasi sebagai metrik evaluasi, dan dilengkapi dengan mekanisme *early stopping* untuk menghentikan pelatihan ketika tidak ada peningkatan performa signifikan.

3. 2. Plot Training dan Validation



Gambar 3.1 Plot *Loss Training Validation*



Gambar 3.2 Plot A*ccuracy Training Validation*

Berdasarkan gambar 3.1 dan 3.2, nilai kerugian (loss) pada data pelatihan dan validasi menurun seiring dengan jumlah epoch. *Training loss* terus menurun, menunjukkan model yang semakin baik dalam mempelajari pola dari data pelatihan. *Validation loss* juga menurun dengan beberapa fluktuasi, mengindikasikan risiko overfitting, tetapi secara keseluruhan menunjukkan kemampuan generalisasi yang cukup baik. Sementara itu, akurasi model pada data pelatihan dan validasi meningkat seiring bertambahnya epoch.

Training accuracy menunjukkan peningkatan konsisten, sedangkan garis hijau *Validation accuracy* juga meningkat meskipun dengan fluktuasi yang lebih besar, yang bisa disebabkan oleh overfitting. Namun, peningkatan keseluruhan dalam akurasi validasi menunjukkan kemampuan generalisasi model yang baik.

3. 3. Hasil Akurasi pada Data Test

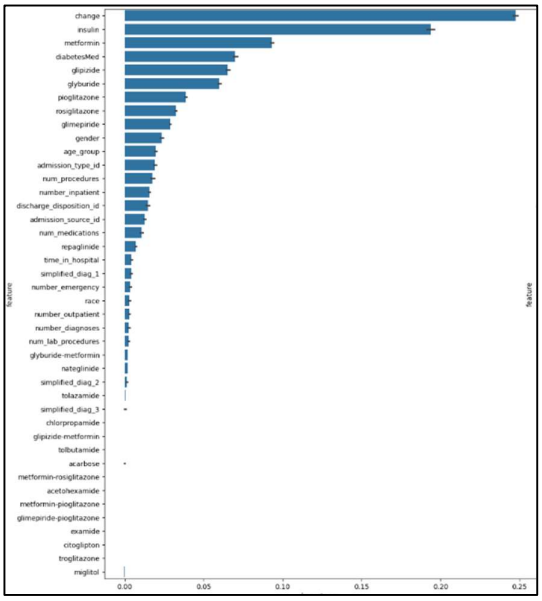
Model ANN yang telah dibentuk kemudian digunakan untuk memprediksi data *test*.

	precision	recall	f1-score	support
0	0.79	0.83	0.81	18079
1	0.82	0.78	0.80	18085
accuracy			0.80	36164
macro avg	0.81	0.80	0.80	36164
weighted avg	0.81	0.80	0.80	36164

Gambar 3.3 Metrik Evaluasi Data Test

Gambar 3.3 menunjukkan bahwa model memiliki akurasi keseluruhan sebesar 80%, dengan presisi, recall, dan f1-score yang seimbang antara dua kelas. Kelas 0 memiliki presisi 0,79, recall 0,83, dan f1-score 0,81, sementara kelas 1 memiliki presisi 0,82, recall 0,78, dan f1-score 0,80. Hasil ini menunjukkan bahwa model cukup baik dalam memprediksi kedua kelas.

3. 4. Feature Importance



Gambar 3.4 Plot *Feature Importance*

Berdasarkan gambar 3.4 dapat disimpulkan mengenai beberapa fitur penting yang digunakan dalam model untuk melakukan klasifikasi pasien diabetes yang mengalami readmisi, diantaranya adalah

1. Change: Fitur ini adalah yang paling penting dalam model. Ini mengindikasikan bahwa perubahan dalam pengobatan sangat mempengaruhi hasil prediksi model.
2. Insulin dan metamorfin: Ini menunjukkan apakah insulin dan metamorfin diberikan kepada pasien dan bagaimana dosisnya berubah. Pentingnya fitur ini menunjukkan bahwa penggunaan insulin dan metamorfin adalah faktor kritis dalam prediksi model.
3. diabetesMed: Fitur ini menunjukkan apakah ada obat diabetes yang diresepkan kepada pasien. Pentingnya fitur ini menunjukkan bahwa pengobatan diabetes secara umum sangat relevan dalam prediksi model.
4. glipizide, glyburide, pioglitazone, rosiglitazone, glimepiride**:** Obat-obatan diabetes lainnya juga memiliki peran yang signifikan, menunjukkan bahwa jenis dan perubahan dosis obat-obatan ini mempengaruhi hasil prediksi model.
5. gender**:** Pentingnya fitur ini menunjukkan bahwa jenis kelamin pasien mempengaruhi hasil prediksi. Ini mungkin terkait dengan perbedaan biologis atau perilaku kesehatan antara pria dan wanita.
6. age_group**:** Pentingnya fitur ini menunjukkan bahwa umur adalah faktor penting dalam prediksi, mungkin karena risiko komplikasi diabetes yang meningkat dengan bertambahnya usia.

Fitur-fitur lain yang lebih rendah dalam grafik menunjukkan kontribusi yang lebih kecil terhadap model. Misalnya, admission_type_id, num_procedures**,** number_inpatient**,** discharge_disposition_id, admission_source_id, num_medications, num_medications, time_in_hospital, race, number_outpatient, dan number_lab_procedures . Hal ini menunjukkan bahwa fitur-fitur tersebut kurang berpengaruh dalam prediksi model dibandingkan fitur utama yang telah disebutkan.

4. KESIMPULAN DAN SARAN

4.1 Kesimpulan

Berdasarkan analisis yang telah dilakukan pada bagian 3, dapat disimpulkan bahwa klasifikasi readmisi pasien dengan penyakit diabetes dapat dilakukan menggunakan model *Artificial Neural Network*. Model ini memiliki akurasi keseluruhan sebesar 80%, dengan presisi, recall, dan f1-score yang seimbang antara dua kelas. Dari model tersebut kemudian didapatkan fitur-fitur penting yang digunakan dalam mengklasfikasikan readmisi pasien dengan penyakit diabetes. Fitur-fitur tersebut diantaranya adalah perubahan pengobatan pada pasien, pemberian dosis obat generik seperti insulin, metamorfin, glipizide, glyburide, pioglitazone, rosiglitazone, dan glimepiride. Selain itu, jenis kelamin dan umur juga termasuk fitur penting dalam mengklasifikasi readmisi pasien diabetes.

4.2 Saran

Berdasarkan analisis, berikut saran yang dapat dilakukan untuk mengurangi angka readmisi pasien diabetes dan meningkatkan kualitas perawatan serta manajemen penganganan penyakit diabetes

1. Pengoptimalan Pengobatan**:** Menggunakan informasi tentang perubahan pengobatan dan dosis obat generik seperti insulin, metformin, glipizide, glyburide, pioglitazone, rosiglitazone, dan glimepiride untuk menyesuaikan rencana pengobatan yang lebih efektif dan mencegah readmisi.
2. Personalisasi Perawatan: Mempertimbangkan jenis kelamin dan umur pasien dalam perencanaan perawatan untuk menyediakan pendekatan yang lebih personal dan tepat sasaran,
3. Pemantauan Terus Menerus**:** Menerapkan sistem pemantauan berkelanjutan bagi pasien yang memiliki perubahan pengobatan atau yang diberikan dosis obat generik tertentu untuk mendeteksi risiko readmisi lebih awal.
4. Integrasi Data: Mengintegrasikan data dari berbagai sumber seperti catatan medis elektronik untuk memperbarui model ANN secara berkala dan meningkatkan akurasi prediksi.

DAFTAR PUSTAKA

Julpan, Nababan E. B., & Zarlis, M. (2015). Analisis Fungsi Aktivasi Sigmoid Biner Dan Sigmoid Bipolar Dalam Algoritma Backpropagation Pada Prediksi Kemampuan Siswa. Jurnal Teknovasi Volume 02, Nomor 1. https://core.ac.uk/download/pdf/235004108.pdf

Putra, J. W. G. (2020). Pengenalan Konsep Pembelajaran Mesin dan Deep Learning Edisi 1.4. https://wiragotama.github.io/resources/ebook/parts/JWGP-intro-to-ml-chap11-secured.pdf

LAMPIRAN

```
import pandas as pd
df = pd.read_csv("C:/Users/Lenovo/Downloads/diabetic_data.csv")
df['readmitted'] = df['readmitted'].replace({'>30': 'NO', '<30': 'YES'})
df.replace("?", pd.NA, inplace=True)
missing_percentage = df.isna().mean() * 100
print(missing_percentage)
#Visualisasi sebaran missing value
import missingno as msno
import matplotlib.pyplot as plt
msno.matrix(df)
plt.show()
msno.bar(df)
plt.show()
#Menghapus variabel yang presentase missing value nya lebih dari 35%
df.drop(columns = ['weight'],axis = 1, inplace = True)
df.drop(columns = ['payer_code'],axis = 1, inplace = True)
df.drop(columns = ['medical_specialty'],axis = 1, inplace = True)
df.drop(columns = ['max_glu_serum'],axis = 1, inplace = True)
df.drop(columns = ['A1Cresult'],axis = 1, inplace = True)
#Imputasi missing value variabel kategorik dengan modus dari setiap kelas
kategori
def fill_na_by_mode(df, column, reference):
    # Hitung modus dari setiap kategori di kolom column berdasarkan
kategori di kolom reference
    mode_per_reference = df.groupby(reference)[column].apply(lambda x:
x.mode()[0])

    # Fungsi untuk mengisi nilai NA di kolom column berdasarkan kategori
di kolom reference
    def fill_na(row):
        if pd.isna(row[column]):
            return mode_per_reference[row[reference]]
        else:
            return row[column]

    # Mengisi nilai NA di kolom column berdasarkan kategori di kolom
reference
    df[column] = df.apply(fill_na, axis=1)

    return df

# Contoh penggunaan fungsi untuk mengisi nilai NA di kolom "gender"
berdasarkan kategori di kolom "readmitted"
df = fill_na_by_mode(df, 'gender', 'readmitted')
df = fill_na_by_mode(df, 'race', 'readmitted')
df = fill_na_by_mode(df, 'diag_1', 'readmitted')
df = fill_na_by_mode(df, 'diag_2', 'readmitted')
df = fill_na_by_mode(df, 'diag_3', 'readmitted')
# Fungsi untuk mengkategorikan diagnosis
def categorize_diagnosis(diag):
    if 'V' in diag or 'E' in diag:
        return 'Other'
    elif '250' in diag:
        return 'Diabetes'
    else:
        try:
            diag_int = int(diag)
            if 390 <= diag_int <= 459 or diag_int == 785:
```

```
                return 'Circulatory'
            elif 460 <= diag_int <= 519 or diag_int == 786:
                return 'Respiratory'
            elif 520 <= diag_int <= 579 or diag_int == 787:
                return 'Digestive'
            elif 580 <= diag_int <= 629 or diag_int == 788:
                return 'Genitourinary'
            elif 140 <= diag_int <= 239:
                return 'Neoplasms'
            elif 710 <= diag_int <= 739:
                return 'Musculoskeletal'
            elif 800 <= diag_int <= 999:
                return 'Injury'
            else:
                return 'Other'
        except ValueError:
            return 'Other'

df['simplified_diag_1'] = df['diag_1'].apply(categorize_diagnosis)
df['simplified_diag_2'] = df['diag_2'].apply(categorize_diagnosis)
df['simplified_diag_3'] = df['diag_3'].apply(categorize_diagnosis)
#menghapus kolom diag_1 diag_2 diag_3 karena sudah digantikan dengan
primary diagnosis
df.drop(columns=['diag_1'], inplace=True)
df.drop(columns=['diag_2'], inplace=True)
df.drop(columns=['diag_3'], inplace=True)
#melihat persebaran age disetiap kategori
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='age', hue='readmitted')
plt.title('Distribution of Age by Readmitted Status')
plt.xlabel('Age Group')
plt.ylabel('Count')
plt.legend(title='Readmitted')
plt.show()
def categorize_age(age):
    if age in ['[0-10)', '[10-20)', '[20-30)', '[30-40)', '[40-50)']:
        return 'Dibawah 50'
    else:
        return 'Diatas 50'

# Menambahkan kolom age_group dan menghapus kolom age
df['age_group'] = df['age'].apply(categorize_age)
df.drop(columns=['age'], inplace=True)
#drop kolom encounter_id dan patient_nbr karena tidak akan digunakan
dalam modelling
df.drop(columns=['encounter_id'], inplace=True)
df.drop(columns=['patient_nbr'], inplace=True)
#melakukan label encoding pada kolom kategorik
from sklearn.preprocessing import LabelEncoder
object_data = df.select_dtypes(include=['object']).copy()
numeric_data = df.select_dtypes(include=['number']).copy()

# Menggunakan Label Encoder untuk data objek
le = LabelEncoder()
for col in object_data.columns:
    object_data[col] = le.fit_transform(object_data[col])

# Menggabungkan data objek dan numerik
processed_data = pd.concat([numeric_data, object_data], axis=1)
X = processed_data.drop(['readmitted'],axis=1)
y = processed_data['readmitted']
```

```
# Hitung jumlah nilai "yes" dan "no" dalam kolom "readmitted"
readmitted_counts = df['readmitted'].value_counts()

# Plot barplot
plt.figure(figsize=(8, 6))
sns.barplot(x=readmitted_counts.index, y=readmitted_counts.values)
plt.xlabel('Readmitted')
plt.ylabel('Count')
plt.title('Barplot of Readmitted')
plt.show()
#balancing data dengan smote
from imblearn.over_sampling import SMOTE
smote = SMOTE()
# Lakukan oversampling
X_resampled, y_resampled = smote.fit_resample(X, y)
import pandas as pd
import sklearn.neural_network as ann
import sklearn.model_selection as ms
X_train, X_test, y_train, y_test =
ms.train_test_split(X_resampled,y_resampled,test_size=0.2)
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from keras import callbacks
from keras.optimizers import Adam

early_stopping = callbacks.EarlyStopping(
    min_delta=0.001, # minimium amount of change to count as an
improvement
    patience=20, # how many epochs to wait before stopping
    restore_best_weights=True,
)

# Initialising the NN
model = Sequential()

# layers

model.add(Dense(units = 32, kernel_initializer = 'uniform', activation =
'relu', input_dim = 42))
model.add(Dense(units = 32, kernel_initializer = 'uniform', activation =
'relu'))
model.add(Dense(units = 16, kernel_initializer = 'uniform', activation =
'relu'))
model.add(Dropout(0.25))
model.add(Dense(units = 8, kernel_initializer = 'uniform', activation =
'relu'))
model.add(Dropout(0.5))
model.add(Dense(units = 1, kernel_initializer = 'uniform', activation =
'sigmoid'))

# Compiling the ANN
opt = Adam(learning_rate=0.00009)
model.compile(optimizer = opt, loss = 'binary_crossentropy', metrics =
['accuracy'])
# Train the ANN
history = model.fit(X_train, y_train, batch_size = 32, epochs = 100,
callbacks=[early_stopping], validation_split=0.2)
history_df = pd.DataFrame(history.history)
plt.plot(history_df.loc[:, ['loss']], "#BDE2E2", label='Training loss')
```

```
plt.plot(history_df.loc[:, ['val_loss']],"#C2C4E2", label='Validation
loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(loc="best")
plt.show()
history_df = pd.DataFrame(history.history)
plt.plot(history_df.loc[:, ['accuracy']], "#BDE2E2", label='Training
accuracy')
plt.plot(history_df.loc[:, ['val_accuracy']], "#C2C4E2",
label='Validation accuracy')
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
# Predicting the test set results
from sklearn.metrics import precision_score, recall_score,
confusion_matrix, classification_report, accuracy_score, f1_score
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5)
print(classification_report(y_test, y_pred))
from sklearn.inspection import permutation_importance
import shap
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from keras import callbacks
from keras.optimizers import Adam
from sklearn.base import BaseEstimator, ClassifierMixin

class KerasClassifierWrapper(BaseEstimator, ClassifierMixin):
    def __init__(self, model):
        self.model = model

    def fit(self, X, y):
        self.model.fit(X, y, batch_size=32, epochs=100,
callbacks=[early_stopping], validation_split=0.2)
        return self

    def predict(self, X):
        return (self.model.predict(X) > 0.5).astype("int32")

    def score(self, X, y):
        return self.model.evaluate(X, y, verbose=0)[1]

def plot_fi(model, X, y):
    # Compute permutation importance
    perm = permutation_importance(model, X, y, n_repeats=10,
random_state=0, n_jobs=-1)

    # Organize permutation importance results into a DataFrame
    perm2 = pd.DataFrame({'feature': X.columns.tolist()*10, 'importance':
perm["importances"].transpose().reshape(-1)})
    perm2["importance"] = perm2.importance/perm2.importance.sum()*10
    urut = perm2.groupby("feature").mean().sort_values("importance",
ascending=False)
```

```
    # Plotting
    plt.figure(figsize=(8, 6))
    fig, axs = plt.subplots(1, 2, figsize=(25, 15))

    sns.barplot(x='importance', y='feature', data=perm2,
order=urut.index, ax=axs[0])
    sns.stripplot(x='importance', y='feature', data=perm2,
order=urut.index, s=5, ax=axs[1])
    plt.show()

    return urut
# Build and compile the neural network
early_stopping = callbacks.EarlyStopping(min_delta=0.001, patience=20,
restore_best_weights=True)
model = Sequential()
model.add(Dense(units=32, kernel_initializer='uniform',
activation='relu', input_dim=X_train.shape[1]))
model.add(Dense(units=32, kernel_initializer='uniform',
activation='relu'))
model.add(Dense(units=16, kernel_initializer='uniform',
activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(units=8, kernel_initializer='uniform',
activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(units=1, kernel_initializer='uniform',
activation='sigmoid'))

opt = Adam(learning_rate=0.00009)
model.compile(optimizer=opt, loss='binary_crossentropy',
metrics=['accuracy'])

# Wrap the Keras model
wrapped_model = KerasClassifierWrapper(model)

# Train the wrapped model
wrapped_model.fit(X_train, y_train)

# Use the plot_fi function to display feature importance
plot_fi(wrapped_model, X_test, y_test)
```