

Laporan Tugas Case Based Classification

Mata Kuliah Penambangan Data



Nama: Egi Shidqi Rabbani

NIM: 1301190443

Kelas: IF-43-PIL-DS01

Kode Dosen: DDR

Pernyataan :

Dalam pengerjaan tugas ini penulis tidak mengerjakan dengan cara yang melanggar aturan perkuliahan dan kode etik akademisi.

Program Studi Sarjana Informatika

Fakultas Informatika

Universitas Telkom

Bandung

2022

Bagian 1: Data

- **Penjelasan Data**

Data yang digunakan pada tugas ini berasal dari:

<https://www.kaggle.com/datasets/whenamancodes/predict-diabities>

Data tersebut adalah data untuk prediksi penyakit diabetes yang berasal dari National Data Institute of Diabetes and Digestive and Kidney Disease. Seluruh data yang ada di dalam data set adalah data dari pasien wanita yang berumur setidaknya 21 tahun. Seluruh variable pada data set adalah variabel independen dan hanya terdapat satu target variabel dependen, yaitu “Outcome”.

Contoh isi data adalah sebagai berikut.

	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

Gambar 1 Contoh Isi Data

Penjelasan dari setiap variabel pada data adalah sebagai berikut:

Pregnancies : Jumlah kehamilan pada pasien

Glucose : Jumlah glukosa pasien

BloodPressure : Tekanan darah pasien

SkinThickness : Ketebalan kulit pasien

Insulin : Jumlah insulin pasien

BMI : Nilai indeks masa tubuh (Body Mass Index) pasien

DiabetesPedigreeFunction : Nilai dari kerabat pasien yang mengidap diabetes

Outcome : Informasi diabetes pasien (0 = tidak mengidap diabetes, 1 = mengidap diabetes)

- **Eksplorasi Data**

Berikut adalah informasi dari data yang didapat dengan menggunakan .info() pada Google Colab.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Pregnancies            768 non-null   int64
1   Glucose                768 non-null   int64
2   BloodPressure          768 non-null   int64
3   SkinThickness          768 non-null   int64
4   Insulin                768 non-null   int64
5   BMI                   768 non-null   float64
6   DiabetesPedigreeFunction 768 non-null   float64
7   Age                   768 non-null   int64
8   Outcome                768 non-null   int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Gambar 2 Info Data

Penulis juga memeriksa apakah ada data yang duplikat dengan menggunakan .duplicated(). Berikut adalah hasilnya.

```
[59] duplicate = df[df.duplicated()]
      duplicate
      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI  DiabetesPedigreeFunction  Age  Outcome
```

Gambar 3 Duplicate Data

Berdasarkan informasi tersebut dapat diketahui bahwa data berjumlah 768 dengan 9 kolom. Seluruh data adalah data numerik dan tidak ditemukan adanya data yang kosong, serta tidak ada data yang duplikat.

Penulis memeriksa jumlah masing-masing nilai Outcome, hasilnya adalah sebagai berikut.



Gambar 4 Jumlah Nilai Outcome

Dapat dilihat pada Gambar 4 bahwa jumlah target Outcome yang bernilai 0 berjumlah 500 data sedangkan jumlah target Outcome yang bernilai 1 hanya berjumlah 267 data. Berdasarkan informasi tersebut maka diketahui bahwa data adalah data yang *imbalanced* atau tidak seimbang.

Berikut adalah informasi dari data yang didapat dengan menggunakan `.describe()` pada Google Colab.

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

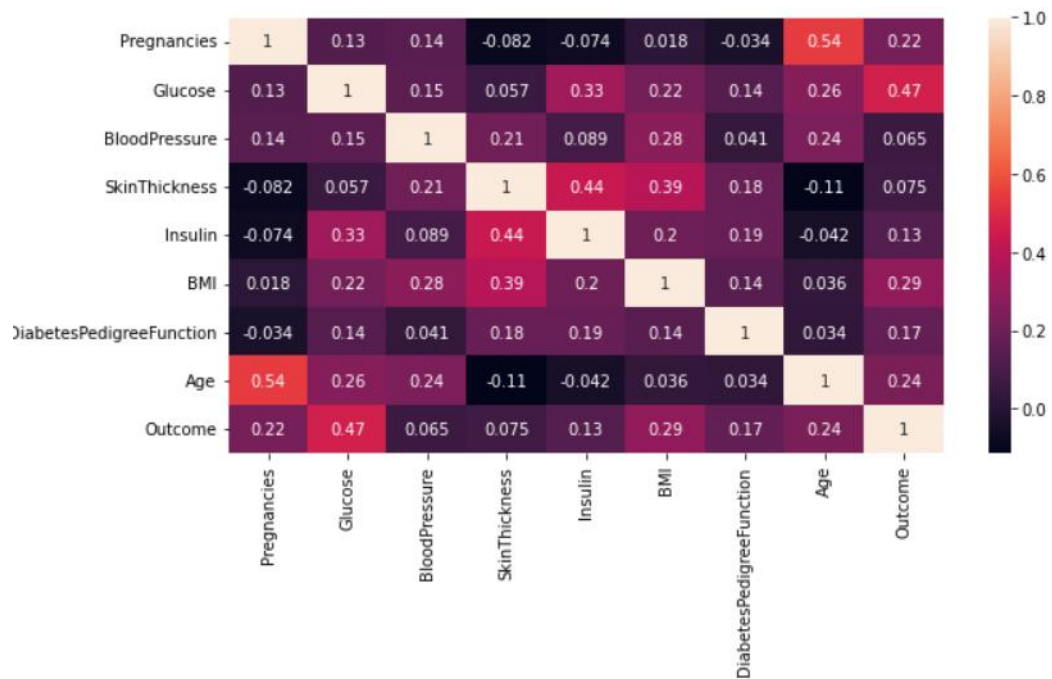
Gambar 5 Describe Data

Dapat dilihat pada Gambar 5 terdapat pasien yang memiliki nilai variabel (Glucose, BloodPressure, SkinThickness, Insulin, BMI) berjumlah 0. Penulis berasumsi bahwa nilai variabel tersebut tidak mungkin dimiliki oleh pasien.

Berikut adalah nilai korelasi data.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

Gambar 6 Tabel Korelasi



Gambar 7 Heatmap Korelasi

Berdasarkan informasi tersebut dapat dilihat bahwa nilai korelasi variabel dengan target bermacam-macam. Variabel yang memiliki nilai korelasi > 0.5 adalah Pregnancies, Glucose, BMI dan Age. Variabel yang memiliki nilai korelasi > 0.4 namun < 0.5 adalah Insulin dan DiabetesPedigreeFunction. Variabel yang memiliki nilai korelasi < 0.4 adalah BloodPressure dan SkinThickness.

Bagian 2: Pre-processing Data

- **Feature Selection**

Setelah penulis mengetahui nilai korelasi masing-masing variabel dengan target, maka penulis memutuskan untuk melakukan *feature selection*. Variabel yang akan digunakan untuk klasifikasi adalah variabel yang memiliki nilai korelasi > 0.2 , di antaranya adalah Pregnancies, Glucose, BMI dan Age.

```
[13] #Drop Data Yang Tidak Digunakan
df_use = df.drop(['BloodPressure', 'SkinThickness', 'Insulin', 'DiabetesPedigreeFunction'], axis=1)
df_use.head()
```

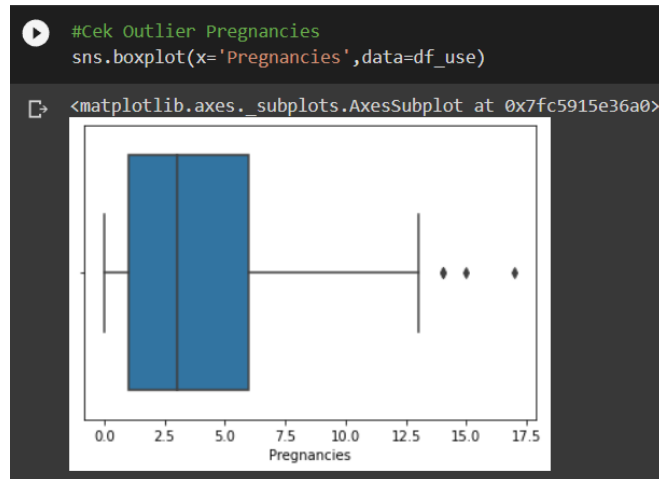
	Pregnancies	Glucose	BMI	Age	Outcome
0	6	148	33.6	50	1
1	1	85	26.6	31	0
2	8	183	23.3	32	1
3	1	89	28.1	21	0
4	0	137	43.1	33	1

Gambar 8 Drop Data

Kode pada Gambar 8 adalah untuk men-*drop* variabel yang tidak akan digunakan, di antaranya adalah BloodPressure, SkinThickness, Insulin dan DiabetesPedigreeFunction. Setelah dilakukan *drop* variabel yang tidak digunakan, data set hanya terdiri dari variabel Pregnancies, Glucose, BMI, Age dan target Outcome.

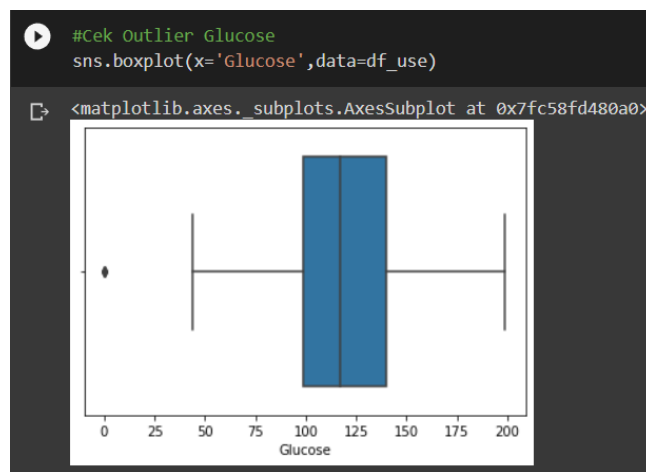
- **Data Cleansing**

Selanjutnya penulis memeriksa outlier pada data dengan menampilkannya pada boxplot. Berikut adalah hasilnya.



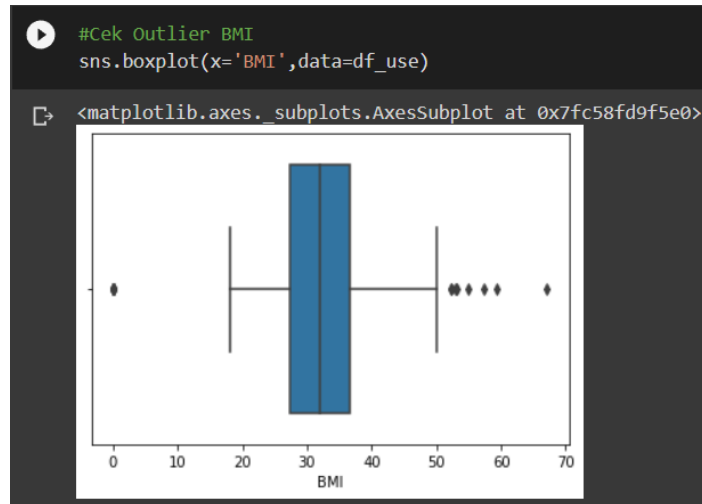
Gambar 9 Outlier Pregnancies

Setelah penulis menampilkan boxplot menggunakan seaborn, dapat dilihat pada Gambar 9 bahwa variabel Pregnancies memiliki outlier.



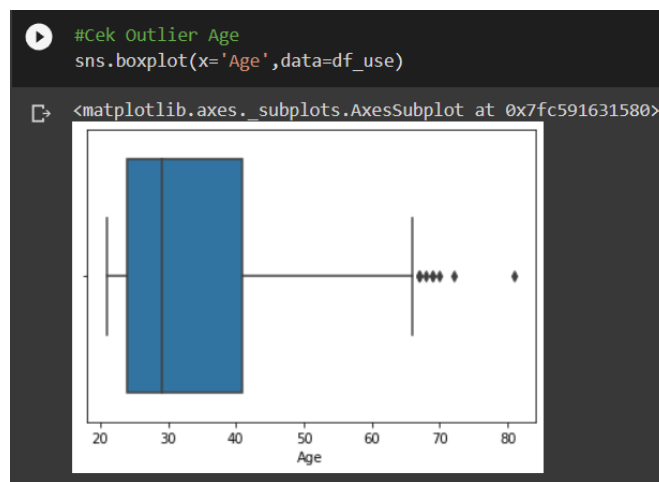
Gambar 10 Outlier Glucose

Setelah ditampilkan boxplot, variabel Glucose juga diketahui terdapat outlier.



Gambar 11 Outlier BMI

Setelah boxplot ditampilkan, diketahui bahwa variabel BMI juga memiliki outlier.



Gambar 12 Outlier Age

Begitu juga dengan variabel Age, setelah ditampilkan boxplot, diketahui terdapat outlier pada variabel Age.

Setelah diketahui bahwa setiap variabel terdapat outlier, penulis melakukan penghitungan nilai batas atas dan batas bawah untuk mengetahui data mana saja yang menjadi outlier.

```

#Data Pregnancies
q1_preg, q3_preg = np.percentile(df_use['Pregnancies'],[25,75])
selisih_preg = q3_preg - q1_preg
ba_preg = q3_preg + (1.5*selisih_preg)
bb_preg = q1_preg - (1.5*selisih_preg)

print('Nilai Q1:', q1_preg)
print('Nilai Q3:', q3_preg)
print('Nilai Selisih:', selisih_preg)
print('Nilai Batas Atas:', ba_preg)
print('Nilai Batas Bawah:', bb_preg)

```

```

Nilai Q1: 1.0
Nilai Q3: 6.0
Nilai Selisih: 5.0
Nilai Batas Atas: 13.5
Nilai Batas Bawah: -6.5

```

Gambar 13 Batas Pregnancies

Kode pada Gambar 13 adalah proses untuk mengetahui nilai batas atas dan batas bawah variabel Pregnancies. Keteranganannya adalah sebagai berikut.

q1_preg : Quartil 1 variabel Pregnancies

q3_preg : Quartil 3 variabel Pregnancies

selisih_preg : Selisih quartil 3 dengan quartil 1

ba_preg : Batas atas variabel Pregnancies

bb_preg : Batas bawah variabel Pregnancies

Nilai batas atas dari variabel Pregnancies adalah 13,5 sedangkan nilai batas bawahnya adalah -6,5. Setelah diketahui nilai batas atas dan batas bawahnya, penulis menjadikan nilai-nilai tersebut sebagai patokan untuk mencari data outlier.

```

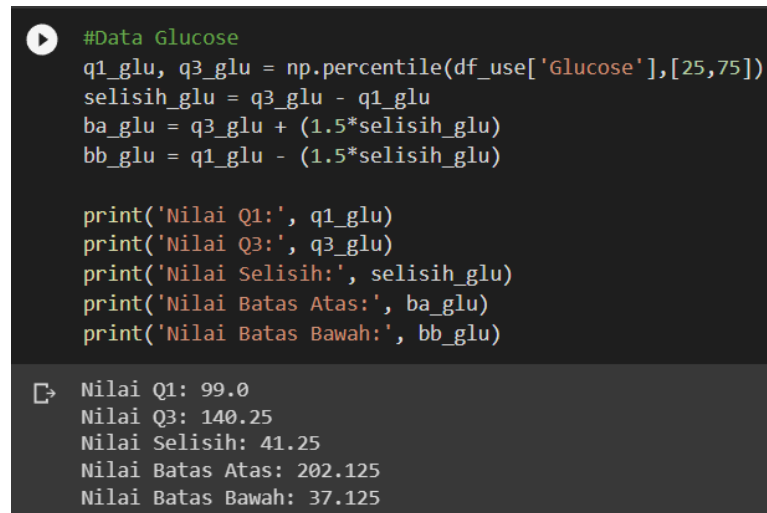
[20] #Data Outlier Pregnancies
out_preg = df_use[(df_use['Pregnancies'] < bb_preg) | (df_use['Pregnancies'] > ba_preg)]
out_preg

```

	Pregnancies	Glucose	BMI	Age	Outcome
88	15	136	37.1	43	1
159	17	163	40.9	47	1
298	14	100	36.6	46	1
455	14	175	33.6	38	1

Gambar 14 Data Outlier Pregnancies

Kode pada Gambar 14 adalah untuk menampilkan data Outlier. Jika nilai variabel Pregnancies kurang dari batas bawah atau lebih dari batas atasnya, maka data tersebut adalah data outlier. Dapat dilihat pada Gambar 14, terdapat 4 buah data outlier pada variabel Pregnancies.



```
#Data Glucose
q1_glu, q3_glu = np.percentile(df_use['Glucose'],[25,75])
selisih_glu = q3_glu - q1_glu
ba_glu = q3_glu + (1.5*selisih_glu)
bb_glu = q1_glu - (1.5*selisih_glu)

print('Nilai Q1:', q1_glu)
print('Nilai Q3:', q3_glu)
print('Nilai Selisih:', selisih_glu)
print('Nilai Batas Atas:', ba_glu)
print('Nilai Batas Bawah:', bb_glu)
```

```
Nilai Q1: 99.0
Nilai Q3: 140.25
Nilai Selisih: 41.25
Nilai Batas Atas: 202.125
Nilai Batas Bawah: 37.125
```

Gambar 15 Batas Glucose

Kode pada Gambar 15 adalah untuk menampilkan nilai batas atas dan batas bawah variabel Glucose. Keterangannya adalah sebagai berikut.

- q1_glu : Quartil 1 variabel Glucose
- q3_glu : Quartil 3 variabel Glucose
- selisih_glu : Selisih quartil 3 dengan quartil 1
- ba_glu : Batas atas variabel Glucose
- bb_glu : Batas bawah variabel Glucose

Nilai batas atas dari variabel Glucose adalah 202,125 sedangkan nilai batas bawahnya adalah 37,125. Setelah diketahui nilai batas atas dan batas bawahnya, penulis menjadikan nilai-nilai tersebut sebagai patokan untuk mencari data outlier.

```
#Data Outlier Glucose
out_glu = df_use[(df_use['Glucose'] < bb_glu) | (df_use['Glucose'] > ba_glu)]
out_glu
```

	Pregnancies	Glucose	BMI	Age	Outcome
75	1	0	24.7	22	0
182	1	0	27.7	21	0
342	1	0	32.0	22	0
349	5	0	41.0	37	1
502	6	0	39.0	41	1

Gambar 16 Data Outlier Glucose

Kode pada Gambar 16 adalah untuk menampilkan data Outlier. Jika nilai variabel Glucose kurang dari batas bawah atau lebih dari batas atasnya, maka data tersebut adalah data outlier. Dapat dilihat pada Gambar 16, terdapat 5 buah data outlier pada variabel Glucose.

```
#Data BMI
q1_bmi, q3_bmi = np.percentile(df_use['BMI'],[25,75])
selisih_bmi = q3_bmi - q1_bmi
ba_bmi = q3_bmi + (1.5*selisih_bmi)
bb_bmi = q1_bmi - (1.5*selisih_bmi)

print('Nilai Q1:', q1_bmi)
print('Nilai Q3:', q3_bmi)
print('Nilai Selisih:', selisih_bmi)
print('Nilai Batas Atas:', ba_bmi)
print('Nilai Batas Bawah:', bb_bmi)
```

```
Nilai Q1: 27.3
Nilai Q3: 36.6
Nilai Selisih: 9.3
Nilai Batas Atas: 50.550000000000004
Nilai Batas Bawah: 13.35
```

Gambar 17 Batas BMI

Kode pada Gambar 17 adalah untuk menampilkan nilai batas atas dan batas bawah variabel BMI. Keterangannya adalah sebagai berikut.

- q1_bmi : Quartil 1 variabel BMI
- q3_bmi : Quartil 3 variabel BMI
- selisih_bmi : Selisih quartil 3 dengan quartil 1
- ba_bmi : Batas atas variabel BMI

bb_bmi : Batas bawah variabel BMI

Nilai batas atas dari variabel BMI adalah 50,55 sedangkan nilai batas bawahnya adalah 13,35. Setelah diketahui nilai batas atas dan batas bawahnya, penulis menjadikan nilai-nilai tersebut sebagai patokan untuk mencari data outlier.

```
#Data Outlier BMI
out_bmi = df_use[(df_use['BMI'] < bb_bmi) | (df_use['BMI'] > ba_bmi)]
out_bmi
```

	Pregnancies	Glucose	BMI	Age	Outcome
9	8	125	0.0	54	1
49	7	105	0.0	24	0
60	2	84	0.0	21	0
81	2	74	0.0	22	0
120	0	162	53.2	25	1
125	1	88	55.0	26	1
145	0	102	0.0	21	0
177	0	129	67.1	26	1
193	11	135	52.3	40	1
247	0	165	52.3	23	0
303	5	115	52.9	28	1
371	0	118	0.0	21	0
426	0	94	0.0	25	0
445	0	180	59.4	25	1
494	3	80	0.0	22	0
522	6	114	0.0	26	0
673	3	123	57.3	22	0
684	5	136	0.0	69	0
706	10	115	0.0	30	1

Gambar 18 Data Outlier BMI

Kode pada Gambar 18 adalah untuk menampilkan data Outlier. Jika nilai variabel BMI kurang dari batas bawah atau lebih dari batas atasnya, maka data tersebut adalah data outlier. Dapat dilihat pada Gambar 18, terdapat 19 buah data outlier pada variabel BMI.

```
#Data Age
q1_age, q3_age = np.percentile(df_use['Age'],[25,75])
selisih_age = q3_age - q1_age
ba_age = q3_age + (1.5*selisih_age)
bb_age = q1_age - (1.5*selisih_age)

print('Nilai Q1:', q1_age)
print('Nilai Q3:', q3_age)
print('Nilai Selisih:', selisih_age)
print('Nilai Batas Atas:', ba_age)
print('Nilai Batas Bawah:', bb_age)

Nilai Q1: 24.0
Nilai Q3: 41.0
Nilai Selisih: 17.0
Nilai Batas Atas: 66.5
Nilai Batas Bawah: -1.5
```

Gambar 19 Batas Age

Kode pada Gambar 19 adalah untuk menampilkan nilai batas atas dan batas bawah variabel Age. Keterangannya adalah sebagai berikut.

q1_age : Quartil 1 variabel Age
q3_age : Quartil 3 variabel Age
selisih_age : Selisih quartil 3 dengan quartil 1
ba_age : Batas atas variabel Age
bb_age : Batas bawah variabel Age

Nilai batas atas dari variabel Age adalah 66,5 sedangkan nilai batas bawahnya adalah -1,5. Setelah diketahui nilai batas atas dan batas bawahnya, penulis menjadikan nilai-nilai tersebut sebagai patokan untuk mencari data outlier.

```
#Data Outlier Age
out_age = df_use[(df_use['Age'] < bb_age) | (df_use['Age'] > ba_age)]
out_age
```

	Pregnancies	Glucose	BMI	Age	Outcome
123	5	132	26.8	69	0
363	4	146	38.5	67	1
453	2	119	19.6	72	0
459	9	134	25.9	81	0
489	8	194	26.1	67	0
537	0	57	21.7	67	0
666	4	145	32.5	70	1
674	8	91	35.6	68	0
684	5	136	0.0	69	0

Gambar 20 Data Outlier Age

Kode pada Gambar 20 adalah untuk menampilkan data Outlier. Jika nilai variabel Age kurang dari batas bawah atau lebih dari batas atasnya, maka data tersebut adalah data outlier. Dapat dilihat pada Gambar 20, terdapat 19 buah data outlier pada variabel Age.

```
#Drop Data Outlier
df_nonout = df_use.drop([88, 159, 298, 455,
                        75, 182, 342, 349,
                        502, 9, 49, 60,
                        81, 120, 125, 145,
                        177, 193, 247, 303,
                        371, 426, 445, 494,
                        522, 673, 684, 706,
                        123, 363, 453, 459,
                        489, 537, 666, 674])
df_nonout.head()
```

	Pregnancies	Glucose	BMI	Age	Outcome
0	6	148	33.6	50	1
1	1	85	26.6	31	0
2	8	183	23.3	32	1
3	1	89	28.1	21	0
4	0	137	43.1	33	1

Gambar 21 Drop Outlier

Setelah data-data outlier diketahui, maka selanjutnya adalah men-*drop* data-data outlier tersebut. Kode pada Gambar 21 adalah untuk men-*drop* data-data yang sudah diketahui sebagai outlier. Nilai yang terdapat pada kode adalah index dari data-data outlier.

```
df_nonout.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 732 entries, 0 to 767
Data columns (total 5 columns):
#   Column          Non-Null Count  Dtype  
---  -
0   Pregnancies     732 non-null   int64   
1   Glucose         732 non-null   int64   
2   BMI             732 non-null   float64  
3   Age             732 non-null   int64   
4   Outcome         732 non-null   int64   
dtypes: float64(1), int64(4)
memory usage: 34.3 KB
```

Gambar 22 Info Non-Outlier

Setelah menyingkirkan data-data outlier, tersisa 732 data dan 5 kolom pada dataset, seperti yang terlihat pada Gambar 22.

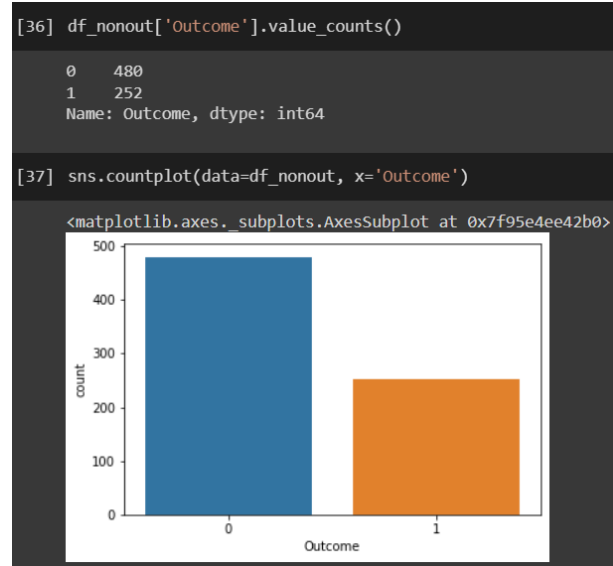
	count	mean	std	min	25%	50%	75%	max
Pregnancies	732.0	3.792350	3.276672	0.0	1.0	3.0	6.000	13.0
Glucose	732.0	121.599727	30.455092	44.0	99.0	117.0	141.000	199.0
BMI	732.0	32.215164	6.488070	18.2	27.5	32.0	36.425	50.0
Age	732.0	32.924863	11.147860	21.0	24.0	29.0	40.000	66.0
Outcome	732.0	0.344262	0.475452	0.0	0.0	0.0	1.000	1.0

Gambar 23 Describe Non-Outlier

Setelah data outlier disingkirkan, nilai-nilai pada masing-masing variabel pun menjadi lebih masuk akal.

- **Balancing Data**

Setelah data outlier disingkirkan, jumlah dari masing-masing nilai Outcome adalah sebagai berikut.



Gambar 24 Nilai Outcome Non-Outlier

Dapat dilihat pada Gambar 24 bahwa jumlah target Outcome yang bernilai 0 berjumlah 480 data sedangkan jumlah target Outcome yang bernilai 1 hanya berjumlah 252 data. Data tersebut tidak seimbang, oleh karena itu dilakukan penyeimbangan data dengan *men-drop* Sebagian data yang nilai target Outcome nya adalah 0.

```
[38] #Menghapus Sebagian Data Outcome "0" Agar Seimbang
df_balanced = (df_nonout.groupby('Outcome', as_index=False)
               .apply(lambda x: x.sample(n=252))
               .reset_index(drop=True))
```

Gambar 25 Code Balance

Kode pada Gambar 25 adalah untuk menyeimbangkan data. Jumlah masing-masing nilai Outcome ditentukan menjadi hanya 252 untuk masing-masing nilai.

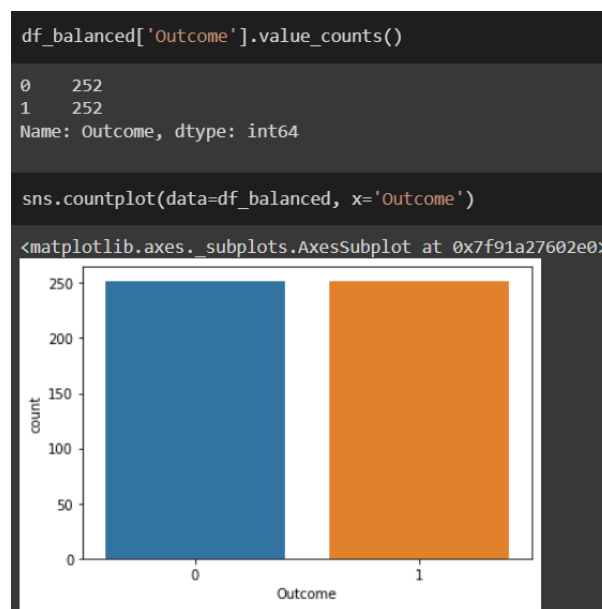
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 504 entries, 0 to 503
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Pregnancies  504 non-null    int64
1   Glucose      504 non-null    int64
2   BMI          504 non-null    float64
3   Age         504 non-null    int64
4   Outcome      504 non-null    int64
dtypes: float64(1), int64(4)
memory usage: 19.8 KB

```

Gambar 26 Info Balance

Setelah dilakukan penyeimbangan data, dapat dilihat pada Gambar 26 total data berubah menjadi 504 data dengan 5 kolom.



Gambar 27 Outcome Balance

Seperti yang telah dibuktikan pada Gambar 27, kini data set sudah seimbang, masing-masing nilai target Outcome berjumlah 252 data.

- **Splitting dan Normalisasi Data**

Setelah data bersih dan seimbang, selanjutnya adalah memisahkan data independen dengan data dependen sebelum selanjutnya dilakukan pelatihan dan pengujian dengan model *machine learning*.

```
[44] x = df_balanced.drop('Outcome', axis=1)
      y = df_balanced['Outcome']

[45] x.head()
```

	Pregnancies	Glucose	BMI	Age
0	5	147	29.9	28
1	2	141	25.4	24
2	1	119	35.5	25
3	2	96	21.1	26
4	2	108	30.8	21

```
[46] y.head()

0    0
1    0
2    0
3    0
4    0
Name: Outcome, dtype: int64
```

Gambar 28 Data Independen dan Dependen

Kode pada Gambar 28 adalah untuk memisahkan data independen dan dependen. Data independen di-assign ke dalam variabel x sedangkan data dependen di-assign ke dalam variabel y.

Diketahui bahwa nilai masing-masing variabel pada data set sangat beragam, untuk memaksimalkan hasil pemodelan *machine learning*, dilakukan normalisasi data yang bertujuan untuk mengubah skala dari nilai masing-masing variabel. Pada tugas ini, penulis menggunakan MinMaxScaler untuk menormalisasi data.

```
#Normalisasi data menggunakan MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0,1)) #Skala 0-1
x = scaler.fit_transform(x)
x = pd.DataFrame(x)
x.head()
```

	0	1	2	3
0	0.076923	0.380645	0.037736	0.022222
1	0.307692	0.251613	0.349057	0.288889
2	0.076923	0.174194	0.471698	0.000000
3	0.230769	0.400000	0.399371	0.066667
4	0.000000	0.329032	0.830189	0.022222

Gambar 29 MinMaxScaler

Kode pada Gambar 29 adalah untuk melakukan normalisasi data pada data dependen, sehingga skala nya adalah 0-1. MinMaxScaler ini diambil dari library yang sudah ada pada python yaitu SKlearn.

```
#Splitting data train dan test
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.3, random_state=0) #jumlah test 30% dan train 70%
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(352, 4)
(152, 4)
(352,)
(152,)
```

Gambar 30 Split Data

Selanjutnya adalah memisahkan data menjadi data *train* dan data *test*. Dapat dilihat pada kode di Gambar 30, penulis menggunakan library SKlearn untuk melakukan *splitting data*. Penulis menentukan ukuran data *test* sebesar 30% dari total data, sedangkan 70% sisanya adalah data *train*.

Bagian 3: Algoritma KNN (K-Nearest Neighbour)

Algoritma KNN adalah algoritma *supervised learning* yang menggunakan *distance function* untuk menghitung ketetanggaan. Algoritma KNN dapat digunakan baik untuk klasifikasi maupun regresi. KNN ini mengklasifikasikan/memprediksi sesuatu ke dalam kelas tertentu berdasarkan kelas mayoritas ketetanggaan.

Terdapat 3 *distance function* untuk menghitung jarak dalam algoritma KNN, di antaranya adalah Euclidian, Manhattan dan Minkowski.

Rumus dari Euclidian adalah sebagai berikut:

$$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$$

Rumus 1 Rumus Euclidian

Rumus dari Manhattan adalah sebagai berikut:

$$\sum_{i=1}^k |x_i - y_i|$$

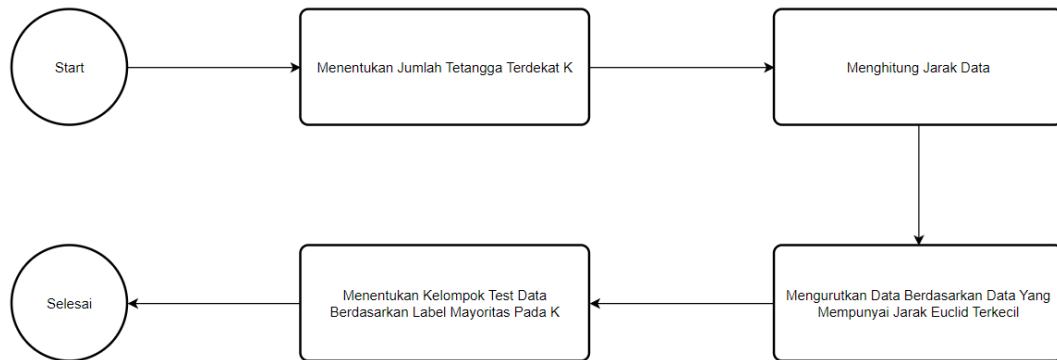
Rumus 2 Rumus Manhattan

Rumus dari Minkowski adalah sebagai berikut:

$$\left(\sum_{i=1}^k (|x_i - y_i|)^q \right)^{\frac{1}{q}}$$

Rumus 3 Rumus Minkowski

Alur kerja dari algoritma KNN adalah sebagai berikut.



Gambar 31 Flowchart Algoritma KNN

1. Menentukan jumlah tetangga terdekat K
Jumlah K (ketertanggaaan terdekat) ditentukan secara manual oleh *user*.
User harus menentukan berapa jumlah K yang kira-kira optimal.
2. Menghitung jarak data
Melakukan penghitungan jarak dengan menggunakan salah satu *distance function*.
3. Mengurutkan data berdasarkan data yang mempunyai jarak Euclid terkecil
Data yang telah dihitung jaraknya menggunakan *distance function* akan diurutkan mulai dari yang terkecil hingga terbesar.
4. Menentukan kelompok data berdasarkan label mayoritas pada K
Tetangga (K) yang paling dekat dengan kelompok tertentu akan dimasukkan/diprediksi menjadi kelompok tersebut.

```
#Klasifikasi dengan KNN
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
knn = KNeighborsClassifier(n_neighbors=9) #Jumlah K=9
knn.fit(x_train, y_train)

#Nilai akurasi (%)
print("Train Set Accuracy: "+str(accuracy_score(y_train, knn.predict(x_train))*100))
print("Test Set Accuracy: "+str(accuracy_score(y_test,knn.predict(x_test))*100))

Train Set Accuracy: 78.125
Test Set Accuracy: 80.26315789473685
```

Gambar 32 Code KNN

Pada tugas ini, penulis menggunakan library SKlearn untuk mengklasifikasi menggunakan KNN. Dapat dilihat pada kode di Gambar 32, jumlah tetangga yang digunakan adalah 9. Setelah melakukan *fitting* data terhadap model, dapat diperoleh nilai akurasi untuk data *train* sebesar 78% sedangkan nilai akurasi untuk data *test* sebesar 80%.

- **Link Code**

Code penulis yang ada di laporan ini dapat dilihat melalui:

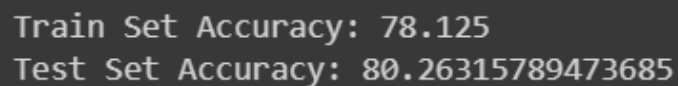
<https://colab.research.google.com/drive/1Tr-RTIsT9H-pQmy0G-0dnP91iSnKjfQZ?usp=sharing>

Referensi code yang penulis gunakan berasal dari:

<https://www.youtube.com/watch?v=ET6fESozek4&t=1470s>

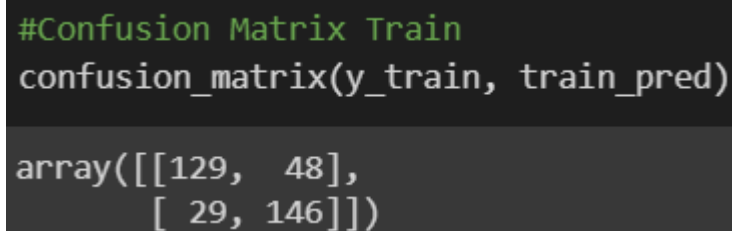
Bagian 4: Evaluasi

Performansi dari algoritma KNN (K-Nearest Neighbour) dalam mengklasifikasikan data diabetes cukup baik. Nilai akurasi yang diperoleh untuk data *train* adalah sebesar 78% sedangkan nilai akurasi yang diperoleh untuk data *test* adalah sebesar 80%.



```
Train Set Accuracy: 78.125
Test Set Accuracy: 80.26315789473685
```

Gambar 33 Hasil Akurasi



```
#Confusion Matrix Train
confusion_matrix(y_train, train_pred)

array([[129,  48],
       [ 29, 146]])
```

Gambar 34 Confusion Matrix Training

Gambar 34 menunjukkan *confusion matrix* hasil klasifikasi data *train*. Dapat dilihat jumlah *true positive* (pasien diabetes yang diprediksi diabetes) adalah

129 sedangkan jumlah *false positive* (pasien non-diabetes yang diprediksi diabetes) berjumlah 48. Jumlah *true negative* (pasien non-diabetes yang diprediksi non-diabetes) adalah 146 sedangkan jumlah *false negative* (pasien diabetes yang diprediksi non-diabetes) adalah 29.

```
#Confusion Matrix Test
confusion_matrix(y_test, test_pred)

array([[60, 15],
       [15, 62]])
```

Gambar 35 Confusion Matrix Test

Gambar 35 menunjukkan *confusion matrix* hasil klasifikasi data *train*. Dapat dilihat jumlah *true positive* (pasien diabetes yang diprediksi diabetes) adalah 60 sedangkan jumlah *false positive* (pasien non-diabetes yang diprediksi diabetes) berjumlah 15. Jumlah *true negative* (pasien non-diabetes yang diprediksi non-diabetes) adalah 62 sedangkan jumlah *false negative* (pasien diabetes yang diprediksi non-diabetes) adalah 15.

Berdasarkan nilai akurasi yang diperoleh, algoritma KNN (K-Nearest Neighbour) adalah algoritma yang cukup baik dalam memprediksi diabetes berdasarkan data yang digunakan. Nilai akurasi dari algoritma juga dapat dipengaruhi oleh berbagai faktor, seperti pada variabel yang digunakan, banyak datanya yang digunakan, pembagian data *train* dan data *test* atau jumlah K (ketetanggaan). *Preprocessing data* yang lebih akan menghasilkan performansi model yang lebih baik juga.