Prediksi Hospital Readmission Menggunakan Machine Learning

Nova Lailatul Rizkiyah, Devi Noor Endrawati, Amirul Mu'minin

1. Background

Hospital readmission atau penerimaan kembali di rumah sakit adalah ketika seorang pasien keluar dari rumah sakit diterima kembali dalam periode waktu tertentu. Hospital readmission merupakan salah satu indikator kualitas rumah sakit yang diberikan oleh *Centers for Medicare & Medicaid Services*. Semakin tinggi nilai hospital readmission pada waktu tertentu, semakin rendah nilai perawatan/pelayanan yang baik. Pada tahun 2011, rumah sakit di Amerika Serikat menghabiskan lebih dari 41 miliar dolar akibat pasien diabetes yang mengalami hospital readmission dalam waktu kurang dari 30 hari setelah dipulangkan. Mengingat pentingnya parameter Hospital readmission, kami menggunakan data dari 130 rumah sakit di Amerika Serikat dari Machine Learning Repository UCI. Darisana kami akan mencoba untuk memprediksi faktor-faktor apa saja yang paling berpengaruh dari hospital readmission dalam rentang waktu 30 hari dan memprediksi seberapa akurat klasifikasi yang digunakan.

Identifikasi awal pasien yang dihadapi risiko penerimaan kembali yang tinggi dapat memungkinkan penyedia layanan kesehatan untuk melakukan penyelidikan tambahan dan mungkin mencegah readmissions di mas depan. Ini tidak hanya meningkatkan kualitas perawatan tetapi juga mengurangi biaya medis pada saat masuk kembali. Pembelajaran mesin metode telah dimanfaatkan pada data kesehatan masyarakat untuk membangun sebuah sistem untuk mengidentifikasi pasien diabetes yang menghadapi risiko tinggi di masa depan pendaftaran kembali.

2. Objective

Klasifikasi adalah proses menemukan kumpulan pola atau fungsi-fungsi yang mendeskripsikan dan memisahkan kelas data satu dengan lainnya, dapat digunakan untuk memprediksi data yang belum memiliki kelas data tertentu (Han dan Kamber, 2001). Klasifikasi digunakan untuk memprediksi label kelas yang bersifat kategori.

Dataset merupakan data pasien diabetes selama tahun 1999-2008 yang diambil dari 130 rumah sakit di Amerika Serikat. Banyaknya atribut sebanyak 50 meliputi pertemuan rawat inap, gender, usia, obat, tes lab dsb. Instance sekitar 100,000. Sebelum data diolah, lebih dahulu dipreprocessing. Tujuan dari tugas besar ini adalah memprediksi dan mengklasifikasi Hospital readmission pasien diabetes sebelum 30 hari atau tidak.

3. Related Work

Menurut [1] Number inpatient visit), discharge dispotition dan admission type merupakan strong predictor(factor untuk memprediksi) dari readmission. Selanjutnya, number of laboratory test, discharge dispotition dapat memprediksi secara cepat apakah pasien akan readmission kurang dari 30 hari atau setelah 30 hari atau tidak sama sekali sehingga didapat performansi algoritma yang didapat dibawah ini.

Tabel 1 . Membandingkan performansi algoritma yang berbeda pada pasien yang berisiko tinggi.

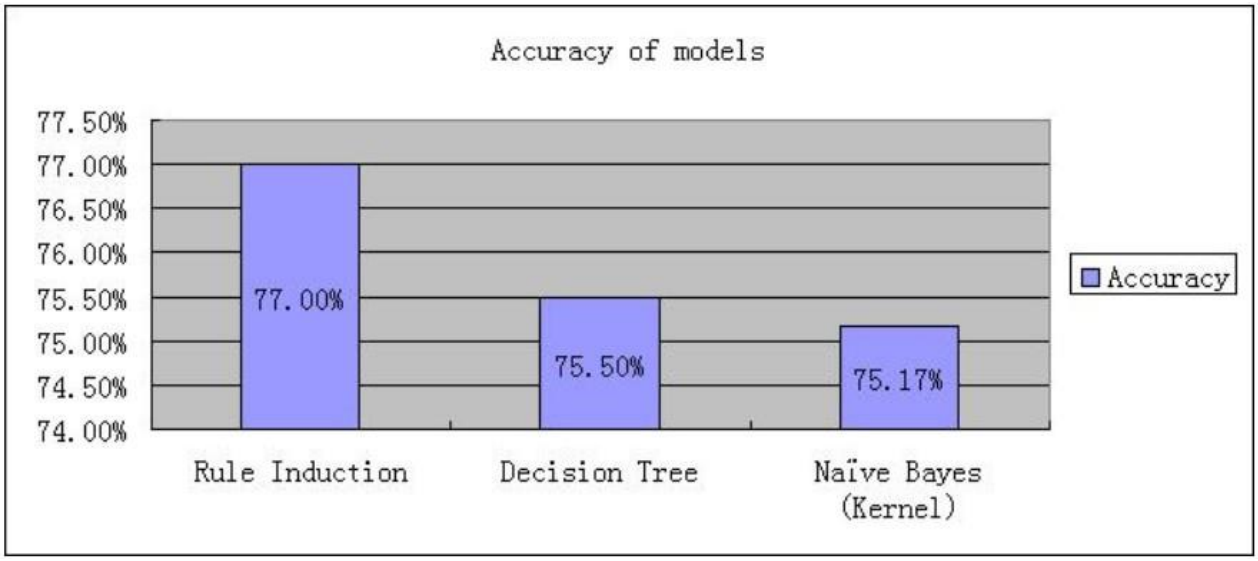
Classifier	Area Under Precision-Recall Curve	
	Class <30	Class <30 + Class >30
Naive Bayes	0.214	0.63
Bayes Network	0.208	0.637
Random Forest	0.242	0.65
Adaboost Trees	0.167	0.569
Neural Networks	0.233	0.654

Dalam jurnal [2] menjelaskan korelasi feature terhadap radmission ditunjukkan pada table dibawah ini.

Tabel 2. Korelasi features dengan readmission

Feature	Value of correlation
Discharge disposition	0.3356
Admission type	0.3012
Number of procedures	0.0888
Number of lab procedures	0.0558
Metformin	-0.0506
Insulin	-0.0704
Glucose serum test result	-0.0748
Number of outpatient visits	-0.0823
Number of medications	-0.0933
Time in hospital	-0.0934
Number of emergency visits	-0.0989
Number of diagnoses	-0.1118
Admission source	-0.1157
Pioglitazone	-0.1278
Number of inpatient visits	-0.2798

Didapatkan nilai akurasi masing-masing algoritma di bawah ini.

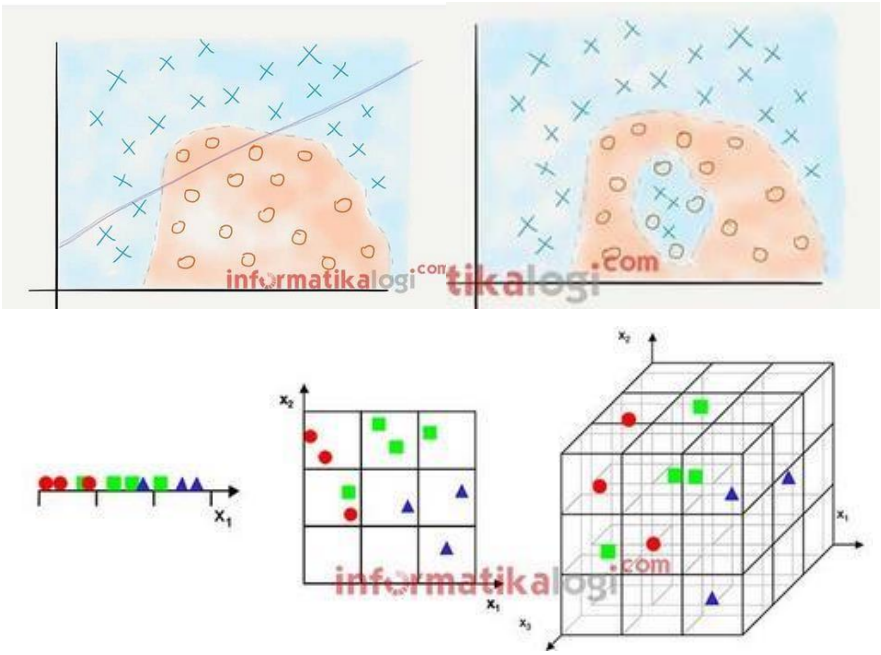


Gambar 1. Akurasi dari model

4. Proposed Method

Ada beberapa algoritma yang digunakan:

a. KNN (K-Nearest Neigbor) algoritma yang menyatakan mayoritas kedekatan jarak dengan tetangga. Misal data baru x diantara 6 instans berlabel y dan 3 instans berlabel z dengan k=9. Algoritma ini menghitung jarak terkecil dengan instans (kuadrat jarak)



Gambar 2. Penggambaran penerapan algoritma KNN

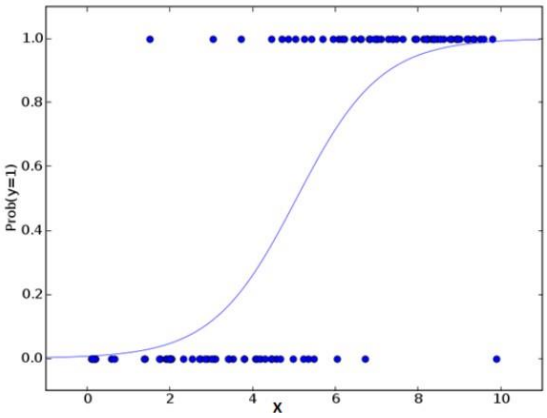
- Kelebihan: sangat non linear. Karena berdasarkan model nonparametrik: model yang tidak mengasumsikan apa-apa mengenai distribusi instance dalam dataset Mudah dipahami dan diimplementasikan, fungsi penghitung jarak kedekatan
- Kekurangan: perlu menunjukkan parameter k(seberapa banyak tetangga terdekat yang diambil), tidak menangani nilai yang hilang, sensitive terhadap data pencilan, rentan terhadap variable yang noninformatif, rentan terhadap dimensionalitas yang tinggi, rentan terhadap perbedaan rentang variable

b. Naïve Bayes : merupakan algoritma yang menunjukkan peluang dari suatu kejadian. Tingkat akurasi cukup tinggi yang juga merupakan teknik klasifikasi berdasarkan Teorema Bayes yang semua fitur-fiturnya diasumsikan independen satu sama lain yang berkonstribusi pada probabilitas.

- Kelebihan: Jumlah data tidak terlalu banyak

- Kekurangan: asumsi variable independen atau tidak bergantung satu sama lain, tidak berlaku jika probabilitas kondisionalnya adalah nol, apabila nol maka probabilitas prediksi akan bernilai nol juga

c. Logistik Regression: Ini adalah metode statistik untuk menganalisis set data di mana ada satu atau lebih variabel independen yang menentukan hasil. Hasilnya diukur dengan variabel dikotomis (di mana hanya ada dua hasil yang mungkin). Tujuan dari regresi logistik adalah untuk menemukan model fitting terbaik untuk menggambarkan hubungan antara karakteristik dikotomis yang menarik (variabel dependen = respon atau variabel hasil) dan satu set variabel independen (prediktor atau penjelas). Ini lebih baik daripada klasifikasi biner lainnya seperti tetangga terdekat karena juga menjelaskan secara kuantitatif faktor-faktor yang menyebabkan klasifikasi.

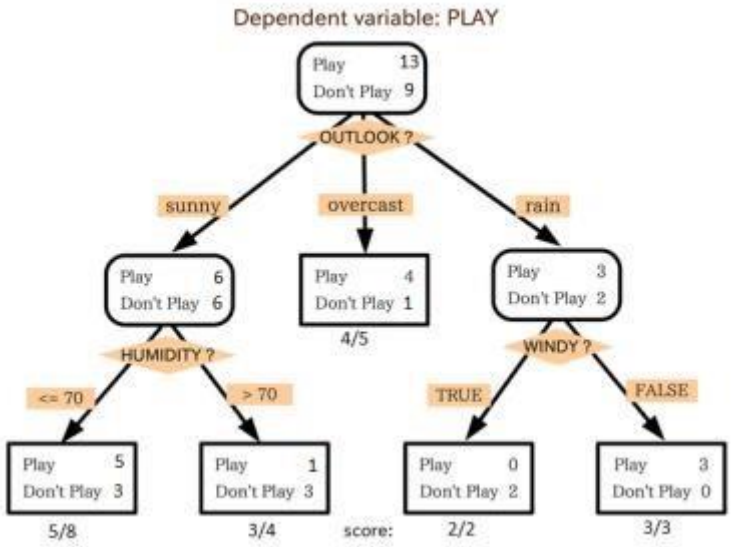


Gambar 3. Grafik Algoritma Logit

- Kelebihan: model paling efektif dan efisien dalam prediksi cacat sautu program.
- Kekurangan: rentan terhadap *underfitting* pada dataset yang kelasnya tidak seimbang, sehingga akan menghasilkan akurasi yang rendah

d. Decision Tree: Decision tree membangun model klasifikasi atau regresi dalam bentuk struktur pohon. Ini memecah kumpulan data menjadi himpunan bagian yang lebih kecil dan lebih kecil sementara pada saat yang sama pohon keputusan terkait dikembangkan secara bertahap. Hasil akhirnya adalah pohon dengan simpul keputusan dan simpul daun. Node keputusan memiliki dua atau lebih cabang dan node daun mewakili klasifikasi atau keputusan. Node keputusan teratas dalam pohon yang sesuai dengan prediktor terbaik disebut simpul akar. Pohon keputusan dapat menangani data kategorikal dan numerik.

- Kelebihan: Daerah pengambilan keputusan yang sebelumnya kompleks dan sangat global, dapat diubah menjadi lebih simpel dan spesifik dengan mengeliminasi kelas yang tidak diperlukan, selain itu fleksibel untuk memilih fitur dari internal node yang berbeda, fitur yang terpilih akan membedakan suatu kriteria dibandingkan kriteria yang lain dalam node yang sama.
- Kekurangan: Terjadi overlap terutama ketika kelas-kelas dan criteria yang digunakan jumlahnya sangat banyak. Hal tersebut juga dapat menyebabkan meningkatnya waktu pengambilan keputusan dan jumlah memori yang diperlukan. Pengakumulasian jumlah eror dari setiap tingkat dalam sebuah pohon keputusan yang besar. Kesulitan dalam mendesain pohon keputusan yang optimal. Hasil kualitas keputusan yang didapatkan dari metode pohon keputusan sangat tergantung ada bagaimana pohon tersebut didesain.



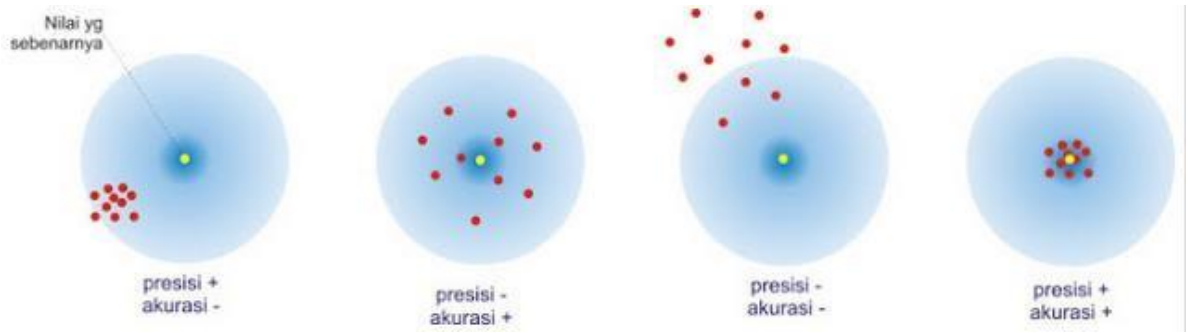
Gambar 4. Contoh Tree di sebuah kasus Klasifikasi

e. Random Forest: Random forest atau random decision forest adalah metode pembelajaran ensembel untuk klasifikasi, regresi dan tugas-tugas lain, yang beroperasi dengan membangun banyak pohon keputusan pada waktu pelatihan dan menghasilkan kelas yang merupakan mode kelas (klasifikasi) atau prediksi rata-rata (regresi)) dari masing-masing pohon. Hutan keputusan acak dikoreksi karena kebiasaan pohon keputusan terlalu pas dengan set pelatihan mereka.
   - Kelebihan: Random forest memiliki varian yang lebih sedikit daripada decision tree. Artinya, algoritma ini bekerja dengan benar untuk sejumlah besar item data daripada decision tree. Random forest sangat fleksibel dan memiliki akurasi sangat tinggi.

Random forest tidak memerlukan persiapan input data. Anda tidak perlu mengukur data. Itu juga menjaga akurasi bahkan ketika sebagian besar data hilang.

- Kekurangan: Random forest sangat kompleksi. Mereka jauh lebih sulit dan memakan waktu untuk dibangun daripada decision tree. Random forest juga membutuhkan lebih banyak sumber daya komputasi dan juga kurang intuitif. Ketika Anda memiliki banyak koleksi decision tree, sulit untuk memiliki pemahaman intuitif tentang hubungan yang ada dalam data input. Selain itu, proses prediksi menggunakan random forest memakan waktu daripada algoritma lainnya.

Evaluasi Metrik



Gambar 5. Penggambaran evaluasi Metrik

- Presisi = Jika terjadi pengulangan, selalu tepat di nilai tersebut
- Recall = Tingkat keberhasilan system dalam menemukan kembali informasi
- Akurasi = Kedekatan nilai prediksi dan nilai sebenarnya • F1-score = Perbandingan rata-rata presisi dan recall

Tabel 3. Contoh hasil prediksi mahasiswa potensi DO

Nim	Status Sebenarnya	Hasil Prediksi
001	Tidak DO	Tidak DO
002	Tidak DO	Tidak DO
003	Tidak DO	Tidak DO
004	Tidak DO	DO
005	Tidak DO	DO
006	DO	Tidak DO
007	DO	DO
008	DO	DO
009	DO	DO
010	DO	DO

Dari hasil prediksi di atas, hanya ada kemungkinan 4 kasus yang terjadi:

- True Positive (TP) : kasus dimana mahasiswa diprediksi (Positif) DO, memang benar(True) DO.Nilai TP nya adalah 4
- True Negative (TN) : kasus dimana mahasiswa diprediksi tidak(Negatif) DO dan sebenarnya mahasiswa tersebut memang (True) tidak DO. TN nya adalah 3
- False Positve (FP) : kasus dimana mahasiswa yang diprediksi positif DO, ternyata tidak DO. Prediksinya salah (False). FP adalah 2
- False Negatif (FN): kasus dimana mahasiswa yang diprediksi tidak DO (Negatif), tetapi ternyata sebenarnya(TRUE) DO. FN = 1

Cara mudah untuk mengingatnya adalah :

1. Jika diawali dengan TRUE maka prediksinya benar, entah diprediksi terjadi atau tidak terjadi. Dalam kasus di atas, entah diprediksi DO atau Tidak DO, data sebenarnya adalah seperti itu.
2. Jika diawali dengan FALSE maka menyatakan prediksinya salah
3. Positif dan Negatif merupakan hasil prediksi dari program.

Pengukuran Performance

1. Accuracy

Merupakan rasio prediksi Benar (positif dan negatif) dengan keseluruhan data. Akurasi menjawab pertanyaan “Berapa persen mahasiswa yang benar diprediksi DO dan Tidak DO dari kesuluruhan mahasiswa”

Akurasi = (TP + TN) / (TP+FP+FN+TN)
Akurasi = (4+3) / (4+2+1+3) = 7/10 = 70%

2. Precission

Merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positf. Precission menjawab pertanyaan “Berapa persen mahasiswa yang benar DO dari keseluruhan mahasiswa yang diprediksi DO?”

Precission = (TP) / (TP+FP)

Precission = 4 / (4+2) = 4/6 = 67%.

3. Recall (Sensitifitas)

Merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Recall menjawab pertanyaan “Berapa persen mahasiswa yang diprediksi DO dibandingkan keseluruhan mahasiswa yang sebenarnya DO”.

Recall = (TP) / (TP + FN) ❼ Recall = 4/(4+1) = 4/5 =80%.

4. Specificity

Merupakan kebenaran memprediksi negatif dibandingkan dengan keseluruhan data negatif. Specificity menjawab pertanyaan “Berapa persen mahasiswa yang benar diprediksi tidak DO dibandingkan dengan keseluruhan mahasiswa yang sebenarnya tidak DO”.

Specificity = (TN)/ (TN + FP)
Specificity = 3/(3+2) = 3/5 = 60%

5. F1 Score

F1 Score merupakan perbandingan rata-rata presisi dan recall yang dibobotkan
F1 Score = 2 * (Recall*Precission) / (Recall + Precission) F1
Score = 2* (80%*67%) / (80% + 67%) = 72,93%

Penerapan dalam Data Prediksi Hospital Readmission

Pilih algoritma yang memiliki Accuracy tinggi jika

Akurasi sangat bagus kita gunakan sebagai acuan performansi algoritma JIKA dataset kita memiliki jumlah data False Negatif dan False Positif yang sangat mendekati (Symmetric). Namun jika jumlahnya tidak mendekati, maka sebaiknya gunakan F1 Score sebagai acuan.

Pada contoh di atas, nilai FP dan FN mendekati, maka accuracy bisa digunakan sebagai acuan ukuran performansi tersebut.

Pilih algoritma yang memiliki Recall tinggi jika kita lebih memilih

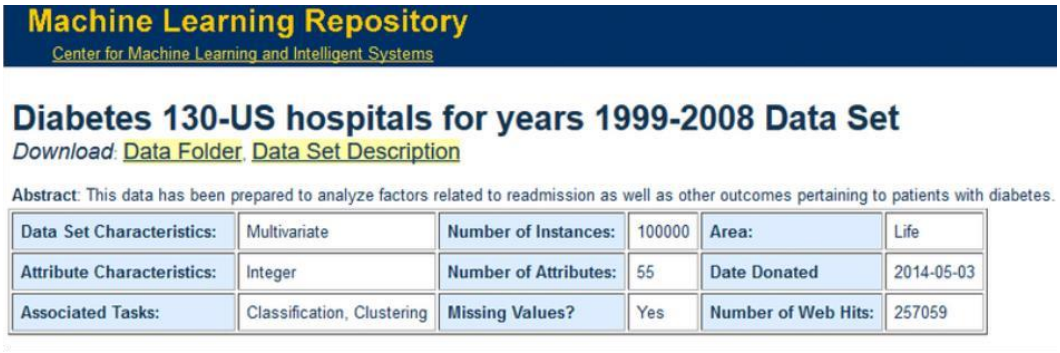
False Positif lebih baik terjadi daripada False Negatif.

Dalam contoh di atas, maka kita mempertimbangkan Recall karena lebih baik algoritma kita memprediksi mahasiswa positif DO tetapi sebenarnya tidak DO daripada algoritma salah memprediksi bahwa mahasiswa diprediksi tidak DO padahal sebenarnya dia DO. **Pilih algoritma yang memiliki Precission tinggi jika**

kita lebih menginginkan terjadinya True Positif dan sangat tidak menginginkan terjadinya False Positif. Contohnya adalah pada kasus klasifikasi email SPAM atau tidak. Kita lebih memilih jika email yang sebenarnya SPAM namun diprediksi tidak SPAM (sehingga tetap ada pada kotak masuk email kita), daripada email yang sebenarnya bukan SPAM tapi diprediksi SPAM (sehingga tidak ada pada kotak masuk kita) **Pilih algoritma yang memiliki Specificity tinggi jika**

kita tidak menginginkan terjadinya false positif. Contohnya pada prediksi apakah seseorang kecanduan narkoba atau tidak maka kita sangat mengharapkan tidak terjadi salah mendeteksi orang yang sebenarnya negatif tapi dinyatakan positif. Kasihan dia masuk penjara padahal tidak kecanduan narkoba.

Data Hospital Readmitted



Gambar 6. Data diambil dari UCI Machine Learning Repository

Tabel 4. Penjelasan dan Nilai *Missing Value* pada masing-masing Feature

Nama Feature	Tipe	Deskripsi dan Nilai	% missing
Encounter ID	Numerik	Nomor ID masuk rumah sakit	0%
Patient Number	Numerik	Nomor Identifikasi Pasien	0%
Race	Nominal	Values: Caucasian, Asian, African, American	2%
Gender	Nominal	Values: Laki-laki, Perempuan, unknown/invalid	0%
Age	Nominal	Grup umur 10 tahun interval	0%
Weight	Numerik	Berat badan satuan pounds	97%
Admission Type	Nominal	Identifier Integer yang sesuai pada 9 nilai berbeda: darurat, mendesak baru lahir dll	0%
Discharge Dispotition	Nominal	Identifier integer yang sesuai dengan 29 nilai: discharged to home, expired	0%
Admission Source	Nominal	Identifier integer yang sesuai dengan 21 nilai: emergency room, physician referral	0%
Time in hospital	Numerik	banyaknya hari diantara masuk dan keluar rumah sakit	0%
Payer Code	Nominal	Identifier Integer dengan 23 nilai seperti self pay, medicare	0%
Medical Specialty	Nominal	Values: kardiologi, internal medicine, operasi dsb	53%

Number of Lab Procedures	Numerik	Jumlah tes lab yang dilakukan selama pertemuan	0%
Number of procedures	Numerik	Jumlah prosedur lain(bukan lab) selama pertemuan	0%
Number of medications	Numerik	Jumlah nama generik yang berbeda yang diberikan selama pertemuan	0%
Number of outpatiens visit	Numerik	Jumlah kunjungan pasien rawat jalan pada tahun sebelum pertemuan	0%
Number of emergency visits	Numerik	Jumlah kunjungan darurat pasien pada tahun sebelum pertemuan	0%
Number of impatient visits	Numerik	Jumlah kunjungan rawat inap pasien pada tahun sebelum pertemuan	0%
Diagnosis 1	Nominal	Diagnosis primer	0%
Diagnosis 2	Nominal	Diagnosis sekunder	0%
Diagnosis 3	Nominal	Diagnosis sekunder tambahan	1%
Number of diagnosis	Nominal	Jumlah diagnosis yang dimasukkan ke sistem	0%
Glucose serum test result	Nominal	Values: "> 200", "> 300", "normal," dan, "none" jika tidak diukur	0%
A1c test result	Nominal	Values: , "> 7" hasilnya >7% dan <8%, ">8" nilainya >8%, dan "none" jika tidak diukur	0%
Change of Medications	Nominal	Jika ada perubahan pada obat diabetes. Nilai: "berubah" dan "tidak ada perubahan	0%
Diabetes Medications	Nominal	Values: "yes" or "no" terhadap obat generik yang diberikan	0%
24 features medications	Nominal	values: "up", "down", "steady"	0%
Readmitted	Nominal	">30 hari", "<30 hari", "no"	0%

Pada data kami terdapat 50 features, yang dapat dilihat dibawah ini. Ada beberapa data yang missing value (?), ada pula yang bahkan berisi objek semisal V27 dsb. Data kami terdiri dari 101766 instance/baris.

```
data.shape
```

```
(101766, 50)
```

Gambar 7. Jumlah data Instance dan Feature Data awal

```
data.head().T
```

	0	1	2	3	4
encounter_id	2278392	149190	64410	500364	16680
patient_nbr	8222157	55629189	86047875	82442376	42519267
race	Caucasian	Caucasian	AfricanAmerican	Caucasian	Caucasian
gender	Female	Female	Female	Male	Male
age	[0-10)	[10-20)	[20-30)	[30-40)	[40-50)
weight	?	?	?	?	?
admission_type_id	6	1	1	1	1
discharge_disposition_id	25	1	1	1	1
admission_source_id	1	7	7	7	7
time_in_hospital	1	3	2	2	1
payer_code	?	?	?	?	?
medical_specialty	Pediatrics-Endocrinology	?	?	?	?
num_lab_procedures	41	59	11	44	51
num_procedures	0	0	5	1	0
num_medications	1	18	13	16	8
number_outpatient	0	0	2	0	0
number_emergency	0	0	0	0	0

(a)

number_inpatient	0	0	1	0	0
diag_1	250.83	276	648	8	197
diag_2	?	250.01	250	250.43	157
diag_3	?	255	V27	403	250
number_diagnoses	1	9	6	7	5
max_glu_serum	None	None	None	None	None
A1Cresult	None	None	None	None	None
metformin	No	No	No	No	No
repaglinide	No	No	No	No	No
nateglinide	No	No	No	No	No
chlorpropamide	No	No	No	No	No
glimepiride	No	No	No	No	No
acetohexamide	No	No	No	No	No
glipizide	No	No	Steady	No	Steady
glyburide	No	No	No	No	No
tolbutamide	No	No	No	No	No
pioglitazone	No	No	No	No	No
rosiglitazone	No	No	No	No	No
acarbose	No	No	No	No	No
miglitol	No	No	No	No	No

(b)

troglitazone	No	No	No	No	No
tolazamide	No	No	No	No	No
examide	No	No	No	No	No
citoglipton	No	No	No	No	No
insulin	No	Up	No	Up	Steady
glyburide-metformin	No	No	No	No	No
glipizide-metformin	No	No	No	No	No
glimepiride-pioglitazone	No	No	No	No	No
metformin-rosiglitazone	No	No	No	No	No
metformin-pioglitazone	No	No	No	No	No
change	No	Ch	No	Ch	Ch
diabetesMed	No	Yes	Yes	Yes	Yes
readmitted	NO	>30	NO	NO	NO

(c)

Gambar 8. Data Hospital Readmitted

5. Implementation

a) Data Prepocessing

```
data.shape
```

```
(101766, 50)
```

```
data = data.drop(['encounter_id','patient_nbr','gender','weight','payer_code','medical
mide','glyburide','tolbutamide','pioglitazone','rosiglitazone','acarbose','miglitol','
itoglipton','glyburide-metformin','glipizide-metformin','glimepiride-pioglitazone','me
itazone','nateglinide','glimepiride','admission_type_id'], axis=1)
```

Gambar 9. Perintah untuk menghapus kolom-kolom data

Kami menghapus banyak feature yang tidak berkorelasi dengan readmission maupun feature yang memang tidak berkaitan atau mempengaruhi kembalinya pasien ke rumah sakit seperti *payer code*. Dan terdapat beberapa feature yang memiliki cukup banyak *missing value*, yang kemungkinan data pasien-pasien pada rumah sakit tidak tercatat sehingga menyebabkan banyaknya *missing value.* Salah satu contoh feature yang memiliki banyak missing value adalah weight.

Sehingga dengan melakukan pendekatan pada korelasi feature pada target, keterkaitan secara actual feature dengan target dan juga banyak nya missing value pada sebuah feature, diputuskan untuk mengapus sebanyak 26 feature sehingga menyisakan 24 data feature saja.

```
data.shape
```

```
(101766, 24)
```

```
data = data.drop(data[(data.diag_3 == '?') |(data.diag_2 == '?') | (data.diag_1 == '?')].index)
```

```
data.shape
```

```
(100244, 24)
```

Gambar 10. Perintah untuk menghapus baris data yang memiliki

missing value pada diagnosis

Pada gambar 10 terlihat proses penghapusan baris yang memiliki missing value pada baris data pada diagnosis 1, diagnosis 2, dan diagnosis 3. Diagnosis itu sendiri merupakan data yang

diberikan oleh dokter maka dari itu data diag_1, diag_2 dan diag_3 dianggap data yang penting untuk menentukan pasien akan kembali ke rumah sakit atau tidak. Maka dari itu dengan tidak adanya data pada salah satu feature diag_1, diag_2 dan diag_3, baris data tersebut dianggap cacat dan tidak layak untuk diolah menjadi bahan training model.

```
data['diag_3'] = data['diag_3'].map(lambda x: x.lstrip('E').rstrip('E'))
data['diag_2'] = data['diag_2'].map(lambda x: x.lstrip('E').rstrip('E'))
data['diag_1'] = data['diag_1'].map(lambda x: x.lstrip('E').rstrip('E'))
data['diag_3'] = data['diag_3'].map(lambda x: x.lstrip('V').rstrip('V'))
data['diag_2'] = data['diag_2'].map(lambda x: x.lstrip('V').rstrip('V'))
data['diag_1'] = data['diag_1'].map(lambda x: x.lstrip('V').rstrip('V'))
```

```
data['diag_1'] = data['diag_1'].astype('float64')
data['diag_2'] = data['diag_2'].astype('float64')
data['diag_3'] = data['diag_3'].astype('float64')
```

```
data = data.drop(data[(data.diag_3 < 100) &(data.diag_2 < 100) & (data.diag_1 < 100)].index)
```

```
data.shape
```

```
(100192, 24)
```

```
data = data.drop(data[(data.diag_3 < 100) |(data.diag_2 < 100) | (data.diag_1 < 100)].index)
```

```
data.shape
```

```
(88289, 24)
```

Gambar 11. Perintah untuk menghapus baris data yang memiliki *typo*

Pada gambar 11 terlihat proses penghapusan baris yang memiliki data salah ketik/ typo pada baris data diag_1, diag_2 dan diag_3 karena baris data diagnosis dianggap data yang penting. Jadi apabila ada salah satu data dari feature diagnosis yang memiliki missing value, maka data tersebut akan dihapus.

```
data['A1Cresult'] = data['A1Cresult'].replace('>7', 1)
data['A1Cresult'] = data['A1Cresult'].replace('>8', 1)
data['A1Cresult'] = data['A1Cresult'].replace('Norm', 0)
data['A1Cresult'] = data['A1Cresult'].replace('None', -99)
```

```
data.A1Cresult.value_counts()
```

```
-99    73356
 1     10547
 0      4386
Name: A1Cresult, dtype: int64
```

Gambar 12. Perintah untuk merubah data pada feature AC1result

```
data['max_glu_serum'] = data['max_glu_serum'].replace('>200', 1)
data['max_glu_serum'] = data['max_glu_serum'].replace('>300', 1)
data['max_glu_serum'] = data['max_glu_serum'].replace('Norm', 0)
data['max_glu_serum'] = data['max_glu_serum'].replace('None', -99)
```

```
data.max_glu_serum.value_counts()
```

```
-99    83574
 1      2416
 0      2299
Name: max_glu_serum, dtype: int64
```

Gambar 13. Perintah untuk merubah data pada feature max_glu serum

```
for col in data:
    data[col] = data[col].replace('No', 0)
    data[col] = data[col].replace('Steady', 1)
    data[col] = data[col].replace('Up', 1)
    data[col] = data[col].replace('Down', 1)
```

```
data['change'] = data['change'].replace('Ch', 1)
data['change'] = data['change'].replace('No', 0)
```

```
data['diabetesMed'] = data['diabetesMed'].replace('Yes', 1)
data['diabetesMed'] = data['diabetesMed'].replace('No', 0)
```

```
data['readmitted'] = data['readmitted'].replace('>30', 0)
data['readmitted'] = data['readmitted'].replace('<30', 1)
data['readmitted'] = data['readmitted'].replace('NO', 0)
```

Gambar 14. Perintah untuk merubah data pada bebrapa feature terakhir

	1	4	5	7	9
race	3.00	3.0	3.0	3.0	3.0
age	15.00	45.0	55.0	75.0	95.0
discharge_disposition_id	1.00	1.0	1.0	1.0	3.0
admission_source_id	7.00	7.0	2.0	7.0	4.0
time_in_hospital	3.00	1.0	3.0	5.0	12.0
num_lab_procedures	59.00	51.0	31.0	73.0	33.0
num_procedures	0.00	0.0	6.0	0.0	3.0
num_medications	18.00	8.0	16.0	12.0	18.0
number_outpatient	0.00	0.0	0.0	0.0	0.0
number_emergency	0.00	0.0	0.0	0.0	0.0
number_inpatient	0.00	0.0	0.0	0.0	0.0
diag_1	276.00	197.0	414.0	428.0	434.0
diag_2	250.01	157.0	411.0	492.0	198.0
diag_3	255.00	250.0	250.0	250.0	486.0
number_diagnoses	9.00	5.0	9.0	8.0	8.0
max_glu_serum	-99.00	-99.0	-99.0	-99.0	-99.0
A1Cresult	-99.00	-99.0	-99.0	-99.0	-99.0
metformin	0.00	0.0	0.0	0.0	0.0
repaglinide	0.00	0.0	0.0	0.0	0.0
glipizide	0.00	1.0	0.0	0.0	0.0
insulin	1.00	1.0	1.0	0.0	1.0
change	1.00	1.0	0.0	0.0	1.0
diabetesMed	1.00	1.0	1.0	1.0	1.0
readmitted	1.00	0.0	1.0	1.0	0.0

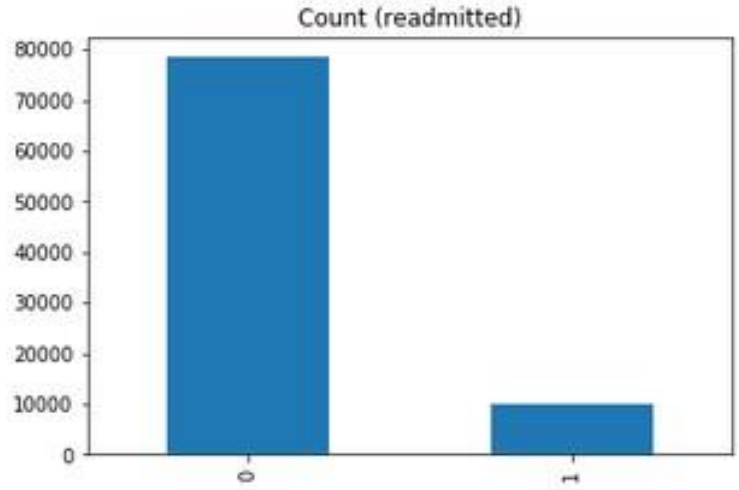
Gambar 15. Tabel data akhir setelah prepocessing

Pada beberapa feature terakhir dapat dilihat apabila kolom tersebut merupakan kolom-kolom data yang berisi tentang data lab pasien dan juga data-data obat apa saja yang diterima oleh pasien diabetes pada setiap rumah sakit.

Pada umumnya tidak semua pasien diabetes akan menerima obat yang sama, obat yang diterima juga mempengaruhi dan dipengaruhi oleh penyakit diabetes yang diderita masingmasing pasien. Dan akan menimbulkan kemungkinan-kemungkinan kembalinya pasien tersebut ke rumah sakit. Maka dr itu data dirubah bentuknya dari string menjadi angka yang berbentuk int agar dapat ikut dalam proses training.

```
print(data.readmitted.value_counts())
data.readmitted.value_counts().plot(kind='bar', title='Count (readmitted)');
```

```
0    78416
1     9873
Name: readmitted, dtype: int64
```



Gambar 16. Data feature ‘Readmitted’ yang tidak seimbang

Setelah banyak kolom dan baris data tersebut dihapus, maka telah menyisakan sekitar 80000 instance. Namun, setelah dilakukan pengecekkan data readmitted yang akan digunakan sebagai target ternyata memiliki ketidakseimbangan data yang ditunjukkan pada gambar 16.

Maka dari itu nantinya akan dilakukan proses balance data guna menyeimbangkan data target tersebut agar setiap pemanggilan yang dilakukan tidak akan berppihak atau condong ke hasil tertentu yang lebih dominan.

b) Data setelah preprocessing

```
1 data.dtypes
```

```
race                        int64
age                         int64
discharge_disposition_id    int64
admission_source_id         int64
time_in_hospital            int64
num_lab_procedures          int64
num_procedures              int64
num_medications             int64
number_outpatient           int64
number_emergency            int64
number_inpatient            int64
diag_1                    float64
diag_2                    float64
diag_3                    float64
number_diagnoses            int64
max_glu_serum               int64
A1Cresult                   int64
metformin                   int64
repaglinide                 int64
glipizide                   int64
insulin                     int64
change                      int64
diabetesMed                 int64
readmitted                  int64
dtype: object
```

Gambar 17. Data sudah menjadi data int dan float

Seperti yang terlihat pada gambar 17, menunjukkan data yang telah di lakukan preprocessing telah siap digunakan. Seluruh data yang akan digunakan telah berubah tipe datanya dari yang sebelumnya objek menjadi int atau float.

```
Y = data['readmitted']
```

```
data2 = data
```

```
data2 = data2.drop(['readmitted'], axis=1)
```

```
data2.head().T
```

Gambar 18. Perintah untuk memisah data untuk target dan data training

Data yang telah siap diolah kemudian di bagi menjadi data yang akan dijadikan model machine learning dan data yang akan dijadikan target(data readmitted). Jadi ‘data2’ merupakan input data training sedangkan data ‘Y’ merupakan data target.

	1	4	5	7	9
race	3.00	3.0	3.0	3.0	3.0
age	15.00	45.0	55.0	75.0	95.0
discharge_disposition_id	1.00	1.0	1.0	1.0	3.0
admission_source_id	7.00	7.0	2.0	7.0	4.0
time_in_hospital	3.00	1.0	3.0	5.0	12.0
num_lab_procedures	59.00	51.0	31.0	73.0	33.0
num_procedures	0.00	0.0	6.0	0.0	3.0
num_medications	18.00	8.0	16.0	12.0	18.0
number_outpatient	0.00	0.0	0.0	0.0	0.0
number_emergency	0.00	0.0	0.0	0.0	0.0
number_inpatient	0.00	0.0	0.0	0.0	0.0
diag_1	276.00	197.0	414.0	428.0	434.0
diag_2	250.01	157.0	411.0	492.0	198.0
diag_3	255.00	250.0	250.0	250.0	486.0
number_diagnoses	9.00	5.0	9.0	8.0	8.0
max_glu_serum	-99.00	-99.0	-99.0	-99.0	-99.0
A1Cresult	-99.00	-99.0	-99.0	-99.0	-99.0
metformin	0.00	0.0	0.0	0.0	0.0
repaglinide	0.00	0.0	0.0	0.0	0.0
glipizide	0.00	1.0	0.0	0.0	0.0
insulin	1.00	1.0	1.0	0.0	1.0
change	1.00	1.0	0.0	0.0	1.0
diabetesMed	1.00	1.0	1.0	1.0	1.0

Gambar 19. Data akhir yang siap di buat model

Di bawah ini merupakan balancing data menggunakan SMOTE yang telah dijelaskan di penjelasan sebelumnya.

```
# Data balancing applied using SMOTE
from imblearn.over_sampling import SMOTE

from collections import Counter
print('Original dataset shape {}'.format(Counter(Y)))
sm = SMOTE(random_state=20)
train_input_new, train_output_new = sm.fit_sample(data2, Y)
print('New dataset shape {}'.format(Counter(train_output_new)))
```

```
Using TensorFlow backend.

Original dataset shape Counter({0: 47378, 1: 40911})
New dataset shape Counter({1: 47378, 0: 47378})
```

Gambar 20. Perintah untuk merubah data agar balance/ seimbang

c) Machine Learning

Setelah data telah seimbang makan tahap selanjutny adalah dilakukan proses training model dengan menggunakan baris program yang ditunjukkan paada gambar 21 dan gambar 22.

```
# define the dictionary of models our script can use
# the key to the dictionary is the name of the model
# (supplied via command line argument) and the value is the model itself
models = {
    "knn": KNeighborsClassifier(n_neighbors=5),
    "naive_bayes": GaussianNB(),
    "logit": LogisticRegression(solver="lbfgs", multi_class="auto"),
    "svm": SVC(kernel="rbf", gamma="auto"),
    "decision_tree": DecisionTreeClassifier(),
    "random_forest": RandomForestClassifier(n_estimators=100),
}
```

Gambar 21. Model yang digunakan

```
train_input_new = pd.DataFrame(train_input_new, columns = list(data2.columns))
```

```
X_train, X_test, Y_train, Y_test = train_test_split(train_input_new,train_output_new, test_size =1/3, random_stat
e=123)
```

```
# train the modelprint("[INFO] using '{}' model".format(model_name))
model = models["knn"]
model.fit(X_train, Y_train) #fit untuk melakukan proses training
# make predictions on our data and show a classification report
print("[INFO] evaluating...")
predictions = model.predict(X_test)
print(classification_report(Y_test, predictions)) #classification_report untuk melakukan hasilnya

print("KNN accuracy : ",accuracy_score(Y_test, predictions , normalize = True))
acc_knn = accuracy_score(Y_test, predictions , normalize = True)
precision_knn = precision_score(Y_test, predictions)
recall_knn = recall_score(Y_test,  predictions)
auc_knn = roc_auc_score(Y_test,  predictions)
```

Gambar 23. Perintah untuk training data dengan menggunakan model

Tabel 5. Hasil training model Percobaan 1

No	Model	Hasil
1	KNN	[INFO] evaluating... precision recall f1-score support 0 0.56 0.52 0.54 15820 1 0.55 0.59 0.57 15766 accuracy 0.55 31586 macro avg 0.55 0.55 0.55 31586 weighted avg 0.55 0.55 0.55 31586 KNN accuracy : 0.5520167162667005

2	Naïve Bayes	[INFO] evaluating... precision recall f1-score support 0 0.55 0.87 0.67 15820 1 0.68 0.28 0.40 15766 accuracy 0.58 31586 macro avg 0.62 0.58 0.53 31586 weighted avg 0.62 0.58 0.54 31586 naive_bayes accuracy : 0.5756347749002723
3	Decision Tree	[INFO] evaluating... precision recall f1-score support 0 0.59 0.58 0.59 15820 1 0.59 0.60 0.59 15766 accuracy 0.59 31586 macro avg 0.59 0.59 0.59 31586 weighted avg 0.59 0.59 0.59 31586 decision_tree accuracy : 0.5906414234154372
4	Random Forest	[INFO] evaluating... precision recall f1-score support 0 0.65 0.71 0.68 15820 1 0.68 0.61 0.64 15766 accuracy 0.66 31586 macro avg 0.66 0.66 0.66 31586 weighted avg 0.66 0.66 0.66 31586 random_forest accuracy : 0.6594693851706452
5	Logit	[INFO] evaluating... precision recall f1-score support 0 0.59 0.60 0.60 15820 1 0.59 0.59 0.59 15766 accuracy 0.59 31586 macro avg 0.59 0.59 0.59 31586 weighted avg 0.59 0.59 0.59 31586 logit accuracy : 0.592794275945039

Cek accuracy menggunakan cross validation

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
X_train, X_dev, Y_train, Y_dev = train_test_split(train_input_new, train_output_new, test_size=0.20, random_state=
0)
```

```
logreg = LogisticRegression(fit_intercept=True, penalty='l1')
print("Cross Validation Score: {:.2%}".format(np.mean(cross_val_score(logreg, X_train, Y_train, cv=10))))
logreg.fit(X_train, Y_train)
print("Dev Set score: {:.2%}".format(logreg.score(X_dev, Y_dev)))
```

```
Cross Validation Score: 61.42%

Dev Set score: 60.90%
```

Gambar 24. Perintah untuk melakukan cross validation

```
knn2 =  KNeighborsClassifier()
print("Cross Validation Score: {:.2%}".format(np.mean(cross_val_score(knn2, X_train, Y_train, cv=10))))
knn2.fit(X_train, Y_train)
print("Dev Set score: {:.2%}".format(knn2.score(X_dev, Y_dev)))
```

```
Cross Validation Score: 55.99%
Dev Set score: 56.28%
```

```
nb = GaussianNB()
print("Cross Validation Score: {:.2%}".format(np.mean(cross_val_score(nb, X_train, Y_train, cv=10))))
nb.fit(X_train, Y_train)
print("Dev Set score: {:.2%}".format(nb.score(X_dev, Y_dev)))
```

```
Cross Validation Score: 57.50%
Dev Set score: 57.54%
```

```
dt = DecisionTreeClassifier()
print("Cross Validation Score: {:.2%}".format(np.mean(cross_val_score(dt, X_train, Y_train, cv=10))))
dt.fit(X_train, Y_train)
print("Dev Set score: {:.2%}".format(dt.score(X_dev, Y_dev)))
```

```
Cross Validation Score: 58.77%
Dev Set score: 58.58%
```

```
rf = RandomForestClassifier(n_estimators=100)
print("Cross Validation Score: {:.2%}".format(np.mean(cross_val_score(dt, X_train, Y_train, cv=10))))
dt.fit(X_train, Y_train)
print("Dev Set score: {:.2%}".format(dt.score(X_dev, Y_dev)))
```

```
Cross Validation Score: 58.67%
Dev Set score: 58.49%
```

Gambar 25. Hasil dari cross validation

Lalu data detraining, dan di test lagi

```
# train the modelprint("[INFO] using '{}' model".format(model_name))
model = models["knn"]
model.fit(X_train, Y_train) #fit untuk melakukan proses training
# make predictions on our data and show a classification report
print("[INFO] evaluating...")
predictions = model.predict(X_test)
print(classification_report(Y_test, predictions)) #classification_report untuk melakukan hasilnya

print("KNN accuracy : ",accuracy_score(Y_test, predictions , normalize = True))
acc_knn = accuracy_score(Y_test, predictions , normalize = True)
precision_knn = precision_score(Y_test, predictions)
recall_knn = recall_score(Y_test,  predictions)
auc_knn = roc_auc_score(Y_test,  predictions)
```

Gambar 26. Perintah untuk melakukan training ulang model setelah pembenaran data ulang setelah dilakukan cross validation

Tabel 6. Hasil training model Percobaan 2

No	Model	Hasil
1	KNN	[INFO] evaluating... precision recall f1-score support 0 0.94 0.61 0.74 26274 1 0.71 0.96 0.82 26004 accuracy 0.79 52278 macro avg 0.83 0.79 0.78 52278 weighted avg 0.83 0.79 0.78 52278 KNN accuracy : 0.7854355560656491
2	Naïve Bayes	[INFO] evaluating... precision recall f1-score support 0 0.65 0.70 0.68 26274 1 0.67 0.62 0.65 26004 accuracy 0.66 52278 macro avg 0.66 0.66 0.66 52278 weighted avg 0.66 0.66 0.66 52278 naive_bayes accuracy : 0.6616358697731359
3	Decision Tree	[INFO] evaluating... precision recall f1-score support 0 0.89 0.87 0.88 26274 1 0.87 0.89 0.88 26004 accuracy 0.88 52278 macro avg 0.88 0.88 0.88 52278 weighted avg 0.88 0.88 0.88 52278 decision_tree accuracy : 0.8760090286545009

4	Random Forest	[INFO] evaluating... precision recall f1-score support 0 0.89 1.00 0.94 26274 1 1.00 0.87 0.93 26004 accuracy 0.94 52278 macro avg 0.94 0.94 0.94 52278 weighted avg 0.94 0.94 0.94 52278 random_forest accuracy : 0.9353456520907456
5	Logit	[INFO] evaluating... precision recall f1-score support 0 0.60 0.64 0.62 26274 1 0.61 0.56 0.58 26004 accuracy 0.60 52278 macro avg 0.60 0.60 0.60 52278 weighted avg 0.60 0.60 0.60 52278 logit accuracy : 0.6027965874746547

Setelah dilakukan percobaan melakukan training pada 5 jenis model yang berbeda, didapatkan satu model yang memiliki nilai accuracy yang tinggi dan juga memiliki precision, recall, f1score yang mencapai angka sebesar 90%. Maka dari itu dipuruskan untuk mengambil random sorest sebagai model terbaik yang selanjutnya akan dicoba untuk melakukan tunning parameter pada model random forest itu sendiri.

Berikut ini [ada gambar 27 dan gambar 28, dilakukan percobaan dengan merubah parameter random forest yang kemudian hasil data nya akan ditampilkan pada experiment result.

```
models = {"random_forest": RandomForestClassifier(n_estimators=10, criterion="gini")} #n_estimators=10 default=
model = models["random_forest"]
model.fit(X_train, Y_train)
print("[INFO] evaluating...")
predictions = model.predict(X_test)
#print(classification_report(Y_test, predictions))
print("random_forest accuracy : ",accuracy_score(Y_test, predictions , normalize = True))
```

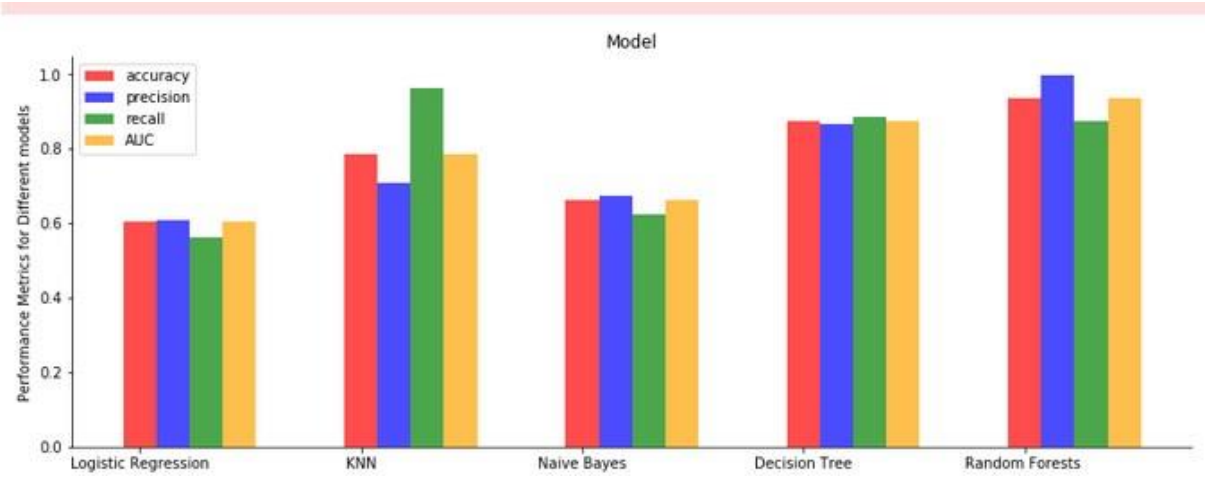
Gambar 27. n estimator=10, 50, 75, 90, 100, dengan criterion=gini

```
models = {"random_forest": RandomForestClassifier(n_estimators=10, criterion="entropy")} #n_estimators=10 defau
model = models["random_forest"]
model.fit(X_train, Y_train)
print("[INFO] evaluating...")
predictions = model.predict(X_test)
#print(classification_report(Y_test, predictions))
print("random_forest accuracy : ",accuracy_score(Y_test, predictions , normalize = True))
```

Gambar 28. n estimator =10, 50, 75, 90, 100, dengan criterion=entropy

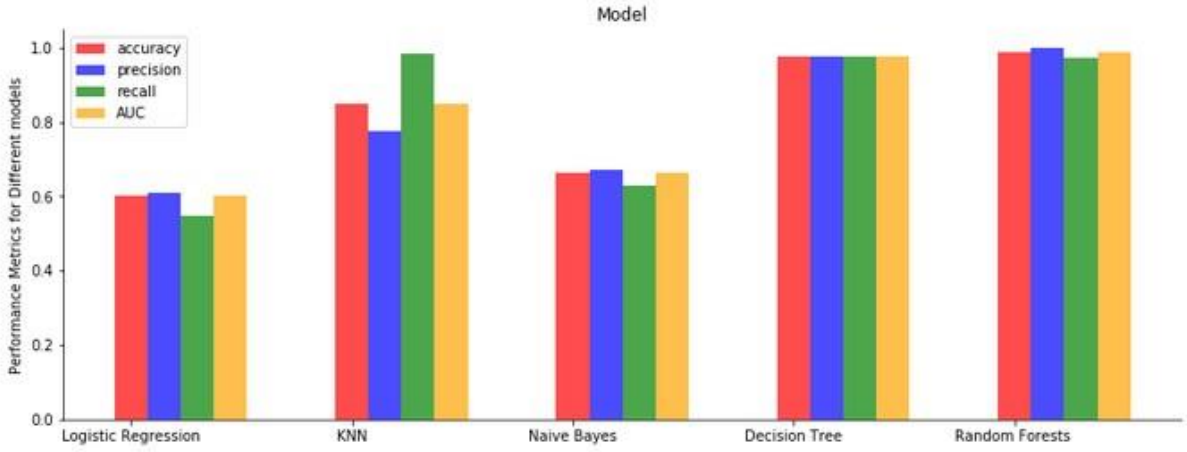
6. Experiment Result

Setelah dilakukan 2 kali percobaan training maka dihasilkan grafik yang menunjukkan nilai dari akurasi, presisi, recal dan f1score pada masing-masing model



Gambar 29. Grafik Hasil training model Percobaan 1

Hasil setelah dibetulkan ulang mengacu pada nilai crossvalidation



Gambar 30. Grafik Hasil training model Percobaan 2

Setelah dilakukan percobaan training kedua, diamana training ini dilakukan setelah melakukan pengecekkan accuracy yang dilakukan dengan menggunakan cross validation. Dan selanjutnya dilakukan sedikit perbaikan data training dan akhirnya dapat menghasilkan data accuracy masing-masing model yang berbeda dari percobaan 1

Tabel 6. Hasil tunning parameter pada model random forest

n esimator	10	50	75	80	90	100	Creation
Akurasi	0.977907	0.98682	0.986878	0.986897	0.986993	0.987127	Gini
	0.978557	0.986687	0.987107	0.986935	0.986897	0.986878	Entropy

Pada percobaan merubah parameter random forest didapatkan hasil seperti tabel 6 yang menunjukkan tingkat akurasi yang memang tidak terlaku jauh berbeda dengan tingkat akurasi menggunakan model random forest dengan parameter default.

7. Discussion

Dapat dilihat bahwa random forest merupakan algoritma yang terbaik untuk kasus hospital readmission baik sebelum dan sesudah cross validation. Hal ini karena algoritma ini bekerja dengan tepat untuk data yang besar. Random forest sangat fleksibel dan memiliki akurasi sangat tinggi. Random forest tidak memerlukan persiapan input data. Menurut Breiman pada tahun 2001, Random Forest menghasilkan error yang lebih rendah, sangat baik dalam kasus klasifikasi dengan dapat mengatasi data training yang cukup besar secara efisien, meskipun ada missing data.

Decision Tree atau pohon pengambil keputusan adalah sebuah diagram alir yang berbentuk seperti pohon yang memiliki sebuah *root node* yang digunakan untuk mengumpulkan data, Sebuah *inner node* yang berada pada *root node* yang berisi tentang pertanyaan tentang data dan sebuah *leaf node* yang digunakan untuk memecahkan masalah serta membuat keputusan. *Decision tree* mengklasifikasikan suatu sampel data yang belum diketahui kelasnya kedalam kelas – kelas yang ada. Penggunaan *decision tree* agar dapat menghindari *overfitting* pada sebuah set data saat mencapai akurasi yang maksimum. *Random forest* adalah kombinasi dari masing – masing *tree* yang baik kemudian dikombinasikan ke dalam satu model. *Random Forest* bergantung pada sebuah nilai vector random dengan distribusi yang sama pada semua pohon yang masing masing *decision tree* memiliki kedalaman yang maksimal.

Untuk nilai parameter n estimator yang terbaik yaitu 100 pada creation gini. Sedangkan untuk nilai parameter n estimator yang terbaik yaitu 75 pada creation entropy. Dua metrik untuk memilih cara memisahkan pohon. Pengukuran Gini adalah probabilitas sampel acak diklasifikasikan secara tidak benar jika kami memilih label secara acak sesuai dengan distribusi di cabang. Entropi adalah ukuran informasi (atau lebih tepatnya kekurangannya). Dengan menghitung perolehan informasi dengan membuat pemisahan. Yang merupakan perbedaan entripies. Ini mengukur bagaimana cara untuk mengurangi ketidakpastian tentang target.

8.

Conclusion

MODEL ALGORITMA yang terbaik dalam memprediksi pasien yang akan mengakami hospital readmission adalah RANDOM FOREST dengan nilai prediksi, akurasi, AUC dan recall berkisar di 0.9 . Random Forest dapat menjadi model terbaik karena random forest itu sendiri memang lebih cocok digunakan untuk data yang memiliki jumlah instance yang banyak seperti data hospital readmission. Namun perlu digaris bawahi kualitas prediksi tidak hanya bergantung pada volume data yang tersedia, tetapi juga variasi data.

Karena dibatasi oleh dataset yang ada, yang merupakan fitur yang cukup komprehensif tetapi tidak lengkap dari semua faktor yang dapat mempengaruhi hospital readmission. Mungkin ada banyak faktor lain tergantung pada situasinya yang dapat memengaruhi hospital readmission. Selain itu, ketersediaan data yang mencakup periode waktu yang lebih lama juga dapat memberikan kontribusi signifikan terhadap kinerja model. Sementara performansi yang tinggi disebabkan oleh data sintetis yang dibuat menggunakan SMOTE, kita dapat mengatakan bahwa performansi akan berada di sekitar itu, dalam perkiraan error konservatif, untuk data yang sebenarnya.

Reference

1. Identifying Diabetic Patients with High Risk of Readmission. arXiv:1602.04257v1
2. Predicting the treatment effect in diabetes patients using classification models3. http://machinelearning.mipa.ugm.ac.id/2018/07/28/random-forest/
3. https://towardsdatascience.com/predicting-hospital-readmission-for-patients-withdiabetes-using-scikit-learn-a2e359b15f0
4. https://medium.com/berkeleyischool/how-to-use-machine-learning-to-predicthospital-readmissions-part-2-616a0c920ab1?