MAKALAH SAINS DATA
PERBANDINGAN METODE KLASIFIKASI UNTUK MENDETEKSI PENYAKIT DIABETES

Disusun sebagai pemenuhan tugas akhir mata kuliah Sains Data

Dosen pengampu
Devvi Sarwinda, M.Kom.

Oleh
Kelompok 3 Kelas Sains Data (B)

1. Abdul Wahhab (2206028485)
2. Hasthabrata Christopher Liatna (2206824741)
3. Kezia Astariza (2206048505)
4. Raissa Anggia Maharani (2206048581)
5. Siti Nur Salamah (2206048833)

DEPARTEMEN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS INDONESIA

2024

ABSTRAK

Masalah penyakit diabetes merupakan masalah yang sudah umum dikalangan masyarakat. Seorang yang terindikasi diabetes memiliki fitur yang menyebabkan terkena diabetes. Penelitian ini dilatarbelakangi oleh diberikannya suatu dataset berisi fitur-fitur yang mungkin menjadi penyebab diabetes, kemudian akan dicari model terbaik dalam mendeteksi terkenanya diabetes seseorang. Tujuan penelitian ini adalah mengurutkan dan membandingkan beberapa model yang terbaik dalam mendeteksi terkenanya penyakit diabetes berdasarkan akurasi dan ROC/AUC.

Model yang kami gunakan untuk prediksi ini adalah model klasifikasi yang di antaranya *Logistic Regression* (LR), *Decision Tree* (DT), *Support Vector Machine* (SVM), *K-Nearest Neighbor* (KNN), dan *Random Forest* (RF). Banyak responden dalam penelitian ini ada 768 orang dengan dataset diambil dari kaggle dengan 9 fitur : kehamilan, kadar glukosa, tekanan darah, ketebalan kulit, insulin, BMI, Diabetes pedegree function, usia, dan diabetes orang tersebut. Dari fitur-fitur tersebut, kami memilih tiga fitur yang menjadi perhatian tambahan kami, yaitu kadar glukosa, insulin, dan BMI.

Hasil penelitian menunjukkan bahwa *Logistic Regression* (LR) memiliki akurasi sekitar 86,84% dan ROC AUC sekitar 87%, *Decision Tree* (DT) memiliki akurasi sekitar 90,13% dan ROC AUC sekitar 86%, *Support Vector Machine* (SVM) memiliki akurasi sekitar 88,82% dan ROC AUC sekitar 88%, *K-Nearest Neighbors* (KNN) memiliki akurasi sekitar 87,5% dan ROC AUC sekitar 86%, serta *Random Forest* (RF) memiliki akurasi sekitar 90,79% dan ROC AUC sekitar 89%. Dapat disimpulkan urutan model yang terbaik secara urutan dalam memprediksi terkenanya diabetes adalah *Random Forest* (RF), *Decision Tree* (DT), *Support Vector Machine* (SVM), *K-Nearest Neighbors* (KNN), dan *Logistic Regression* (LR).

Kata kunci : Diabetes, *Support Vector Machine* (SVM), *Random Forest* (RF), *Logistic Regression* (LR), *K-Nearest Neighbors* (KNN), dan *Decision Tree* (DT), Akurasi, dan ROC AUC.

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Diabetes adalah salah satu masalah kesehatan paling umum di dunia. Diabetes juga dikenal sebagai "*the silent killer*" karena menurut WHO (2016), jumlah penderita diabetes meningkat dari 108 juta pada tahun 1980 menjadi sekitar 422 juta pada tahun 2014. Diabetes terjadi ketika tubuh pengidapnya tidak lagi mampu mengambil gula (glukosa) ke dalam sel dan menggunakannya sebagai energi. Seorang yang terindikasi diabetes biasanya memiliki beberapa fitur yang menyebabkan orang tersebut terkena diabetes.

Jika diberikan suatu dataset masalah diabetes berisi fitur-fitur yang mungkin menjadi penyebab diabetes, hal ini menjadi ketertarikan bagi kami untuk mencari model terbaik dalam mendeteksi terkenanya diabetes seseorang. Kami akan mengurutkan dan membandingkan beberapa model yang terbaik dalam mendeteksi terkenanya penyakit diabetes berdasarkan akurasi dan ROC/AUC.

1.2 Rumusan Masalah

1. Apa model yang digunakan untuk memprediksi terkenanya diabetes seseorang dan bagaimana cara kerjanya?
2. Apa model yang terbaik untuk memprediksi terkenanya diabetes seseorang?

1.3 Tujuan Penelitian

1. Mengetahui cara kerja model dalam memprediksi terkenanya diabetes.
2. Mengetahui model yang terbaik dalam memprediksi terkenanya diabetes seseorang.

BAB II

DATA DAN METODE

2.1. Pengenalan Data dan Eksplorasi Data

Data yang digunakan pada makalah ini ialah dataset diabetes yang tertera pada Kaggle.

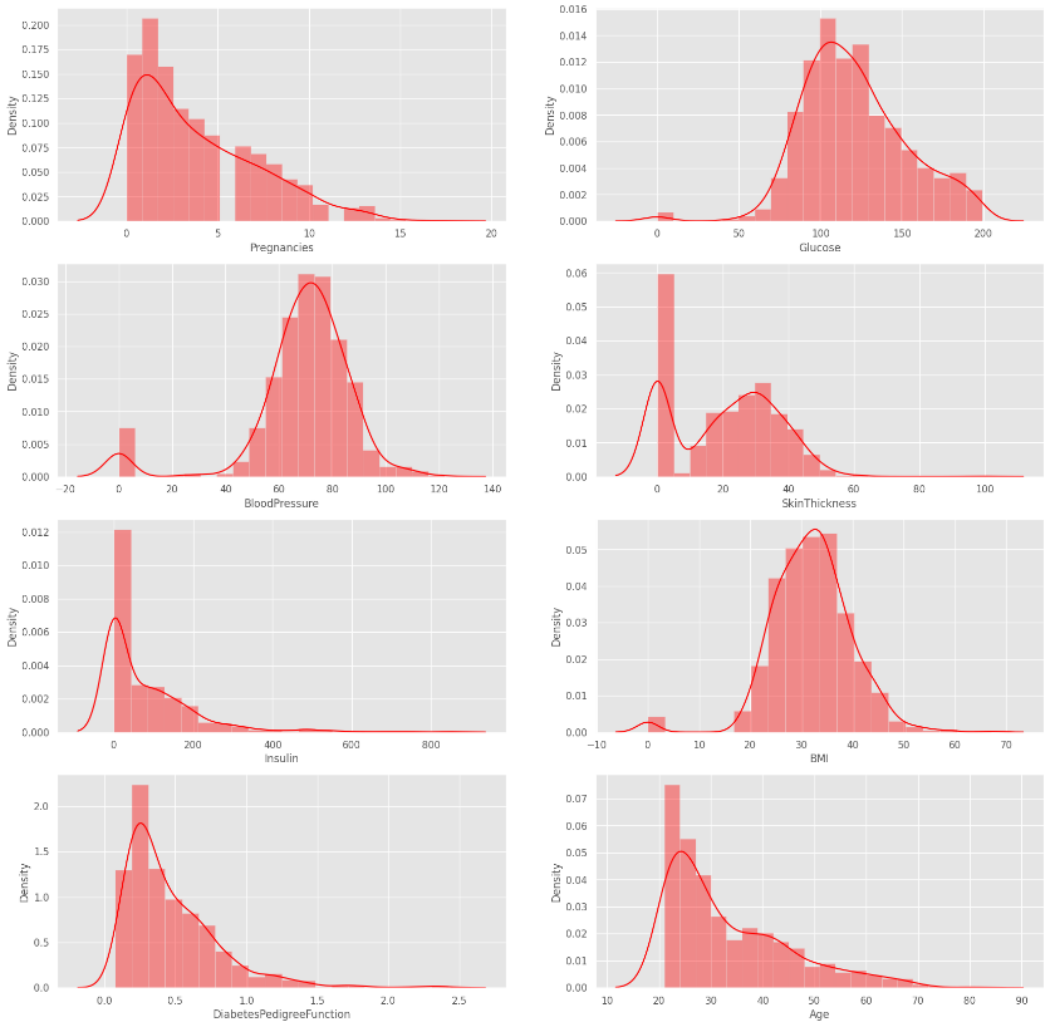
```
[ ]  1 df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Tabel Dataset Diabetes

Data yang diambil adalah data wanita Pima Indian berumur minimal 21 tahun, yang direkam sampai 9 Mei 1990. Dataset ini memiliki fitur-fitur :

Fitur	Deskripsi
Pregnancies	Berapa kali seseorang pernah hamil
Glucose	Konsentrasi Glukosa dengan menggunakan 2 Hour test OGTT (mg/dL)
BloodPressure	Tekanan darah diastolik $(mmHg)$
SkinThickness	Ketebalan lipatan kulit trisep (mm)
Insulin	Insulin Serum 2 Jam $(\mu U/ml)$
BMI	Indeks Massa Tubuh
DiabetesPedigreeFuction	Fungsi yang menilai kemungkinan diabetes berdasarkan riwayat keluarga
Age	Umur
Outcome	Jika 0 maka tidak terkena diabetes dan berlaku sebaliknya untuk 1



Kurva Persebaran Data tiap fitur kecuali Outcome

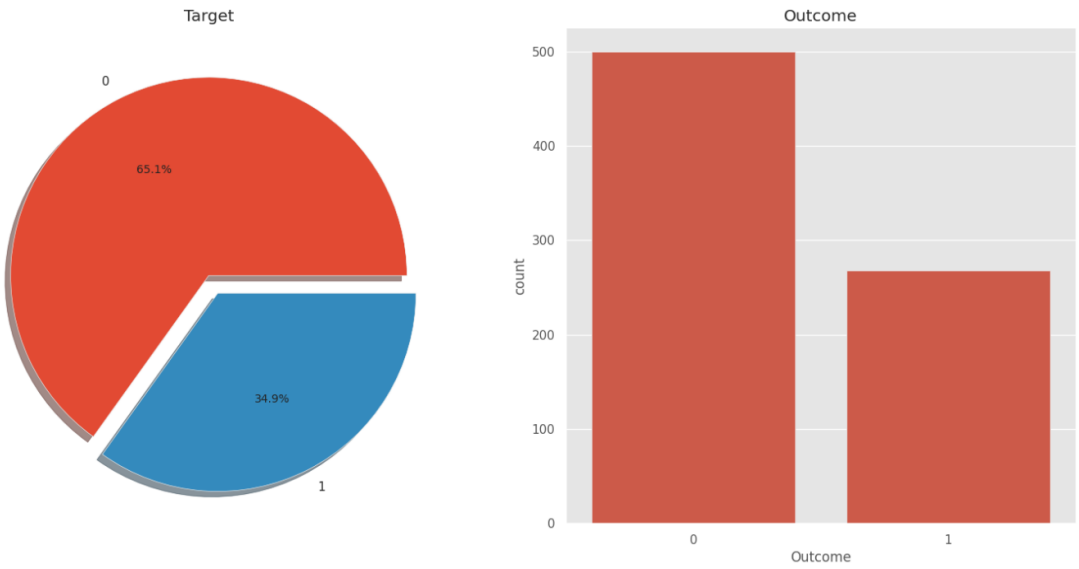
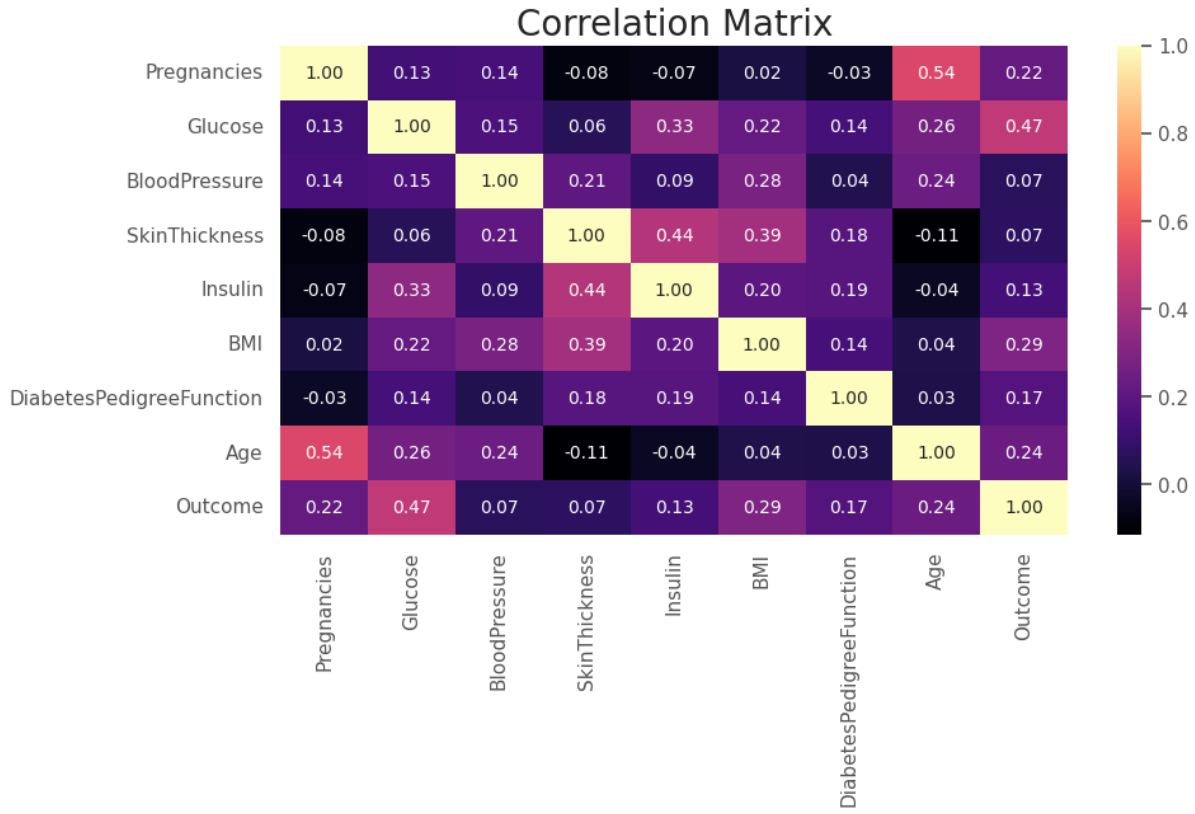


Diagram Persentase Variabel Target



Matrix Korelasi Antar Fitur

Setelah mengekspolarasi dan memvisualisasikan grafik-grafik diatas, dapat disimpulkan bahwa :

- Persebaran data *Glukosa*, BMI, dan *BloodPressure* diasumsikan menghasilkan distribusi normal, dan fitur lainnya menghasilkan distribusi sembarang.
- Lebih banyak pasien yang tidak terkena diabetes
- Fitur yang memiliki korelasi terkuat dengan variabel target adalah Glucose.

2.2. Preprocessing Data

```
6 df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Tabel Deskripsi Dataset Diabetes

Perhatikan bahwa terdapat data-data yang abnormal atau tidak normal, dimana terdapat nilai 0 untuk *'Glucose'*, *'BloodPressure'*, *'SkinThickness'*, *'Insulin'*, dan 'BMI'. Maka dari itu nilai yang tidak normal di asumsikan sebagai nilai kosong atau data *null*.

Metode Imputasi data *null* dilakukan dengan metode median karena keberadaan *outlier* yang ekstrem dapat sangat mempengaruhi nilai mean. Jika imputasi dilakukan menggunakan mean, *outlier* ini akan menarik mean ke arah nilai ekstrem, yang dapat menghasilkan imputasi yang tidak representatif dan

dapat memperkenalkan bias ke dalam data. Dengan menggunakan median, kita dapat menghindari pengaruh negatif dari *outlier*. Median, sebagai nilai tengah dari data yang telah diurutkan, tidak terpengaruh oleh nilai-nilai ekstrem dan memberikan gambaran yang lebih akurat tentang pusat distribusi data. Oleh karena itu, imputasi dengan median menjaga keaslian distribusi data dan mencegah bias yang mungkin muncul jika mean digunakan. Dalam konteks data yang mengandung nilai yang hilang (*null*) dan *outlier*, median memberikan solusi yang lebih stabil dan andal. Metode ini memastikan bahwa nilai yang diimputasi lebih representatif dari data sebenarnya, meningkatkan kualitas analisis dan performa model *machine learning* yang akan menggunakan data tersebut.

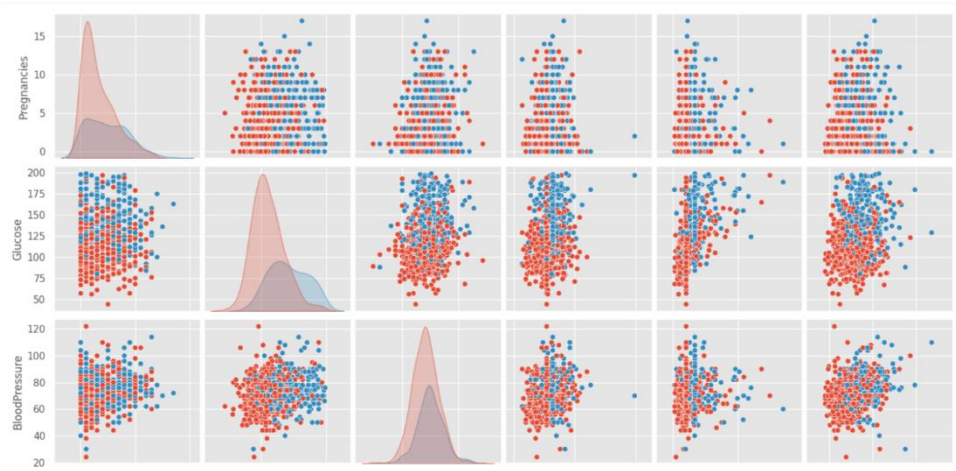
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	169.5	33.6	0.627	50	1
1	1	85.0	66.0	29.0	102.5	26.6	0.351	31	0
2	8	183.0	64.0	32.0	169.5	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1

```
[ ] # Data yang kosong sudah diisi dengan median
    df.isnull().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Tabel Hasil Imputasi Null

Berikut eksplorasi hubungan antara beberapa variabel:



Kurva PairPlot

Pair plot adalah plot yang memungkinkan kita untuk memvisualisasikan hubungan antara pasangan variabel dalam *DataFrame*. Dalam *pair plot*, setiap sumbu akan mewakili satu variabel dalam *DataFrame*, dan setiap sel akan menunjukkan hubungan antara pasangan variabel tersebut. Jika *DataFrame* memiliki kolom yang membedakan kategori atau kelas (seperti kolom "*Outcome*" yang mungkin menunjukkan apakah seseorang sehat atau memiliki diabetes), *parameter hue* dapat digunakan untuk memberi warna pada titik berdasarkan kategori ini.

Dengan memberikan *hue*="*Outcome*", *pair plot* akan menggunakan kolom "*Outcome*" untuk memberi warna pada titik-titik dalam plot, sehingga memungkinkan kita untuk melihat pola hubungan antara pasangan variabel dengan mempertimbangkan kelas atau kategori yang berbeda. Misalnya, kita dapat melihat bagaimana distribusi pasangan variabel berbeda untuk orang-orang yang sehat dan orang-orang yang memiliki diabetes.

2.2.1 Deteksi Outlier dan Penanganan Outlier

```
# Outlier Detection
# IQR+Q1
# 50%
# 24.65->25%+50%
# 24.65->25%
for feature in df:
    Q1 = df[feature].quantile(0.25)
    Q3 = df[feature].quantile(0.75)
    IQR = Q3-Q1
    lower = Q1-1.5*IQR
    upper = Q3+1.5*IQR
    if df[(df[feature]>upper)].any(axis=None):
        print(feature, "yes")
    else:
        print(feature, "no")
```

```
Pregnancies yes
Glucose no
BloodPressure yes
SkinThickness yes
Insulin yes
BMI yes
DiabetesPedigreeFunction yes
Age yes
Outcome no
```

Pendeteksian *outlier* dan penanganan *outlier* dilakukan dengan Menghitung $Q1$, $Q3$, dan IQR dari kolom *'Insulin'* di *DataFrame df*:

Posisi $Q1 : 0.25 \times (n + 1) = 0.25 \times (768 + 1) = 0.25 \times 769 = 192.250.25$

Posisi $Q3 : 0.75 \times (n + 1) = 0.75 \times (768 + 1) = 0.75 \times 769 = 576.750.75$

Posisi ini menunjukkan bahwa $Q1$ adalah antara data ke-192 dan ke-193, dan $Q3$ adalah antara data ke-576 dan ke-577, IQR dihitung dengan mengurangkan Q1 dari Q3. lalu menetapkan batas bawah dan atas untuk *outlier* dengan metode IQR, dan mengganti *outlier* dengan batas atas. Selanjutnya, kita menggunakan *Local Outlier Factor* (LOF) dari *scikit-learn* dengan *n_neighbors*=10 untuk mengidentifikasi *outlier*, menyimpan skor negatif dalam *df_scores*, dan menetapkan *threshold* sebagai skor negatif ketujuh terendah. Dengan *threshold* ini, kita membuat *mask* untuk memilih baris dalam df yang skornya lebih besar dari *threshold* dan menyaring *DataFrame* dan hanya menyisakan baris *non-outlier* menurut LOF.

```
# Outlier sudah diatasi
df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	760.000000	760.000000	760.000000	760.000000	760.000000	760.000000	760.000000	760.000000	760.000000
mean	3.867105	121.543421	72.382895	28.889474	133.565789	32.321447	0.469218	33.205263	0.346053
std	3.367984	30.326388	11.724864	8.377043	58.023446	6.619907	0.325072	11.623200	0.476023
min	0.000000	44.000000	30.000000	7.000000	15.000000	18.200000	0.078000	21.000000	0.000000
25%	1.000000	100.000000	64.000000	25.000000	102.500000	27.500000	0.242500	24.000000	0.000000
50%	3.000000	117.000000	72.000000	28.000000	102.500000	32.000000	0.370000	29.000000	0.000000
75%	6.000000	140.000000	80.000000	32.000000	169.500000	36.500000	0.626250	41.000000	1.000000
max	17.000000	199.000000	114.000000	60.000000	270.000000	57.300000	2.329000	72.000000	1.000000

Tabel Outlier sudah diatasi

2.2.2 Feature Engineering

Seseorang dengan BMI tinggi cenderung memiliki lemak tubuh berlebih, yang bisa menyebabkan resistensi insulin. Resistensi insulin berarti tubuh membutuhkan lebih banyak insulin untuk menurunkan kadar glukosa darah. Kadar glukosa yang tinggi dalam darah bisa menunjukkan bahwa tubuh tidak mampu mengendalikan gula darah secara efektif yang merupakan ciri khas dari diabetes. BMI dibagi menjadi *Normal*, *Obesity 1*, *Obesity 2*, *Obesity 3*, *Overweight*, *Underweight*. Insulin dengan kadar 16 hingga 166 makan normal diluar batas tersebut maka abnormal. Glukosa dibagi menjadi *Low*, *Normal*, *Prediabetes*, *Diabetes.*

```
[ ]  1 # Kategorik untuk BMI
     2 df['NewBMI'] = NewBMI
     3 df.loc[df["BMI"]<18.5, "NewBMI"] = NewBMI[0] # Underweight
     4 df.loc[(df["BMI"]>18.5) & (df["BMI"]<=24.9), "NewBMI"] = NewBMI[1] # Normal
     5 df.loc[(df["BMI"]>24.9) & (df["BMI"]<=29.9), "NewBMI"] = NewBMI[2] # Overweight
     6 df.loc[(df["BMI"]>29.9) & (df["BMI"]<=34.9), "NewBMI"] = NewBMI[3] # Obesity 1
     7 df.loc[(df["BMI"]>34.9) & (df["BMI"]<=39.9), "NewBMI"] = NewBMI[4] # Obesity 2
     8 df.loc[df["BMI"]>39.9, "NewBMI"] = NewBMI[5] # Obesity 3
```

```
[ ]  1 # Kategorik untuk Insulin
     2 # if insulin>=16 & insuline<=166->normal
     3 def set_insuline(row):
     4     if row["Insulin"]>=16 and row["Insulin"]<=166:
     5         return "Normal"
     6     else:
     7         return "Abnormal"
```

```
[ ]  1 # Kategorik untuk Glucose di beberapa Interval
     2 NewGlucose = pd.Series(["Low", "Normal", "Prediabetes", "Diabetes"], dtype = "category")
     3 df["NewGlucose"] = NewGlucose
     4 df.loc[df["Glucose"] <= 70, "NewGlucose"] = NewGlucose[0]
     5 df.loc[(df["Glucose"] > 70) & (df["Glucose"] <= 99), "NewGlucose"] = NewGlucose[1]
     6 df.loc[(df["Glucose"] > 99) & (df["Glucose"] <= 126), "NewGlucose"] = NewGlucose[2]
     7 df.loc[df["Glucose"] > 126 ,"NewGlucose"] = NewGlucose[3]
```

Encoding fitur kategorikal adalah proses mengubah fitur kategorikal (yang terdiri dari nilai non-numerik seperti teks atau label kategori) menjadi format numerik. Dilakukan *One-Hot Encoding* yaitu mengonversi kategori menjadi vektor biner. Setiap kategori unik mendapatkan kolom sendiri dengan nilai 1 jika kategori tersebut ada, dan 0 jika tidak ada.

```
1 from sklearn.preprocessing import RobustScaler
2 transformer = RobustScaler().fit(X)
3 X=transformer.transform(X)
4 X=pd.DataFrame(X, columns = cols, index = index)
```

RobustScaler adalah teknik penskalaan fitur yang penting dalam pra-pemrosesan data, terutama untuk menangani data dengan outlier atau distribusi yang tidak normal. Teknik ini menggunakan median dan *Interquartile Range* (IQR) untuk menentukan skala data, sehingga lebih tahan terhadap pengaruh outlier. Dengan mengurangi median dari setiap fitur dan membaginya dengan IQR, *RobustScaler* memastikan bahwa semua fitur memiliki skala yang seragam. Dampaknya termasuk penanganan outlier yang lebih baik, pengurangan bias, dan normalisasi data, sehingga meningkatkan konsistensi dan keandalan hasil analisis data serta pelatihan model *machine learning*. Dengan demikian, *RobustScaler* menjadi alat yang penting untuk mempersiapkan data untuk berbagai analisis dan pemodelan.

BAB III

IMPLEMENTASI DAN ANALISIS DATA

3.1 Split Data Training dan Data Validasi

```
[ ] # Split Data
    X_train, X_test, y_train , y_test = train_test_split(X,y, test_size=0.2, random_state=0)
    X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((608, 20), (152, 20), (608,), (152,))
```

Dilakukan *splitting* data hasil *preprocessing* menjadi data *training* dan data *testing* dengan perbandingan 80:20 (80% untuk *training* dan 20% untuk *testing*). Diperoleh data *training* sebanyak 608 sampel dan data *testing* sebanyak 152 sampel.

3.2 Membuat Model Regresi Logistik

Regresi logistik adalah metode statistik yang digunakan untuk memodelkan hubungan antara satu atau lebih variabel independen dengan variabel dependen biner (dua kategori).

```
[ ] # Logistic Regreesion
    log_reg = LogisticRegression()
    log_reg.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
[ ] y_pred = log_reg.predict(X_test)
```

```
[ ] accuracy_score(y_train, log_reg.predict(X_train))
```

```
0.8453947368421053
```

Dibuat model menggunakan model regresi logistik untuk membuat prediksi. Pertama, model 'LogisticRegression()' diinisialisasi dan disimpan dalam variabel 'log_reg'. Selanjutnya, model tersebut dilatih menggunakan data *training* 'X_train' (fitur) dan 'y_train' (label target) dengan metode *fit*. Setelah pelatihan, model digunakan untuk memprediksi label target untuk data *testing* 'X_test' menggunakan metode *predict*, hasilnya disimpan dalam 'y_pred'. Terakhir, akurasi model diukur dengan menghitung skor akurasi antara label target sebenarnya 'y_train' dan prediksi model pada data *training* 'log_reg.predict(X_train)' menggunakan fungsi 'accuracy_score'.

3.3 Evaluasi Model Regresi Logistik

```
[ ] print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.86      0.89        98
           1       0.77      0.89      0.83        54

    accuracy                           0.87       152
   macro avg       0.85      0.87      0.86       152
weighted avg       0.88      0.87      0.87       152
```

Diperoleh metrik evaluasi pada data *training* sudah cukup baik. Didapat bahwa *Precision* untuk kelas 0 yaitu 93% dari prediksi kelas 0 benar-benar merupakan kelas 0. Precision untuk kelas 1 yaitu 77% dari prediksi kelas 1 benar-benar merupakan kelas 1.

Selanjutnya *Recall* untuk kelas 0 yaitu 86% dari total data kelas 0 telah diprediksi dengan benar sebagai kelas 0. *Recall* untuk kelas 1 yaitu 89% dari total data kelas 1 telah diprediksi dengan benar sebagai kelas 1.

Kemudian nilai *F1-score* untuk kelas 0 adalah 89% yang merupakan rata-rata harmonik antara precision dan recall untuk kelas 0. Nilai *F1-score* untuk kelas 1 adalah 83% yang merupakan rata-rata harmonik antara *precision* dan *recall* untuk kelas 1.

Didapat bahwa **akurasi keseluruhan dari model adalah 87%** yang menunjukkan seberapa baik model memprediksi dengan benar label kelas secara keseluruhan.

3.4 Membuat Model Decision Tree

Decision tree adalah algoritma pembelajaran mesin yang menggunakan struktur pohon untuk membuat keputusan berdasarkan fitur data. Setiap node internal mewakili tes pada atribut, setiap cabang mewakili hasil tes, dan setiap daun mewakili prediksi atau klasifikasi.

```
[ ]  # Decision Tree
     DT = DecisionTreeClassifier()
     DT.fit(X_train, y_train)
     y_pred = DT.predict(X_test)
     print(accuracy_score(y_train, DT.predict(X_train)))
```

```
[ ]  1 # hyperparameter tuning of dt
     2 grid_param = {
     3     'criterion':['gini','entropy'],
     4     'max_depth' :  [3,5,7,10],
     5     'splitter' : ['best','radom'],
     6     'min_samples_leaf':[1,2,3,5,7],
     7     'min_samples_split':[1,2,3,5,7],
     8     'max_features':['auto','sqrt','log2']
     9 }
    10 grid_search_dt = GridSearchCV(DT, grid_param, cv=50, n_jobs=-1, verbose = 1)
    11 grid_search_dt.fit(X_train, y_train)
```

```
Fitting 50 folds for each of 1200 candidates, totalling 60000 fits
▸ GridSearchCV
▸ estimator: DecisionTreeClassifier
    ▸ DecisionTreeClassifier
```

```
[ ]  1 grid_search_dt.best_params_ # Best parameter yang digunakan untuk DTC
```

```
{'criterion': 'gini',
 'max_depth': 5,
 'max_features': 'sqrt',
 'min_samples_leaf': 3,
 'min_samples_split': 7,
 'splitter': 'best'}
```

Dibuat model Decision Tree dengan beberapa langkah. Pertama, sebuah objek ‘DecisionTreeClassifier’ dibuat dan disimpan dalam variabel DT. Kemudian, model tersebut dilatih menggunakan data *training* (‘X_train’ dan ‘y_train’) dengan memanggil metode *fit*. Setelah model dilatih, prediksi dilakukan terhadap data *testing* (‘X_test’) menggunakan metode *predict*, dan hasil prediksinya disimpan dalam variabel ‘y_pred’. Selanjutnya dilakukan *tuning hyperparameter* untuk mencari hyperparameter terbaik dan gunakan hasil estimator terbaik tersebut. Terakhir, akurasi model diukur dengan membandingkan prediksi model terhadap data pelatihan menggunakan fungsi ‘accuracy_score’.

3.5 Evaluasi Model Decision Tree

```
DT = grid_search_dt.best_estimator_
y_pred = DT.predict(X_test)
print(accuracy_score(y_train, DT.predict(X_train)))
dt_acc = accuracy_score(y_test, DT.predict(X_test))
print(accuracy_score(y_test, DT.predict(X_test)))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
0.9046052631578947
0.9013157894736842
[[91  7]
 [ 8 46]]
              precision    recall  f1-score   support

           0       0.92      0.93      0.92        98
           1       0.87      0.85      0.86        54

    accuracy                           0.90       152
   macro avg       0.89      0.89      0.89       152
weighted avg       0.90      0.90      0.90       152
```

Diperoleh metrik evaluasi pada data *training* sudah cukup baik. Didapat bahwa *Precision* untuk kelas 0 yaitu 92% di prediksi kelas 0 benar-benar merupakan kelas 0. *Precision* untuk kelas 1 yaitu 87% dari prediksi kelas 1 benar-benar merupakan kelas 1.

Selanjutnya *Recall* untuk kelas 0 yaitu 93% dari total data kelas 0 telah diprediksi dengan benar sebagai kelas 0. *Recall* untuk kelas 1 yaitu 85% dari total data kelas 1 telah diprediksi dengan benar sebagai kelas 1.

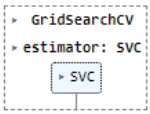
Kemudian nilai *F1-score* untuk kelas 0 adalah 92% yang merupakan rata-rata harmonik antara precision dan recall untuk kelas 0. Nilai *F1-score* untuk kelas 1 adalah 86% yang merupakan rata-rata harmonik antara precision dan recall untuk kelas 1.

Didapat bahwa **akurasi keseluruhan dari model adalah 90%** yang menunjukkan seberapa baik model memprediksi dengan benar label kelas secara keseluruhan.

3.6 Membuat Model Support Vector Machine

Support Vector Machine (SVM) adalah algoritma pembelajaran mesin yang bekerja dengan mencari *hyperplane* optimal yang memisahkan data ke dalam kelas-kelas berbeda dengan margin maksimum. Untuk parameter kernel digunakan *default* dari SVM yaitu RBF.

```
# SVM
svc = SVC(probability=True)
parameter = {
    "gamma":[0.0001, 0.001, 0.01, 0.1],
    'C': [0.01, 0.05,0.5, 0.01, 1, 10, 15, 20]
}
grid_search = GridSearchCV(svc, parameter)
grid_search.fit(X_train, y_train)
```




```
# best_parameter
grid_search.best_params_
```

```
{'C': 0.5, 'gamma': 0.1}
```

```
grid_search.best_score_
```

```
0.8651402249017748
```

```
svc = SVC(C=0.5, gamma = 0.01, probability=True)
svc.fit(X_train, y_train)
```

Dibuat Support Vector Machine (SVM) untuk klasifikasi dengan mencari parameter terbaik menggunakan 'GridSearchCV'. Pertama, sebuah SVM *classifier* dibuat. Kemudian, sebuah set parameter yang ingin dioptimalkan, seperti nilai gamma dan C, ditentukan. Dengan menggunakan 'GridSearchCV', model SVM dilatih dengan berbagai kombinasi parameter untuk mencari yang terbaik berdasarkan skor validasi silang. Setelah kombinasi terbaik ditemukan, SVM dipasang kembali dengan parameter terbaik tersebut. Akhirnya, model SVM yang telah disesuaikan digunakan untuk membuat prediksi pada data *testing* untuk evaluasi performa model.

3.7 Evaluasi Model Support Vector Machine

```
print(accuracy_score(y_train, svc.predict(X_train)))
svc_acc = accuracy_score(y_test, svc.predict(X_test))
print(accuracy_score(y_test, svc.predict(X_test)))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
0.8305921052631579
0.8881578947368421
[[87 11]
 [ 6 48]]
              precision    recall  f1-score   support

           0       0.94      0.89      0.91        98
           1       0.81      0.89      0.85        54

    accuracy                           0.89       152
   macro avg       0.87      0.89      0.88       152
weighted avg       0.89      0.89      0.89       152
```

Diperoleh metrik evaluasi pada data *training* sudah cukup baik. Didapat bahwa *Precision* untuk kelas 0 yaitu 94% dari prediksi kelas 0 benar-benar merupakan kelas 0. *Precision* untuk kelas 1 yaitu 81% dari prediksi kelas 1 benar-benar merupakan kelas 1.

Selanjutnya *Recall* untuk kelas 0 yaitu 89% dari total data kelas 0 telah diprediksi dengan benar sebagai kelas 0. *Recall* untuk kelas 1 yaitu 89% dari total data kelas 1 telah diprediksi dengan benar sebagai kelas 1.

Kemudian nilai *F1-score* untuk kelas 0 adalah 91%, yang merupakan rata-rata harmonik antara precision dan recall untuk kelas 0. Nilai *F1-score* untuk kelas 1 adalah 85%, yang merupakan rata-rata harmonik antara *precision* dan *recall* untuk kelas 1.

Didapat bahwa **akurasi keseluruhan dari model adalah 89%** yang menunjukkan seberapa baik model memprediksi dengan benar label kelas secara keseluruhan.

3.8 Membuat Model K-Nearest Neighbors

K-Nearest Neighbors (KNN) adalah algoritma non-parametrik yang digunakan dalam pembelajaran mesin untuk mengklasifikasikan atau memprediksi nilai berdasarkan kedekatan (*similarity*) dengan data pelatihan di mana kedekatan dihitung menggunakan metrik jarak seperti Euclidean. Untuk parameter K digunakan *default* dari KNN.

```
# KNN
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

Dibuat model KneighborsClassifier dengan beberapa langkah. Pertama, sebuah objek KNeighborsClassifier dibuat dan dilatih menggunakan data *training* (X_train, y_train) dengan metode

fit(). Kemudian, model tersebut digunakan untuk memprediksi label data *testing* (X_test) menggunakan metode predict(). Akurasi prediksi pada data pelatihan dihitung dan ditampilkan dengan accuracy_score(y_train, knn.predict(X_train)), yang bertujuan untuk mengevaluasi seberapa baik model belajar dari data *training*. Selanjutnya, akurasi prediksi pada data *testing* dihitung dan disimpan dalam variabel knn_acc, serta ditampilkan menggunakan accuracy_score(y_test, knn.predict(X_test)), untuk mengevaluasi performa model.

3.9 Evaluasi Model K-Nearest Neighbors

```
print(accuracy_score(y_train, knn.predict(X_train)))
print(accuracy_score(y_test, knn.predict(X_test)))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
0.8766447368421053
0.875
[[89  9]
 [10 44]]
              precision    recall  f1-score   support

           0       0.90      0.91      0.90        98
           1       0.83      0.81      0.82        54

    accuracy                           0.88       152
   macro avg       0.86      0.86      0.86       152
weighted avg       0.87      0.88      0.87       152
```

Diperoleh metrik evaluasi pada data *training* sudah cukup baik. Didapat bahwa *Precision* untuk kelas 0 yaitu 90% dari prediksi kelas 0 benar-benar merupakan kelas 0. *Precision* untuk kelas 1 yaitu 83% dari prediksi kelas 1 benar-benar merupakan kelas 1.

Selanjutnya *Recall* untuk kelas 0 yaitu 91% dari total data kelas 0 telah diprediksi dengan benar sebagai kelas 0. *Recall* untuk kelas 1 yaitu 82% dari total data kelas 1 telah diprediksi dengan benar sebagai kelas 1.

Kemudian nilai *F1-score* untuk kelas 0 adalah 90% yang merupakan rata-rata harmonik antara precision dan recall untuk kelas 0. Nilai *F1-score* untuk kelas 1 adalah 82% yang merupakan rata-rata harmonik antara *precision* dan *recall* untuk kelas 1.

Didapat bahwa **akurasi keseluruhan dari model adalah 88%** yang menunjukkan seberapa baik model memprediksi dengan benar label kelas secara keseluruhan.

3.10 Membuat Model Random Forest

Random Forest adalah algoritma pembelajaran mesin yang digunakan untuk klasifikasi dan regresi, yang terdiri dari banyak pohon keputusan (*decision trees*) yang dibangun pada subset acak dari data pelatihan. Hasil akhir diperoleh dengan menggabungkan prediksi dari semua pohon untuk meningkatkan akurasi dan mengurangi *overfitting*.

```
# RandomForestClassifer
rand_clf = RandomForestClassifier(criterion = 'entropy', max_depth = 15, max_features = 0.75,
                                  min_samples_leaf = 2, min_samples_split = 3, n_estimators = 130)
rand_clf.fit(X_train, y_train)
```

```
RandomForestClassifier(criterion='entropy', max_depth=15, max_features=0.75,
                       min_samples_leaf=2, min_samples_split=3,
                       n_estimators=130)
```

Dibuat Random Forest Classifier untuk klasifikasi. Random Forest Classifier diinisialisasi dengan *hyperparameter* yang telah ditentukan seperti kedalaman maksimum, jumlah fitur maksimum untuk dipertimbangkan saat mencari pembagian terbaik, jumlah minimum sampel yang diperlukan untuk

membagi simpul internal, jumlah minimum sampel yang diperlukan untuk simpul daun, dan jumlah pohon keputusan dalam *ensemble*. Setelah inisialisasi, model dilatih menggunakan data *training* ('X_train', 'y_train'). Model yang telah dilatih kemudian dapat digunakan untuk membuat prediksi pada data *testing*.

3.11 Evaluasi Model Random Forest

```
[ ] y_pred = rand_clf.predict(X_test)
    print(accuracy_score(y_train, rand_clf.predict(X_train)))
    rand_acc = accuracy_score(y_test, rand_clf.predict(X_test))
    print(accuracy_score(y_test, rand_clf.predict(X_test)))
    print(confusion_matrix(y_test, y_pred))
    print(classification_report(y_test, y_pred))
```

```
0.9868421052631579
0.9078947368421053
[[91  7]
 [ 7 47]]
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        98
           1       0.87      0.87      0.87        54

    accuracy                           0.91       152
   macro avg       0.90      0.90      0.90       152
weighted avg       0.91      0.91      0.91       152
```

Diperoleh metrik evaluasi pada data *training* sudah cukup baik. Didapat bahwa *Precision* untuk kelas 0 yaitu 93% dari prediksi kelas 0 benar-benar merupakan kelas 0. *Precision* untuk kelas 1 yaitu 87% dari prediksi kelas 1 benar-benar merupakan kelas 1.

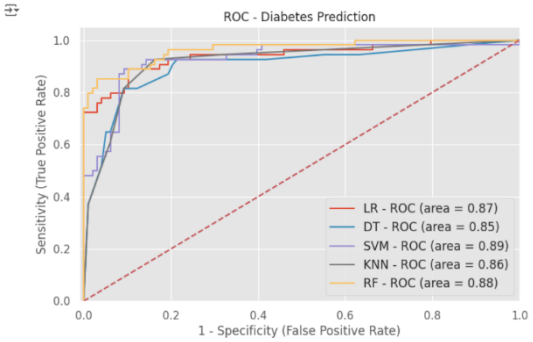
Selanjutnya *Recall* untuk kelas 0 yaitu 93% dari total data kelas 0 telah diprediksi dengan benar sebagai kelas 0. *Recall* untuk kelas 1 yaitu 87% dari total data kelas 1 telah diprediksi dengan benar sebagai kelas 1.

Kemudian nilai *F1-score* untuk kelas 0 adalah 93% yang merupakan rata-rata harmonik antara precision dan recall untuk kelas 0. Nilai *F1-score* untuk kelas 1 adalah 87% yang merupakan rata-rata harmonik antara precision dan recall untuk kelas 1.

Didapat bahwa **akurasi keseluruhan dari model adalah 91%** yang menunjukkan seberapa baik model memprediksi dengan benar label kelas secara keseluruhan.

3.12 Perbandingan Model Berdasarkan Akurasi & ROC

	Model	Score
4	Random Forest Classifier	90.79
3	Decision Tree Classifier	90.13
2	SVM	88.82
1	KNN	87.50
0	Logistic Regression	86.84



Tabel dan Kurva Perbandingan Model Berdasarkan Akurasi & ROC

Hasil diatas merupakan hasil dari akurasi dan kurva ROC-AUC. ROC-AUC (*Receiver Operating Characteristic Area Under the Curve*) merupakan alat evaluasi yang penting untuk membantu mengukur kemampuan model dalam membedakan antara kelas positif dan negatif dengan memperhatikan *trade-off* antara tingkat *True Positive Rate* (TPR) dan tingkat *False Positive Rate* (FPR).

- *True Positive* (TP): Jumlah observasi yang benar-benar positif dan diprediksi dengan benar sebagai positif oleh model. Dalam konteks diabetes TP adalah jumlah kasus di mana model memprediksi pasien memiliki diabetes (positif) dan mereka benar-benar memiliki diabetes berdasarkan data aktual.
- *False Positive* (FP): Jumlah observasi yang sebenarnya negatif tetapi salah diprediksi sebagai positif oleh model. Dalam konteks diabetes FP adalah jumlah kasus di mana model memprediksi pasien memiliki diabetes (positif) tetapi sebenarnya mereka tidak memiliki diabetes.
- *True Negative* (TN): Jumlah observasi yang benar-benar negatif dan diprediksi dengan benar sebagai negatif oleh model. Dalam konteks diabetes TN adalah jumlah kasus di mana model memprediksi pasien tidak memiliki diabetes (negatif) dan mereka memang tidak memiliki diabetes berdasarkan data aktual.
- *False Negative (FN): Jumlah observasi yang sebenarnya positif tetapi salah diprediksi sebagai negatif oleh* model. Dalam konteks diabetes FN adalah jumlah kasus di mana model memprediksi pasien tidak memiliki diabetes (negatif) tetapi sebenarnya mereka memiliki diabetes.

True Positive Rate (TPR), juga dikenal sebagai *Sensitivy* atau *Recall*, dihitung dengan rumus:

$$TPR = \frac{TP}{TP + FN}$$

True Positive Rate mengukur sejauh mana model mampu mengidentifikasi kasus positif (diabetes) dengan benar.

False Positive Rate (FPR), dihitung dengan rumus:

$$FPR = \frac{FP}{FP + TN}$$

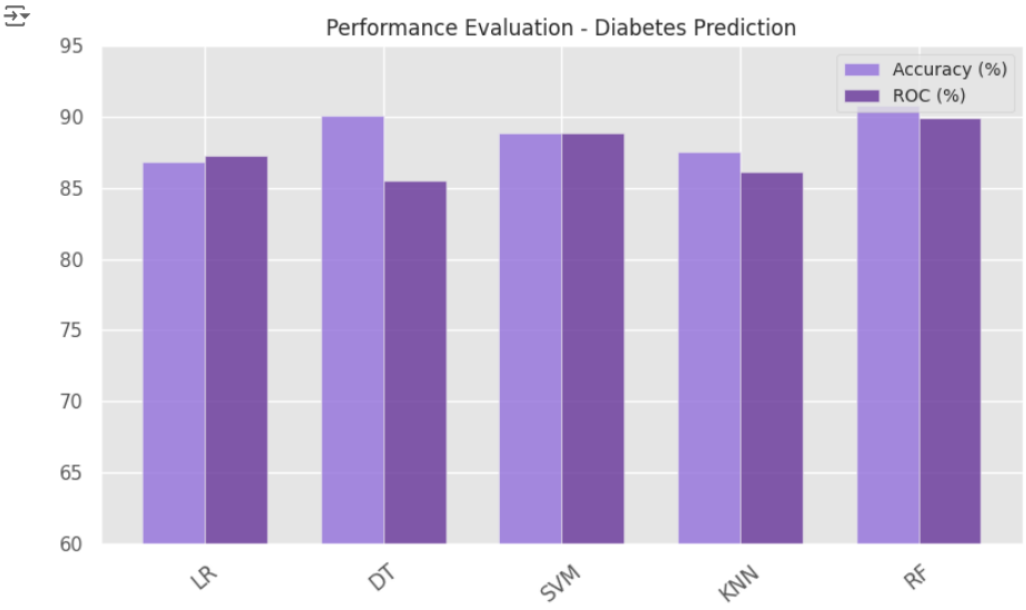
False Positive Rate mengukur sejauh mana model mampu memprediksi kasus negatif sebagai positif. Dalam konteks diabetes FPR adalah proporsi negatif palsu (tidak diabetes) yang salah diprediksi sebagai positif (diabetes) oleh model dibandingkan dengan total jumlah kasus negatif palsu.

Pada ROC, setiap titik mewakili satu ambang batas tertentu. Semakin dekat kurva ROC ke sudut kiri atas, semakin baik kinerja model. Sedangkan *Area Under the Curve* (AUC) adalah suatu nilai yang dihitung dari luas area di bawah kurva ROC. AUC mengukur seberapa baik model membedakan antara kelas positif dan negatif tanpa memperhatikan ambang batas spesifik. Nilai AUC berkisar antara 0 hingga 1, dengan nilai 1 menunjukkan bahwa model sempurna dalam membedakan kelas, sedangkan nilai 0,5 menunjukkan performa yang sama dengan pengacakan.

Area Under the Curve (AUC), dihitung dengan rumus:

$$AUC = \int_0^1 TPR\left(FPR^{-1}(u)\right)du$$

Kemudian, berikut merupakan hasil perbandingan antara akurasi dan kurva ROC dari 5 model klasifikasi.



Berdasarkan grafik yang diberikan, didapatkan bahwa hasil evaluasi beberapa model klasifikasi yaitu sebagai berikut:

Model Klasifikasi	Accuracy	ROC AUC
Logistic Regression (LR)	86,84%	87%
Decision Tree (DT)	90,13%	86%
Support Vector Machine (SVM)	88,82%	88%
K-Nearest Neighbors (KNN)	87,5%	86%
Random Forest (RF)	90,79%	89%

BAB IV

KESIMPULAN DAN SARAN

4.1 Kesimpulan

Regresi logistik adalah metode statistik yang digunakan untuk memodelkan hubungan antara satu atau lebih variabel independen dengan variabel dependen biner (dua kategori). *Decision tree* adalah algoritma pembelajaran mesin yang menggunakan struktur pohon untuk membuat keputusan berdasarkan fitur data. Setiap node internal mewakili tes pada atribut, setiap cabang mewakili hasil tes, dan setiap daun mewakili prediksi atau klasifikasi. *Support Vector Machine* (SVM) adalah algoritma pembelajaran mesin yang bekerja dengan mencari hyperplane optimal yang memisahkan data ke dalam kelas-kelas berbeda dengan margin maksimum. *K-Nearest Neighbors* (KNN) adalah algoritma non-parametrik yang digunakan dalam pembelajaran mesin untuk mengklasifikasikan atau memprediksi nilai berdasarkan kedekatan (similarity) dengan data pelatihan di mana kedekatan dihitung menggunakan metrik jarak seperti Euclidean. *Random Forest* adalah algoritma pembelajaran mesin yang digunakan untuk klasifikasi dan regresi, yang terdiri dari banyak pohon keputusan (decision trees) yang dibangun pada subset acak dari data pelatihan. Hasil akhir diperoleh dengan menggabungkan prediksi dari semua pohon untuk meningkatkan akurasi dan mengurangi *overfitting*.

Didapat bahwa hasil penelitian menunjukkan urutan model yang terbaik secara urutan dalam memprediksi terkenanya diabetes dengan metode klasifikasi adalah *Random Forest* (RF), *Decision Tree* (DT), *Support Vector Machine* (SVM), *K-Nearest Neighbors* (KNN), dan *Logistic Regression* (LR).

4.2 Saran

1. *Random Forest* dan *Decision tree* adalah pilihan terbaik berdasarkan hasil evaluasi, karena keduanya menunjukkan akurasi dan ROC-AUC yang tinggi.
2. Lakukan *tuning hyperparameter* untuk model yang dipilih (misalnya, menggunakan *Grid Search* atau *Random Search*) untuk mendapatkan performa yang lebih optimal.
3. Lakukan validasi silang (*cross-validation*) untuk memastikan bahwa model yang dipilih tidak overfit pada data uji tertentu.
4. Pertimbangkan penggunaan teknik seleksi fitur atau *dimensionality reduction* (misalnya PCA) untuk meningkatkan performa model.
5. Jika memungkinkan, gunakan data tambahan untuk melatih model, yang dapat membantu meningkatkan generalisasi dan performa model.
6. Setelah memilih model yang terbaik, lakukan implementasi dan *monitoring* performa model di lingkungan produksi untuk memastikan bahwa model tetap berperforma baik saat digunakan dengan data nyata.

DAFTAR PUSTAKA

Arora, N., Singh, A., Al-Dabagh, M. Z. N., & Maitra, S. K. (2022). A novel architecture for diabetes patients' prediction using K-means clustering and SVM. *Mathematical Problems in Engineering*, *2022*.

Kadhm, M. S., Ghindawi, I. W., & Mhawi, D. E. (2018). An accurate diabetes prediction system based on K-means clustering and proposed classification approach. *International Journal of Applied Engineering Research*, *13*(6), 4038-4041.

Medscape. (n.d.). Hiperkolesterolemia (penyakit kolesterol tinggi) - Gambaran umum. Diakses pada 5 Juni 2024, dari https://emedicine.medscape.com/article/2089224-overview?form=fpf

National Heart, Lung, and Blood Institute. (n.d.). BMI Calculator: Calculate Your Body Mass Index. Diakses pada 5 Juni 2024, dari https://www.nhlbi.nih.gov/health/educational/lose_wt/BMI/bmicalc.htm

Palimkar, P., Shaw, R. N., & Ghosh, A. (2022). Machine learning technique to prognosis diabetes disease: Random forest classifier approach. In *Advanced Computing and Intelligent Technologies: Proceedings of ICACIT 2021* (pp. 219-244). Springer Singapore.

Posonia, A. M., Vigneshwari, S., & Rani, D. J. (2020, December). Machine learning based diabetes prediction using decision tree J48. In *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)* (pp. 498-502). IEEE.

Rajendra, P., & Latifi, S. (2021). Prediction of diabetes using logistic regression and ensemble techniques. *Computer Methods and Programs in Biomedicine Update*, *1*, 100032.

World Health Organization. (n.d.). Indicator metadata registry: Fasting plasma glucose (mmol/L). Diakses pada 5 Juni 2024, darihttps://www.who.int/data/gho/indicator-metadata-registry/imr-details/2380#:~:text=The%20expected%20values%20for%20normal,and%20monitoring%20glycemia%20are%20recommended