Diabetes Prediction Using Machine Learning

Gagas Abdul Nasheem

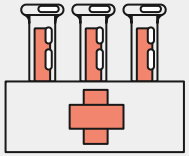
Table of **contents**

Data Understanding

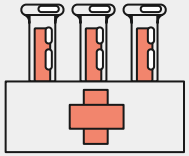
Introduction

Diabetes adalah penyakit yang disebabkan oleh tingginya kandungan gula darah.
Diabetes merupakan penyakit berbahaya, berdasarkan survey yang dilakukan oleh Kementerian Kesehatan diabetes merupakan penyakit penyebab kematian terbesar nomor 3 (6,1%) setelah stroke (21%) dan jantung koroner (14%), jumlah ini terus mengalami peningkatan setiap tahunnya. WHO memperkirakan jumlah pasien diabetes akan meningkat signifikan hingga 16,7% pada tahun 2045. Hal ini terjadi seiring dengan gaya hidup masyarakat yang tidak sehat dan tren konsumsi minuman kekinian seperti kopi. Gula yang terkandung pada minuman tersebut juga dapat meningkatkan kandungan gula darah dalam tubuh.

Objectives



Menentukan faktor penyebab orang terkena diabetes.



Membangun Model Machine Learning untuk klasifikasi seseorang terkena diabetes atau tidak.

Data Understanding

Dataset Details

```
[ ] #Import Dataset
    df = pd.read_csv ('/content/Dataset9_Diabetes_Prediction.csv')
    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

Data Preprocessing

Missing Values

```
[ ]  df.isnull().sum()

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

Dari database yang didapatkan, missing values diganti dengan nilai ‘0’.

Jadi diperlukan untuk mengganti nilai ‘0’ tersebut dengan NaN value.

```
df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)

#Show NaN Values
df.isnull().sum()

Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

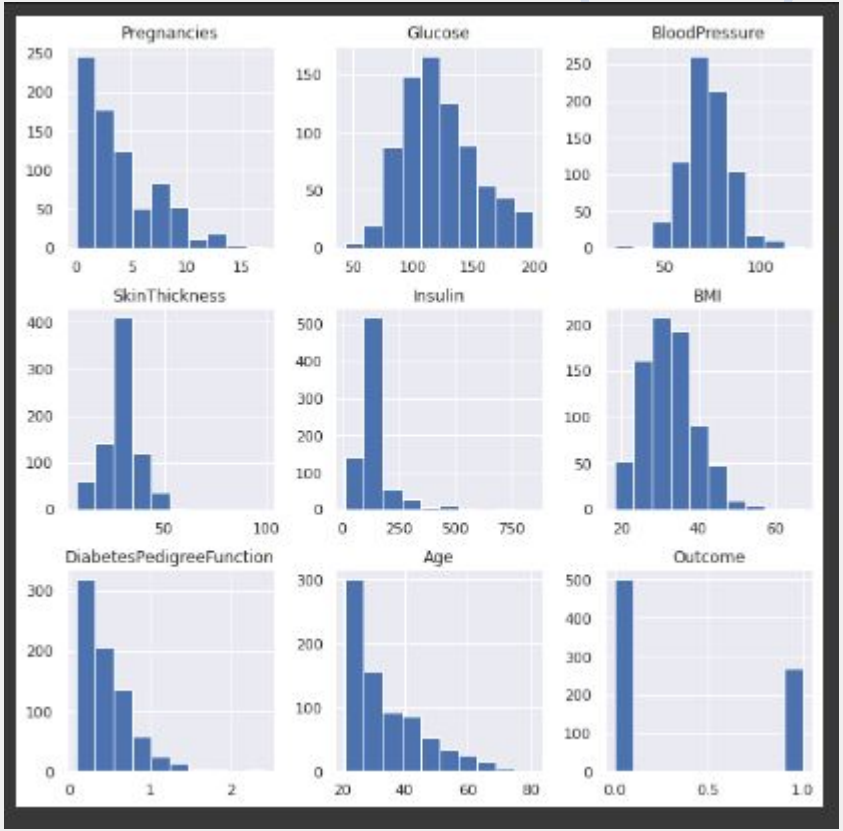
Data Preprocessing

Missing Values

Mengganti nilai NaN dengan nilai median.

```
[ ]  #Computing Median Values
     df['Glucose'].fillna(df['Glucose'].median(), inplace = True)
     df['BloodPressure'].fillna(df['BloodPressure'].median(), inplace = True)
     df['SkinThickness'].fillna(df['SkinThickness'].median(), inplace = True)
     df['Insulin'].fillna(df['Insulin'].median(), inplace = True)
     df['BMI'].fillna(df['BMI'].median(), inplace = True)
     print(df)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6    148.0           72.0           35.0    125.0  33.6
1              1     85.0           66.0           29.0    125.0  26.6
2              8    183.0           64.0           29.0    125.0  23.3
3              1     89.0           66.0           23.0     94.0  28.1
4              0    137.0           40.0           35.0    168.0  43.1
..           ...      ...            ...            ...      ...   ...
763           10    101.0           76.0           48.0    180.0  32.9
764            2    122.0           70.0           27.0    125.0  36.8
765            5    121.0           72.0           23.0    112.0  26.2
766            1    126.0           60.0           29.0    125.0  30.1
767            1     93.0           70.0           31.0    125.0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

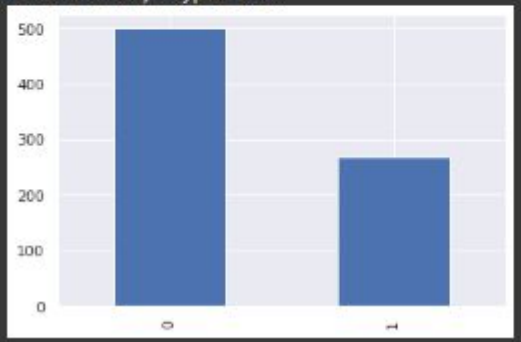


Exploratory Data Analysis

Distribusi Data Outcome

```
color_wheel = {1: "#0392cf", 2: "#7bc043"}
colors = df["Outcome"].map(lambda x: color_wheel.get(x + 1))
print(df.Outcome.value_counts())
p = df.Outcome.value_counts().plot(kind="bar")
```

```
0    500
1    268
Name: Outcome, dtype: int64
```




```
#Outcome Distribution
df['Outcome'].value_counts()*100/len(df)
```

```
0    65.104167
1    34.895833
Name: Outcome, dtype: float64
```

Dapat diketahui bahwa dataset bersifat Imbalanced, jumlah pasien diabetes setengah dari pasien non-diabetes.

Exploratory Data Analysis

Korelasi Data

Correlation value > 0, positive correlation

Correlation value = 0, no correlation

Correlation value < 0, negative correlation

Feature Engineering

Status Diabetes dan Status Bloodpressure

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	Status_Diabetes	Status_Bloodpressure
0	6	148.0	72.0	35.0	125.0	33.6	0.627	50	1	Positif	Normal
1	1	85.0	66.0	29.0	125.0	26.6	0.351	31	0	Negatif	Normal
2	8	183.0	64.0	29.0	125.0	23.3	0.672	32	1	Positif	Normal
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0	Negatif	Normal
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1	Positif	Low

		Status_Bloodpressure	Status_Diabetes
Status_Diabetes	Status_Bloodpressure		
Negatif	High	88	88
	Low	70	70
	Normal	312	312
Positif	High	77	77
	Low	16	16
	Normal	168	168

Modelling

Scale Data

```
sc_X = StandardScaler()
X =  pd.DataFrame(sc_X.fit_transform(df.drop(['Outcome','Status_Diabetes','Status_Bloodpressure'],axis = 1),), columns=['Pregnancies',
'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age'])
X.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	0.639947	0.866045	-0.031990	0.670643	-0.181541	0.166619	0.468492	1.425995
1	-0.844885	-1.205066	-0.528319	-0.012301	-0.181541	-0.852200	-0.365061	-0.190672
2	1.233880	2.016662	-0.693761	-0.012301	-0.181541	-1.332500	0.604397	-0.105584
3	-0.844885	-1.073567	-0.528319	-0.695245	-0.540642	-0.633881	-0.920763	-1.041549
4	-1.141852	0.504422	-2.679076	0.670643	0.316566	1.549303	5.484909	-0.020496

Scaling digunakan untuk menyamakan skala dari beberapa variabel yang berbeda sehingga antara variabel satu dengan yang lain memiliki skala data yang seimbang. StandardScaler yang bertujuan untuk membuat rata-rata 0 dan variansi 1

Modelling

Split Data

```
[30] #Splitting The Dataset
     X = df.drop('Outcome', axis=1)
     y = df['Outcome']

     #Split Dataset to 80:20
     from sklearn.model_selection import train_test_split

     X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.20,
                                                         random_state=25)

     X_train
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
118	4	97.0	60.0	23.0	125.0	28.2	0.443	22
336	0	117.0	72.0	29.0	125.0	33.8	0.932	44
374	2	122.0	52.0	43.0	158.0	36.2	0.816	28
518	13	76.0	60.0	29.0	125.0	32.8	0.180	41
716	3	173.0	78.0	39.0	185.0	33.8	0.970	31
...	...	...	...	...	...	...	...	...
317	3	182.0	74.0	29.0	125.0	30.5	0.345	29
143	10	108.0	66.0	29.0	125.0	32.4	0.272	42
474	4	114.0	64.0	29.0	125.0	28.9	0.126	24
318	3	115.0	66.0	39.0	140.0	38.1	0.150	28
132	3	170.0	64.0	37.0	225.0	34.5	0.356	30

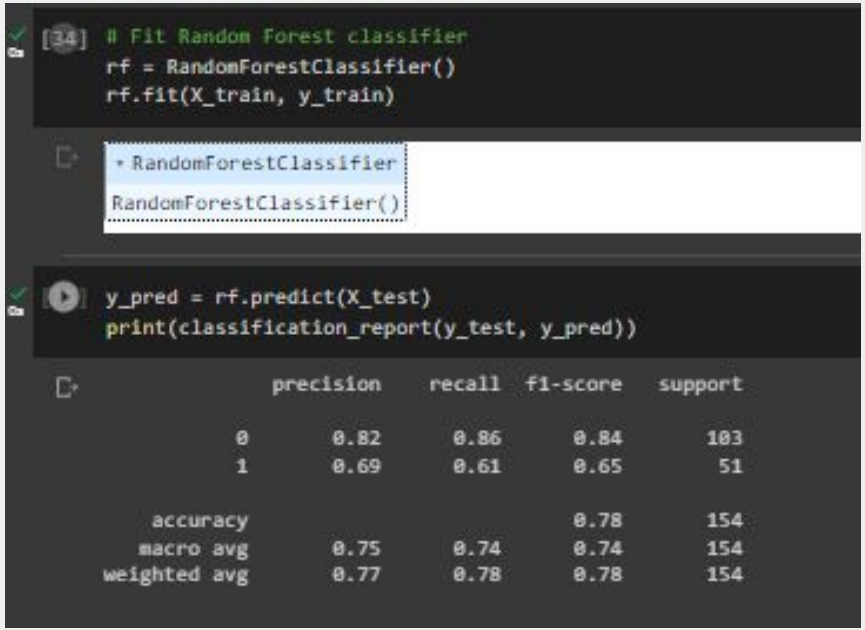
614 rows × 8 columns

```
[31] y_train

118    0
336    0
374    0
518    0
716    1
      ..
317    1
143    1
474    0
318    0
132    1
Name: Outcome, Length: 614, dtype: int64
```

Membagi data menjadi dua jenis yaitu data train dan data set dengan bobot 80:20

Modelling

Random Forest Classifier Model Fit



```
[34] # Fit Random Forest classifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
y_pred = rf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.86      0.84       103
           1       0.69      0.61      0.65        51

    accuracy                           0.78       154
   macro avg       0.75      0.74      0.74       154
weighted avg       0.77      0.78      0.78       154
```

Nilai F1 Score menunjukkan nilai 0.78 yang dapat dikatakan masih tergolong rendah. Jadi diperlukan evaluasi model untuk meningkatkan akurasi tersebut.

Model Evaluation

Resampling Imbalance Dataset - SMOTE

```
[37] y_train.value_counts()

     0    397
     1    217
     Name: Outcome, dtype: int64

[38] y_smote.value_counts()

     0    397
     1    397
     Name: Outcome, dtype: int64

[39] # Fit Random Forest classifier
     rf = RandomForestClassifier()
     rf.fit(X_train, y_train)

     y_pred_rf = rf.predict(X_test)
     print(classification_report(y_test, y_pred_rf))
```

	precision	recall	f1-score	support
0	0.83	0.88	0.86	103
1	0.73	0.65	0.69	51
accuracy			0.81	154
macro avg	0.78	0.77	0.77	154
weighted avg	0.80	0.81	0.80	154

Evaluasi Model dilakukan dengan menggunakan SMOTE dikarenakan kelas tidak seimbang.

SMOTE mensintesis sampel baru dari kelas minoritas untuk menyeimbangkan dataset dengan cara sampling ulang sampel kelas minoritas.

Model Evaluation

Hyper Parameter Tuning

Menggunakan Random Search

```
#Hyper Parameter Tuning
model=RandomForestClassifier(n_estimators=500,criterion='gini',
                             max_features=7,min_samples_leaf=5,random_state=25).fit(X_smote,y_smote)
predictions=model.predict(X_smote)
print(confusion_matrix(y_smote,predictions))
print(accuracy_score(y_smote,predictions))
print(classification_report(y_smote,predictions))
```

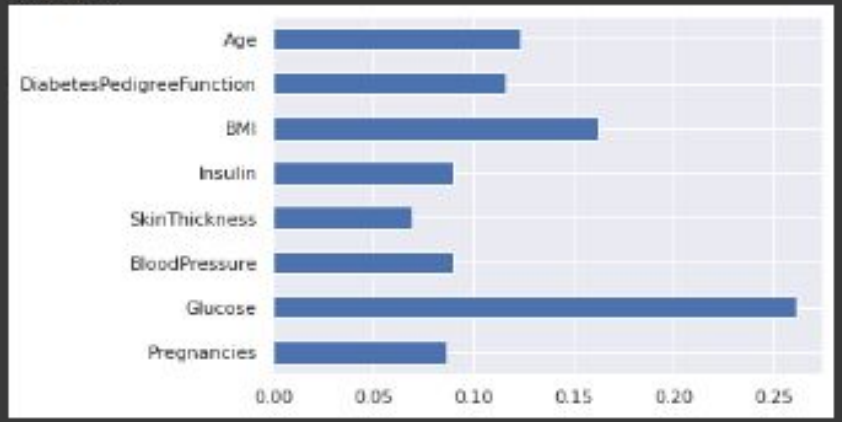
```
[[354  43]
 [ 22 375]]
0.9181360201511335
              precision    recall  f1-score   support

           0       0.94      0.89      0.92       397
           1       0.90      0.94      0.92       397

    accuracy                           0.92       794
   macro avg       0.92      0.92      0.92       794
weighted avg       0.92      0.92      0.92       794
```

Setelah dilakukan evaluasi model dengan menggunakan SMOTE dan Hyper Parameter Tuning, didapatkan akurasi F1 score sebesar 92%

Feature Importance

```
[41] #Plotting feature importances
     (pd.Series(rf.feature_importances_, index=X.columns)
        .plot(kind='barh'))
```

```
<Axes: >
```



Berdasarkan grafik tersebut, diketahui bahwa variabel yang paling berpengaruh terhadap diabetes adalah

1. Glucose (gula dalam tubuh)
2. BMI (Bassal Mass Index)
3. Age (Umur)

Conclusion

1. Model yang digunakan untuk memprediksi seseorang terkena diabetes atau tidak kali ini adalah model Random Forest.
2. Data bersifat Imbalanced, yakni pasien diabetes dan non-diabetes tidak dalam jumlah yang sama sehingga saat model building akurasi yang didapatkan tergolong rendah.
3. SMOTE dan Hyperparameter Tuning dilakukan untuk memperbaiki Imbalanced Dataset. Setelah hal tersebut dilakukan, akurasi meningkat menjadi 92%.
4. Variabel yang paling berpengaruh terhadap penyakit diabetes adalah Glukosa (gula darah), BMI (Bassal Mass Index), dan juga umur.