2. Tahapan analysa data observasi terhadap penyakit Diabetes

2.1 Latar belakang

Diabetes merupakan penyakit yang terjadi ketika kadar glukosa darah menjadi tinggi yang pada akhirnya menimbulkan gangguan kesehatan lain seperti penyakit jantung, penyakit ginjal, dll. Diabetes disebabkan terutama karena konsumsi makanan olahan, kebiasaan konsumsi yang buruk, dll. Menurut WHO , jumlah penderita diabetes terus meningkat dari tahun ke tahun.

Dalam analysis berikut ini akan menggambarkan bagaimana data terkait diabetes dapat dimanfaatkan untuk memprediksi apakah seseorang menderita diabetes atau tidak. Lebih khusus lagi, artikel ini akan berfokus pada bagaimana machine learning dapat dimanfaatkan untuk memprediksi penyakit seperti diabetes. Anda akan dapat memahami konsep-konsep seperti eksplorasi data, pembersihan data, pemilihan fitur, pemilihan model, evaluasi model dan menerapkannya dalam cara yang praktis.

Dengan memperhatikan tingginya pengaruh penyalit diabetes terhadap kematian manusia di dunia, penelitian terhadap pencegahan dan penyembuhan penyakit ini telah banyak di lakukan oleh para ilmuwan kedokteran. Salah satu data penelitian telah dinyatakan di internet terhadap 768 orang wanita dengan 9 factor dari National Institute of Diabetes and Digestive and Kidney Diseases dicatat baik terhadap pasien yang terdiagnosis diabetes atau tidak, dan beberapa faktor tersebut akan ditinjau detile dalam analysis ini. Adapun factor factor selengkap[nya adalah :

- `Kehamilan` : Banyaknya kehamilan yang pernah di alami
- `Glukosa` : Konsentrasi glukosa dalam 2 jam test toleransi glukosa dalam uji gula darah
- `Tekanan Darah` : tekanan darah dalam diastolik (mm/Hg)
- `Tebal Kulit` : ketebalan lipatan dalam otot di bawah kulit (mm)
- `Insulin` : serapan serum insulin dalam2 jam (mu U/ml)
- `BMI(Body Mass Index)` : berat dalam kg/tinggi dalam meter kuadrat
- `Fungsi Pedigree` : sejarah pernah diabetes mellitus yang relatif terkait dengan faktor genetik/keturunan
- `Umur` : usia pasien dalam (tahun)
- `Target Output` : class variable (0 atau 1); `1 untuk terdiagnotis diabetes` dan `0 untuk non diabetes`

> `Prerequisites`

- Python 3.+
- Anaconda (Scikit Learn, Numpy, Pandas, Matplotlib, Seaborn)
- Jupyter Notebook.
- Basic understanding of supervised machine learning methods: specifically classification.

2.2 Fase 0 — Data Preparation

Tugas anda memperoleh dan menyiapkan kumpulan data. Meskipun data melimpah di era ini, namun masih sulit menemukan kumpulan data yang sesuai dengan permasalahan yang ingin Anda atasi. Jika tidak ditemukan kumpulan data yang sesuai, Anda mungkin harus membuatnya sendiri.

Dalam analysis ini kita tidak akan membuat kumpulan data kita sendiri, melainkan kita akan menggunakan kumpulan data yang sudah ada yang disebut " Pima Indians Diabetes Database " yang disediakan oleh UCI Machine Learning Repository (repositori terkenal untuk kumpulan data pembelajaran mesin). Kami akan menjalankan alur kerja pembelajaran mesin dengan kumpulan Data Diabetes yang disediakan di atas.

2.3 Fase 1 — Eksplorasi Data

Ketika dihadapkan dengan suatu kumpulan data, pertama-tama kita harus menganalisis dan " **mengenal** " kumpulan data tersebut. Langkah ini diperlukan untuk membiasakan diri dengan data, untuk mendapatkan pemahaman tentang fitur potensial dan untuk melihat apakah pembersihan data diperlukan.

Pertama, kita akan `mengimpor library yang diperlukan` dan `mengimpor kumpulan data kita` ke notebook Jupyter. Kita dapat mengamati kolom-kolom yang disebutkan dalam kumpulan data.

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import os
import opendatasets as od
import matplotlib.pyplot as plt
plt.style.use("ggplot")  #using style ggplot

# disable Anaconda warnings
import warnings
warnings.simplefilter('ignore')

%matplotlib inline
```

In [2]:
```
# Download the data set files

```

```
# Assign the Kaggle data set URL into variable
dataset = 'https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-datab
# Using opendatasets let's download the data sets
od.download(dataset)
```

```
Skipping, found downloaded files in "./pima-indians-diabetes-database" (use
force=True to force download)
```

In [3]:
```
data_dir ='./pima-indians-diabetes-database'
```

In [4]:
```
os.listdir(data_dir)
```

Out[4]:
```
['diabetes.csv']
```

In [5]:
```
#Importing the dataset and view column

diabetes = pd.read_csv('./pima-indians-diabetes-database/diabetes.csv')
diabetes.columns
```

Out[5]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insuli
n',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [6]:
```
# Menampilkan Data
diabetes.head()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

In [7]:
```
# Tampilkan dimensi data
print(" Dimensi data diabetes :",format(diabetes.shape))
```

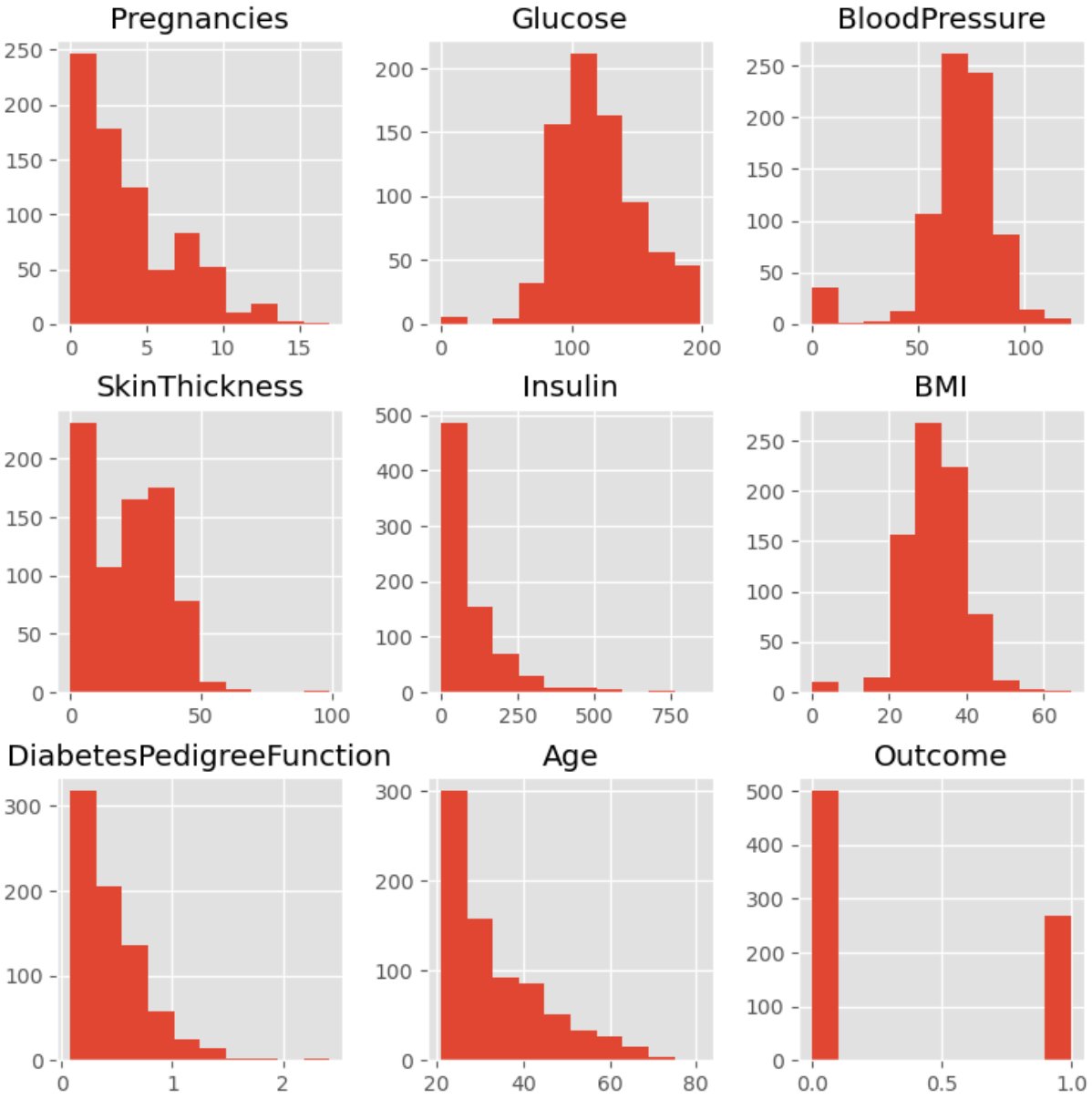
```
Dimensi data diabetes : (768, 9)
```

In [8]:
```
# Identifikasiberapa jumlah penderita diabetes dan non diabetes dengan penge
diabetes.groupby(['Outcome']).size()
```

Out[8]:
```
Outcome
0    500
1    268
dtype: int64
```

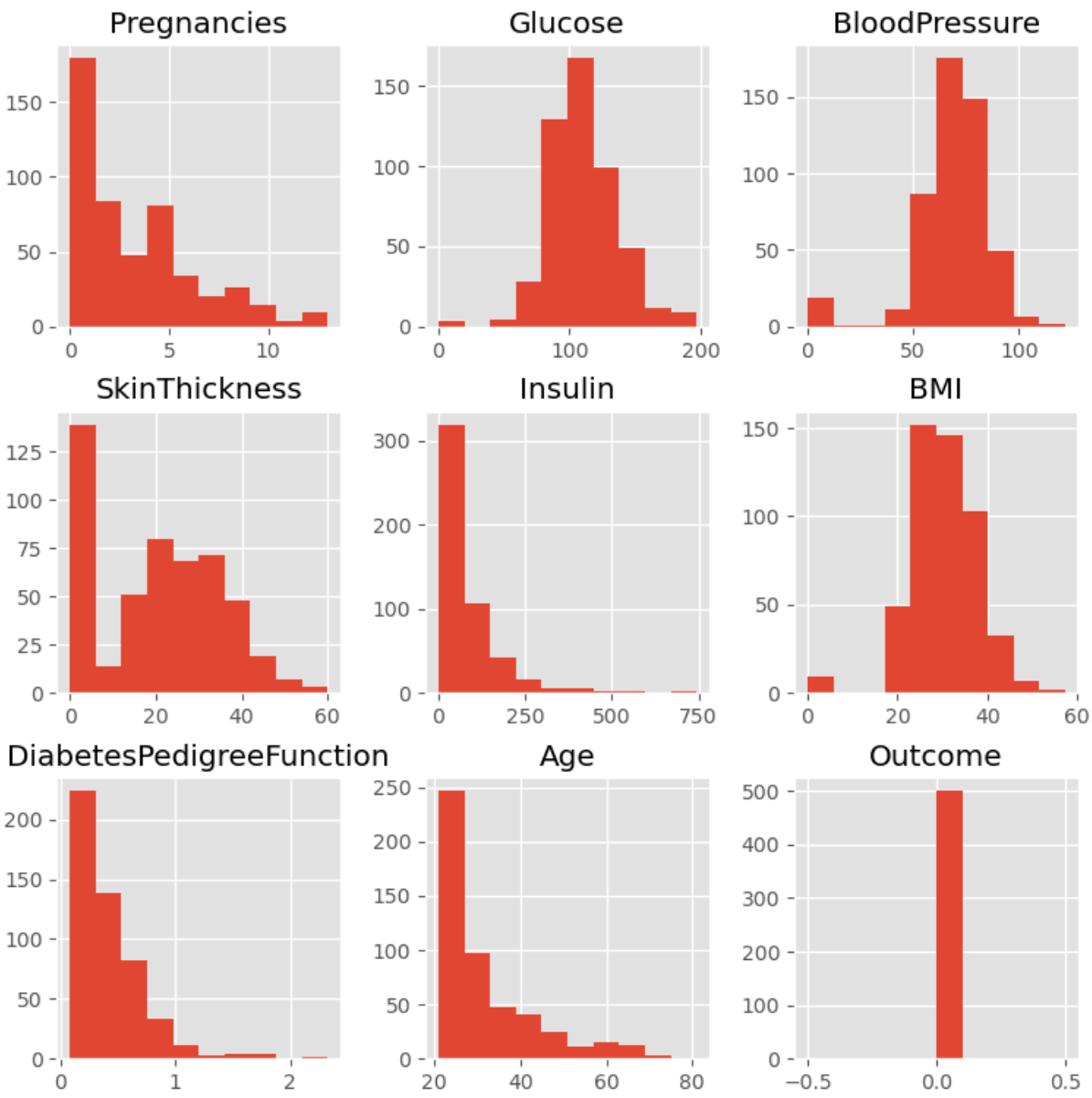
In [9]:
```
# menampilkan grafik histogram diabetes
diabetes.hist(figsize=(9, 9))
```

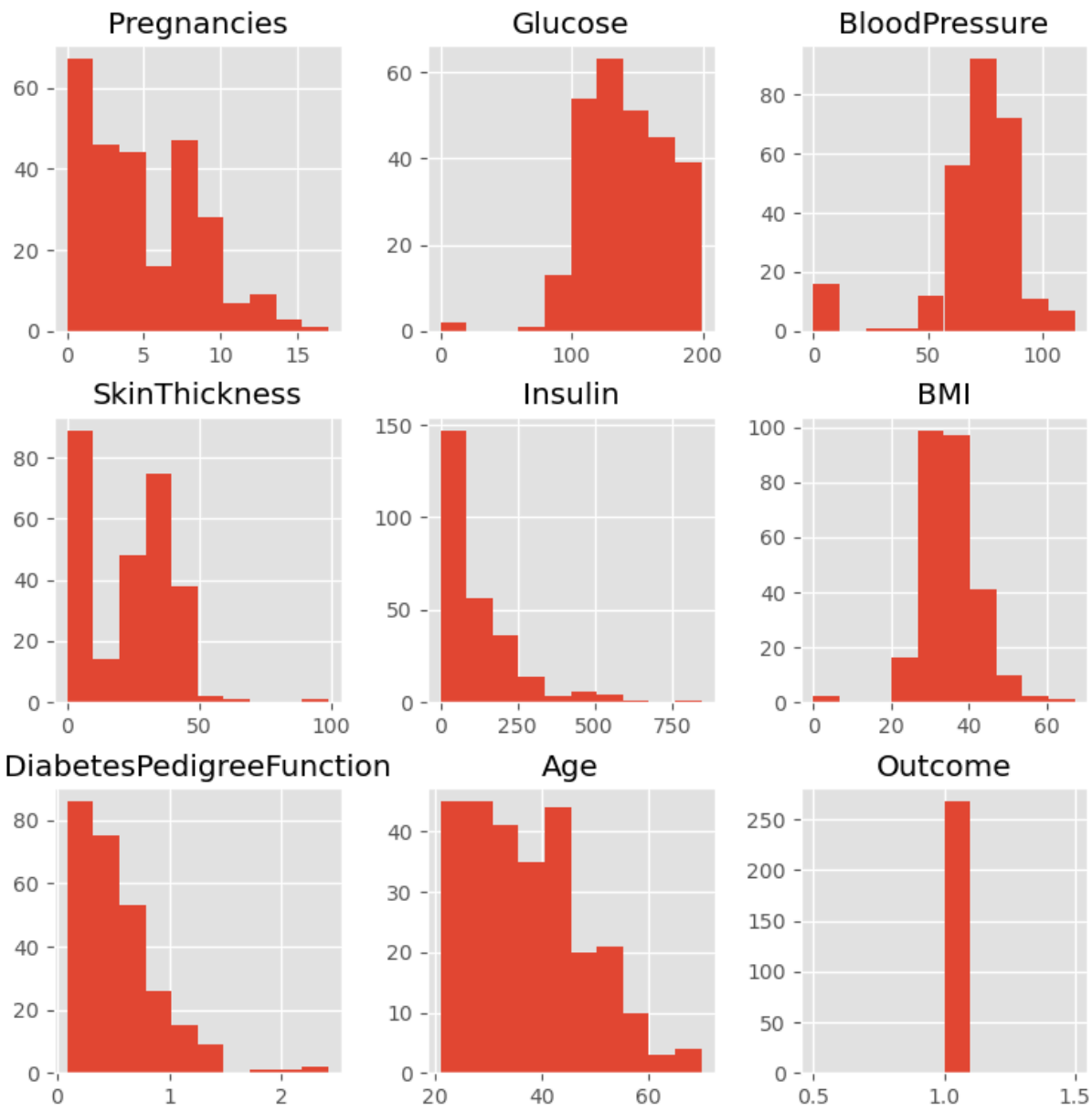
Out[9]:
```
array([[<Axes: title={'center': 'Pregnancies'}>,
        <Axes: title={'center': 'Glucose'}>,
        <Axes: title={'center': 'BloodPressure'}>],
       [<Axes: title={'center': 'SkinThickness'}>,
        <Axes: title={'center': 'Insulin'}>,
        <Axes: title={'center': 'BMI'}>],
       [<Axes: title={'center': 'DiabetesPedigreeFunction'}>,
        <Axes: title={'center': 'Age'}>,
        <Axes: title={'center': 'Outcome'}>]], dtype=object)
```



In [10]:
```
# Menampilkan data histogram factor yang mempengaruhi diabetes
diabetes.groupby(['Outcome']).hist(figsize=(9,9))
```

Out[10]:
```
Outcome
0    [[Axes(0.125,0.666111;0.215278x0.213889), Axes...
1    [[Axes(0.125,0.666111;0.215278x0.213889), Axes...
dtype: object
```


Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome



2.4 Phase 2 — Data Cleaning

Ada beberapa faktor yang perlu dipertimbangkan dalam proses pembersihan data.

- Pengamatan duplikat atau tidak relevan.
- Pelabelan data yang buruk, kategori yang sama muncul berkali-kali.
- Ada data hilang atau null values.
- Adanya outliner yang tidak terduga.

> Kita dapat menemukan data data yang hilang atau null values dari kumpulan data (jika ada) menggunakan fungsi pandas berikut.

In [11]:
```
diabetes.isnull().sum()
```

Out[11]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [12]:
```
diabetes.isna().sum()
```

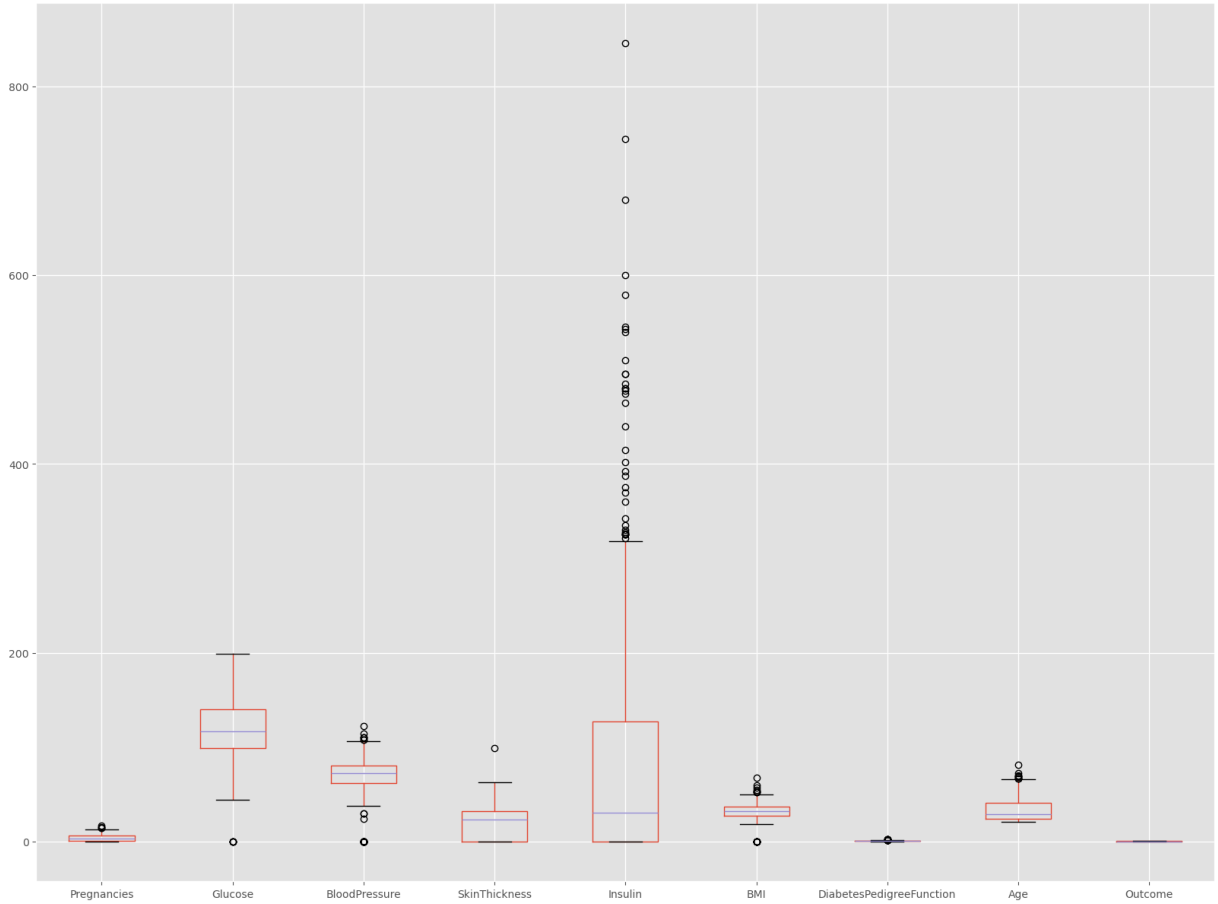
Out[12]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Outliers yang tak di kehendaki muncul

Saat menganalisis histogram kita dapat mengidentifikasi bahwa terdapat beberapa outlier di beberapa kolom. Kami akan menganalisis lebih lanjut outlier tersebut dan menentukan apa yang dapat kami lakukan untuk mengatasinya.

In [13]:
```
# mendetksi secara garis besar adanya outliner di datset diabetes
diabetes.boxplot(figsize=(20,15))
```

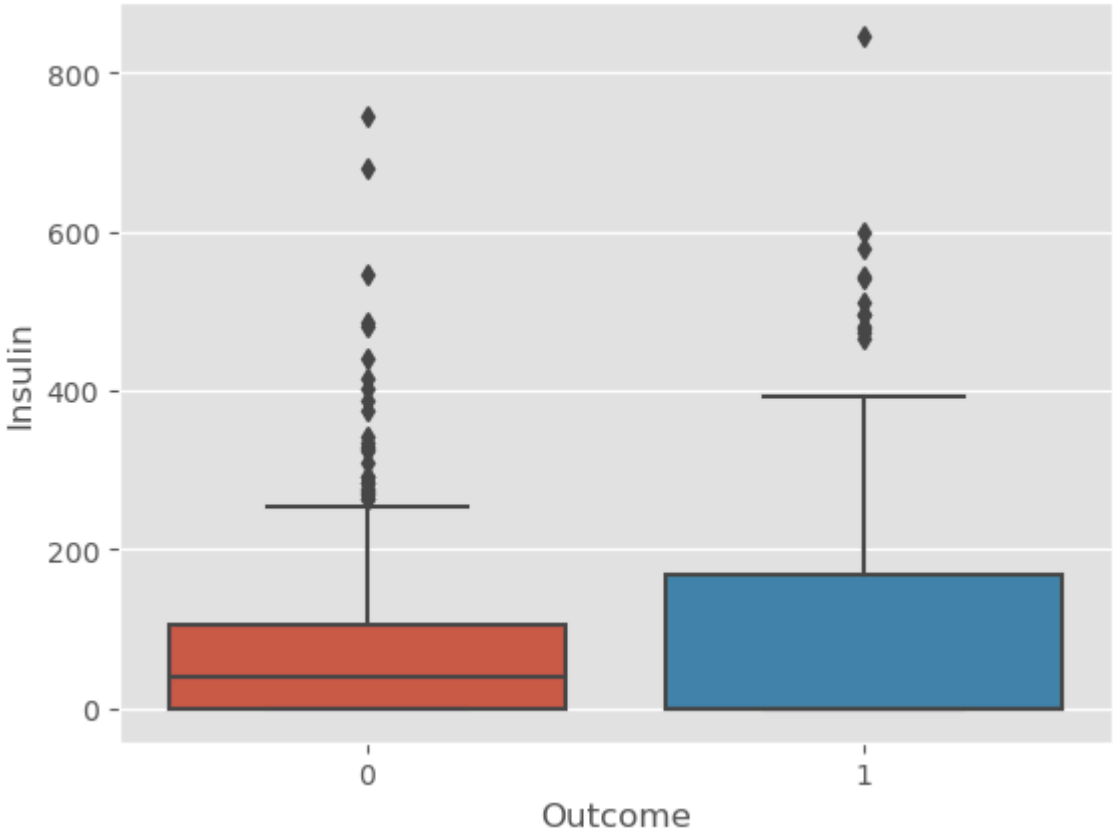
Out[13]:
```
<Axes: >
```



`Insulin` : Dalam situasi yang jarang terjadi, seseorang mungkin tidak mempunyai insulin tetapi dengan mengamati data, kita dapat menemukan bahwa ada total 374 hitungan.

In [14]:
```
sns.boxplot(y='Insulin', x='Outcome', data=diabetes)
```

Out[14]:
```
<Axes: xlabel='Outcome', ylabel='Insulin'>
```



In [15]:
```
print("Total : ", diabetes[diabetes.Insulin == 0].shape[0])
```

```
Total :  374
```

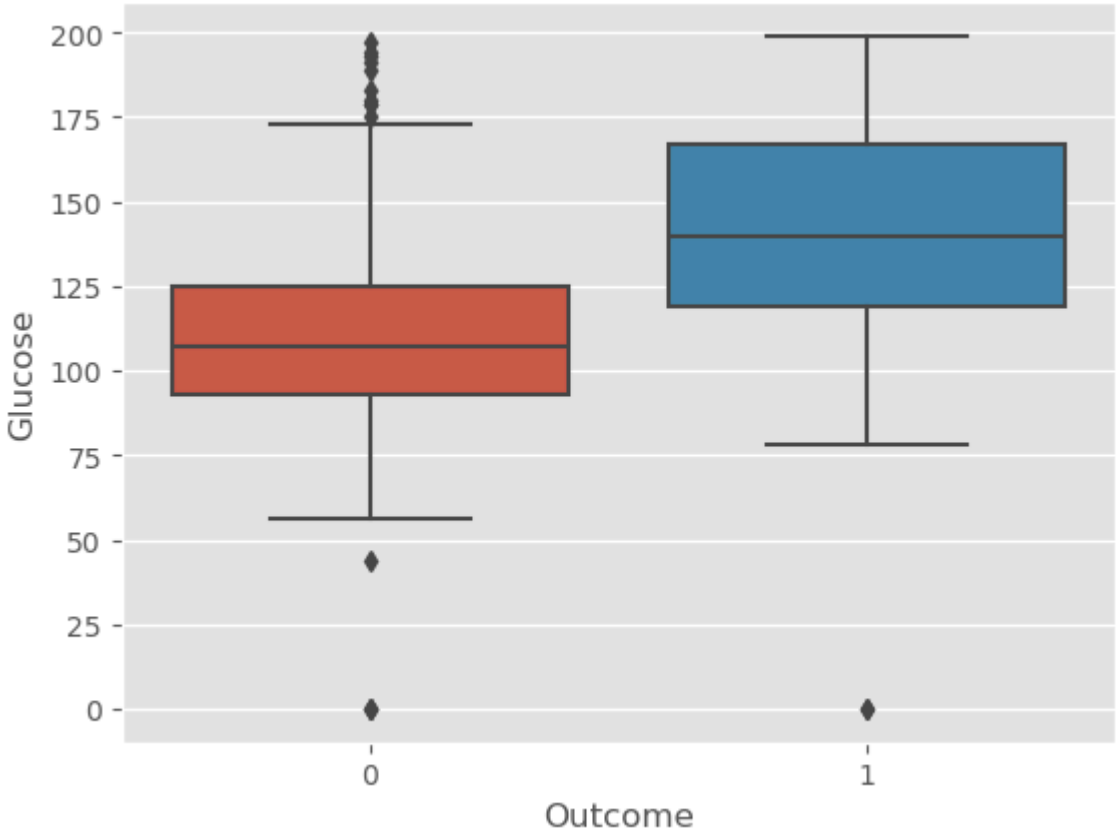
In [16]:
```
print(diabetes[diabetes.Insulin == 0].groupby('Outcome')['Age'].count())
```

```
Outcome
0    236
1    138
Name: Age, dtype: int64
```

`Kadar glukosa plasma` : Bahkan setelah puasa, kadar glukosa tidak akan serendah nol. Oleh karena itu nol adalah pembacaan yang tidak valid. Dengan mengamati data kita dapat melihat 5 hitungan yang nilainya 0.

In [17]:
```
sns.boxplot(y='Glucose', x='Outcome', data=diabetes)
```

Out[17]:
```
<Axes: xlabel='Outcome', ylabel='Glucose'>
```



In [18]:
```
print("Total : ", diabetes[diabetes.Glucose == 0].shape[0])
```

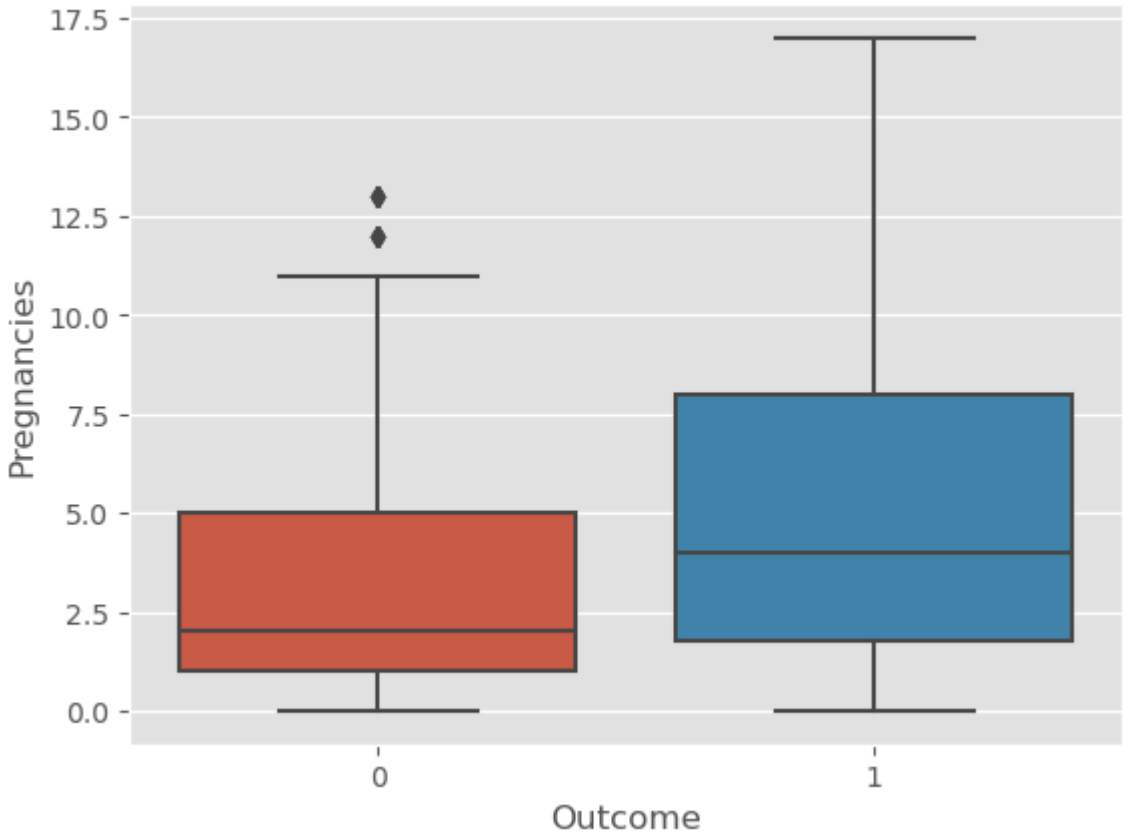
```
Total :  5
```

In [19]:
```
print(diabetes[diabetes.Glucose == 0].groupby('Outcome')['Age'].count())
```

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

In [20]:
```
sns.boxplot(y='Pregnancies', x='Outcome', data=diabetes)
```

Out[20]: <Axes: xlabel='Outcome', ylabel='Pregnancies'>

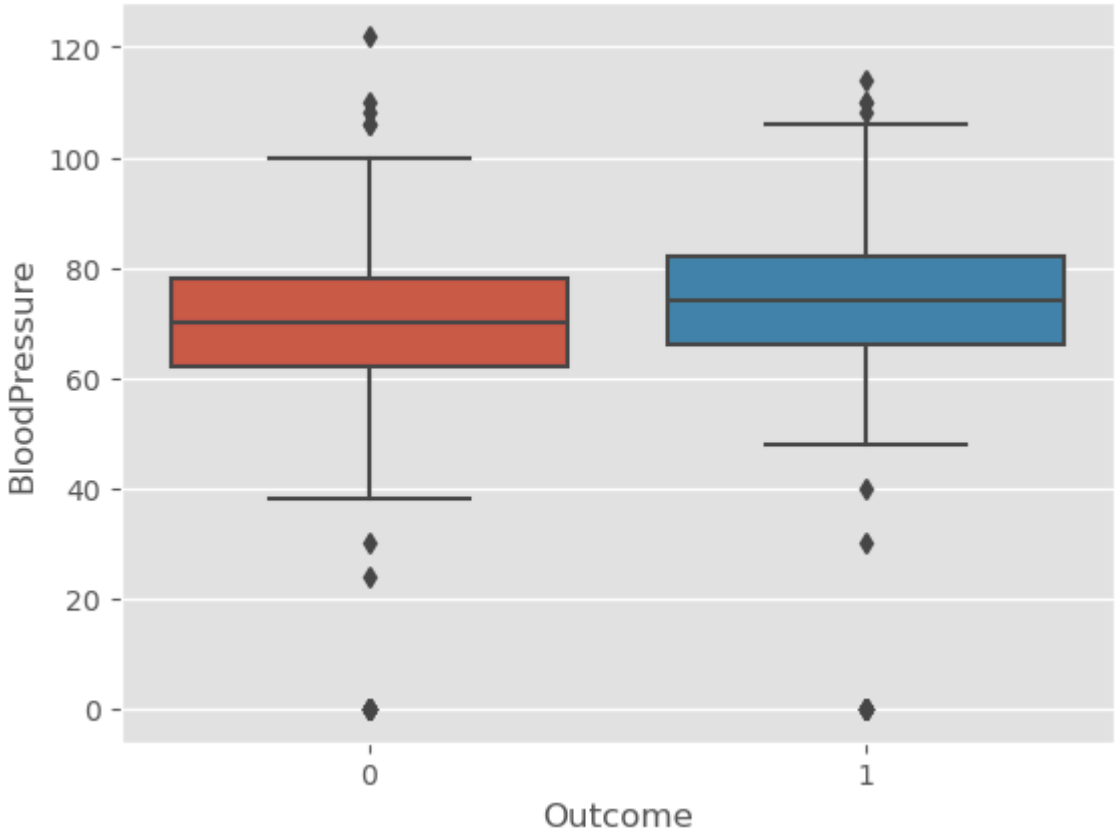


In [21]:
```
print("Total : ", diabetes[diabetes.Pregnancies == 0].shape[0])
```

```
Total :  111
```

In [22]:
```
sns.boxplot(y='BloodPressure', x='Outcome', data=diabetes)
```

Out[22]:
```
<Axes: xlabel='Outcome', ylabel='BloodPressure'>
```



In [23]:
```
print("Total : ", diabetes[diabetes.BloodPressure == 0].shape[0])
```

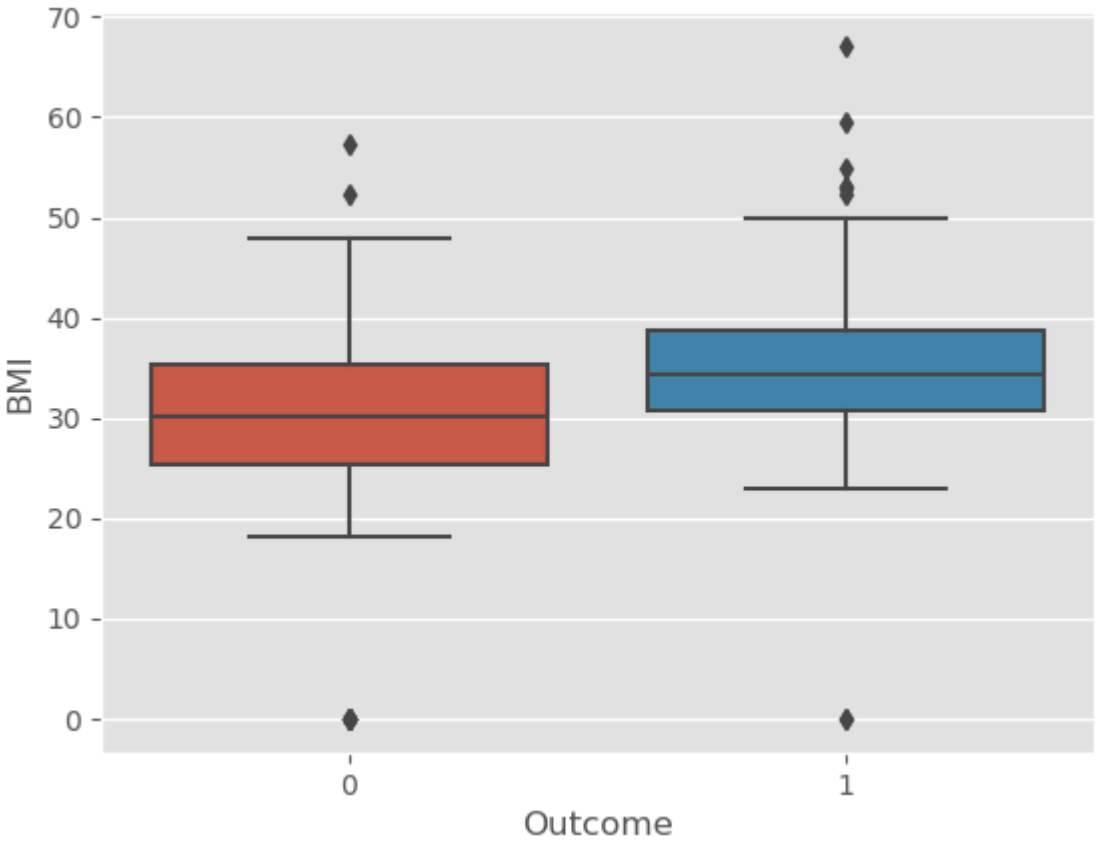
```
Total :  35
```

In [24]:
```
print(diabetes[diabetes.BloodPressure == 0].groupby('Outcome')['Age'].count(
```

```
Outcome
0    19
1    16
Name: Age, dtype: int64
```

In [25]:
```
sns.boxplot(y='BMI', x='Outcome', data=diabetes)
```

Out[25]: <Axes: xlabel='Outcome', ylabel='BMI'>



BMI: Tidak boleh 0 atau mendekati nol kecuali orang tersebut benar-benar kekurangan berat badan yang dapat mengancam nyawa.

In [26]:
```
print("Total : ", diabetes[diabetes.BMI == 0].shape[0])
```

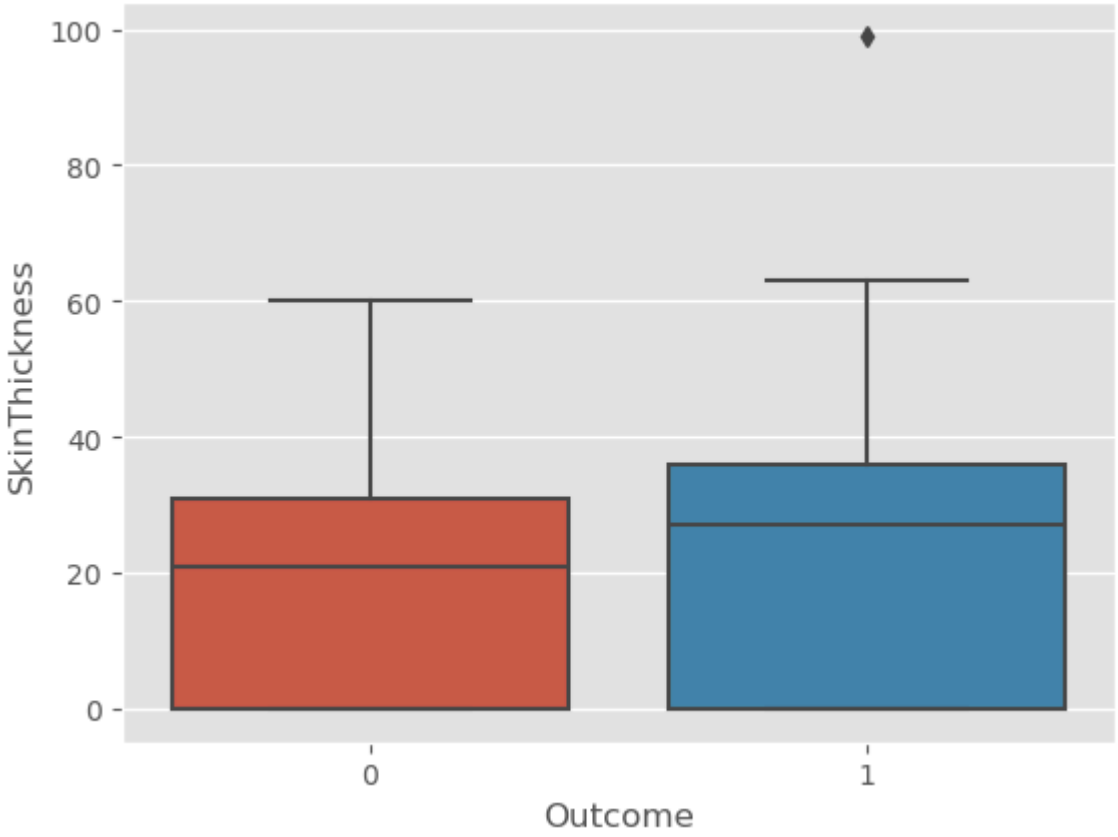
```
Total :  11
```

In [27]:
```
print(diabetes[diabetes.BMI == 0].groupby('Outcome')['Age'].count())
```

```
Outcome
0    9
1    2
Name: Age, dtype: int64
```

In [28]:
```
sns.boxplot(y='SkinThickness', x='Outcome', data=diabetes)
```

Out[28]:
```
<Axes: xlabel='Outcome', ylabel='SkinThickness'>
```



Ketebalan Lipatan Kulit: Untuk orang normal, ketebalan lipatan kulit tidak boleh kurang dari 10 mm lebih baik lagi nol. Jumlah total yang nilainya 0: 227.

In [29]:
```
print("Total : ", diabetes[diabetes.SkinThickness == 0].shape[0])
```

```
Total :  227
```

In [30]:
```
print(diabetes[diabetes.SkinThickness == 0].groupby('Outcome')['Age'].count(
```

```
Outcome
0    139
1     88
Name: Age, dtype: int64
```

> *__Berikut beberapa cara menangani nilai data yang tidak valid__* :

- Abaikan/hapus kasus-kasus ini: Hal ini sebenarnya tidak mungkin dilakukan dalam banyak kasus karena hal itu berarti kehilangan informasi berharga. Dan dalam hal ini kolom “ketebalan kulit” dan “insulin” berarti memiliki banyak poin yang tidak valid. Tapi ini mungkin berhasil untuk titik data “BMI”, “glukosa”, dan “tekanan darah”.
- Masukkan nilai rata-rata/rata-rata: Ini mungkin berhasil untuk beberapa kumpulan data, namun dalam kasus kita, memasukkan nilai rata-rata ke kolom tekanan darah akan mengirimkan sinyal yang salah ke model.
- Hindari penggunaan fitur: Dimungkinkan untuk tidak menggunakan fitur dengan banyak nilai yang tidak valid untuk modelnya. Ini mungkin berhasil untuk “ketebalan kulit” tetapi sulit untuk memprediksinya.

- Pada akhir proses pembersihan data, kami sampai pada kesimpulan bahwa kumpulan data yang diberikan tidak lengkap. Karena ini adalah demonstrasi untuk pembelajaran mesin, kami akan melanjutkan data yang diberikan dengan sedikit penyesuaian.

Kami akan menghapus baris yang “Tekanan Darah”, “BMI” dan “Glukosa” adalah nol.

In [31]:
```
# Modifikasi dataset
diabetes_mod = diabetes[(diabetes.BloodPressure != 0) & (diabetes.BMI != 0)
```

In [32]:
```
diabetes_mod.shape
```

Out[32]:
```
(724, 9)
```

2.5 Phase 3 — Feature Engineering

Proses mengubah data yang dikumpulkan menjadi fitur yang lebih mewakili masalah yang kami coba selesaikan pada model, untuk meningkatkan performa dan akurasinya.

In the data set, we have the following features.

> ‘Pregnancies’, ‘Glucose’, ‘Blood Pressure’, ‘Skin Thickness’, ‘Insulin’, ‘BMI’, ‘Diabetes Pedigree Function’, ‘Age’

we can say that the ‘Skin Thickness’ is not an indicator of diabetes. But we can’t deny the fact that it is unusable at this point.

Therefore we will use all the features available. We separate the data set into features and the response that we are going to predict.

- We will assign the `features to the X variable` and the `response to the y variable`.

In [33]:
```
# Create feature variable
feature_names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
```

In [34]:
```
# Separate the dataset
X = diabetes_mod[feature_names]
y = diabetes_mod.Outcome
```

Secara umum feature engineering ditunjukkan sebelum pemilihan model. Akan tetapi , pada analysis ini kita akan melakukan pendekatan berbeda , yaitu ; mencermati semua faktor dalam data dan medisusikan seberapa penting faktor faktor tersebut dalam model.

2.6 Phase 4 — Model Selection

Pemilihan model atau algoritma merupakan hal paling menarik dalam tahapan machine leraning ini , bahkan menjadi jantungnya proses pembangunan machine learning itu sendiri.

Langkah awal , kami akan menghitung “Akurasi Klasifikasi (Uji Akurasi)” dari sekumpulan model klasifikasi tertentu dengan parameter defaultnya untuk menentukan model mana yang berkinerja lebih baik dengan kumpulan data diabetes.

Kami akan mengimpor perpustakaan yang diperlukan untuk notebook. Kami mengimpor 7 pengklasifikasi yaitu `K–Nearest Neighbors` , `Support Vector Classifier` , `Logistic Regression` , `Gaussian Naive Bayes` , `Random Forest` , dan `Gradient Boost` untuk menjadi diperbandingkan performance pengklasifikasi terbaik yng di hasilkan.

In [35]:
```
# Import library model selection

from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score
```

Kita akan menginisiasi model model klasifikasi dengan parameter yang sudah bagku(default) dan menambahkan parameter tersebut dalam daftar model.

In [36]:
```
# Initial model selection process
models = []

models.append(('KNN', KNeighborsClassifier()))
models.append(('SVC', SVC(gamma='scale')))
models.append(('LR', LogisticRegression(solver='lbfgs', max_iter=4000)))
models.append(('DT', DecisionTreeClassifier()))
models.append(('GNB', GaussianNB()))
models.append(('RF', RandomForestClassifier(n_estimators=100)))
models.append(('GB', GradientBoostingClassifier()))
```

2.7 Evaluation Methods

- It is a general practice to avoid training and testing on the same data.
- The reasons are that the goal of the model is to predict out-of-sample data, and the model could be overly complex leading to overfitting.
- To avoid the aforementioned problems, there are two precautions.

1. Train/Test Split
2. K-Fold Cross-Validation

We will import `“train_test_split”` for train/test split and `“cross_val_score”` for k-fold cross-validation. `“accuracy_score”` is to evaluate the accuracy of the model in the train/test split method.

In [37]:
```
# Mengimport fungsi fungsi yang di perlukan program

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
```

Kami akan melakukan metode yang disebutkan untuk menemukan model dasar dengan kinerja terbaik.

2.8 Train/Test Split

Metode ini membagi kumpulan data menjadi dua bagian: kumpulan pelatihan dan kumpulan pengujian . Set pelatihan digunakan untuk melatih model. Dan set pengujian digunakan untuk menguji model, dan mengevaluasi keakuratannya.

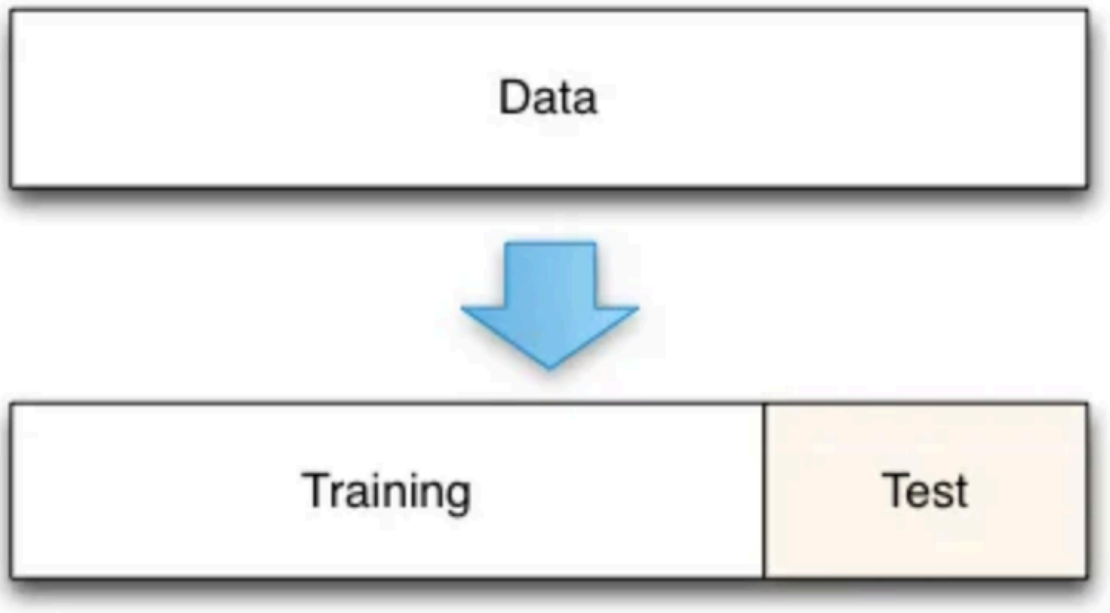


Fig — Train/Test Split

2.9 Train/Test Split with Scikit Learn :

Selanjutnya, kita dapat membagi fitur dan respons menjadi bagian pelatihan dan pengujian. Kami membuat stratifikasi (suatu proses di mana setiap kelas respons harus diwakili dengan proporsi yang sama di setiap porsi) sampel.

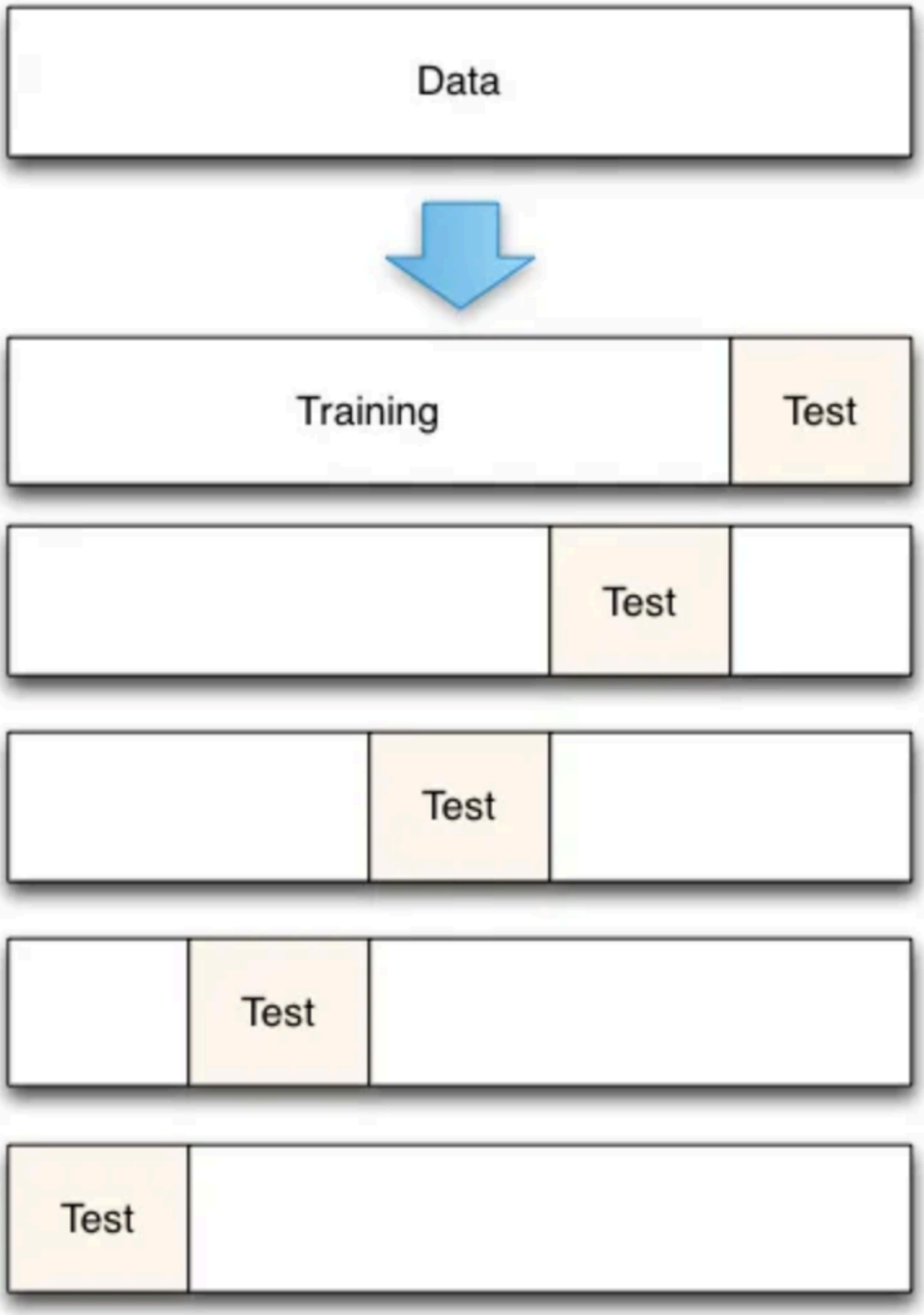
In [38]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = diabete
```

In [39]:
```
# Selanjutnya kita akan memasangkan data terhadap setiap model dan menghitun

names = []
scores = []

for name, model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    scores.append(accuracy_score(y_test, y_pred))
    names.append(name)

tr_split = pd.DataFrame({'Name': names, 'Score': scores})
print(tr_split)
```

```
  Name     Score
0  KNN  0.729282
1  SVC  0.740331
2   LR  0.779006
3   DT  0.729282
4  GNB  0.734807
5   RF  0.756906
6   GB  0.773481
```

2.10 K-Fold Cross-Validation

Metode ini membagi kumpulan data menjadi **K partisi yang sama** (“lipatan”), kemudian menggunakan 1 lipatan sebagai **set pengujian** dan gabungan lipatan lainnya sebagai **set pelatihan** . Kemudian model diuji keakuratannya. Prosesnya akan mengikuti langkah di atas sebanyak K kali, menggunakan lipatan berbeda sebagai set pengujian setiap kali. Akurasi pengujian rata-rata dari proses adalah akurasi pengujian.



Fig— 5-Fold cross validation process

> Metode ini lebih disukai jika kemampuan komputasinya tidak terbatas. Kami akan menggunakan metode ini mulai sekarang.

2.11 Validasi Silang K-Fold dengan Scikit Learn :

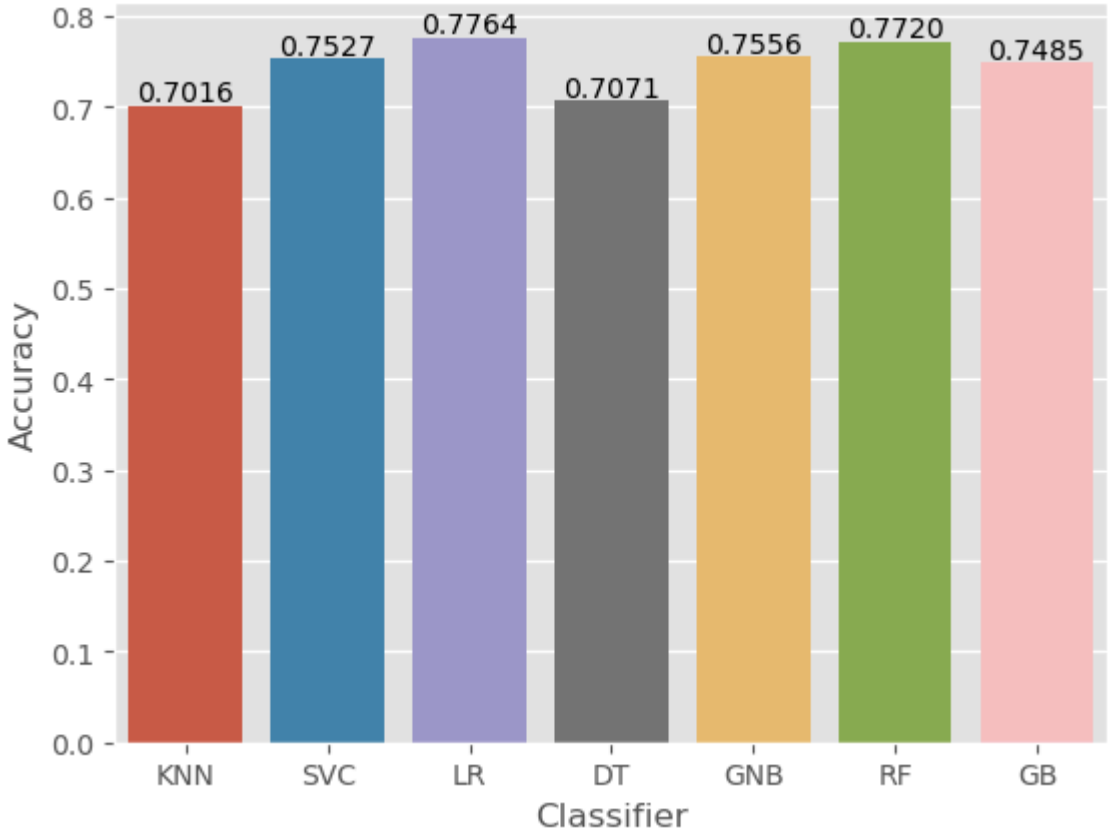
Kami akan melanjutkan dengan validasi silang K-Fold karena lebih akurat dan menggunakan data secara efisien. Kami akan melatih model menggunakan validasi silang 10 kali lipat dan menghitung akurasi rata-rata model. “cross_val_score” menyediakan pelatihannya sendiri dan antarmuka perhitungan akurasi.

In [40]:
```
from sklearn.model_selection import StratifiedKFold
strat_k_fold = StratifiedKFold(n_splits=10,shuffle=True, random_state=10)

names = []
scores = []

for name, model in models:

    score = cross_val_score(model, X, y, cv=strat_k_fold, scoring='accuracy'
    names.append(name)
    scores.append(score)

kf_cross_val = pd.DataFrame({'Name': names, 'Score': scores})
print(kf_cross_val)
```

```
  Name     Score
0  KNN  0.701636
1  SVC  0.752721
2   LR  0.776351
3   DT  0.707059
4  GNB  0.755613
5   RF  0.771994
6   GB  0.748497
```

> We can plot the accuracy scores using seaborn

In [41]:
```
axis = sns.barplot(x = 'Name', y = 'Score', data = kf_cross_val)
axis.set(xlabel='Classifier', ylabel='Accuracy')

for p in axis.patches:
    height = p.get_height()
    axis.text(p.get_x() + p.get_width()/2, height + 0.005, '{:1.4f}'.format(

plt.show()
```



- Kita dapat melihat Regresi Logistik, Gaussian Naive Bayes, Random Forest, dan Gradient Boosting memiliki kinerja lebih baik daripada yang lain.
- Dari tingkat dasar kita dapat mengamati bahwa `Logistik Regression (LR)` berkinerja lebih baik daripada algoritma lainnya.
- Ini akan dipilih sebagai kandidat utama untuk fase berikutnya.

> Setelah pemilihan model, kami dapat mengidentifikasi bahwa Regresi Logistik memiliki kinerja lebih baik dibandingkan model klasifikasi terpilih lainnya. Dalam artikel ini kita akan membahas tahapan selanjutnya dari alur kerja pembelajaran mesin, rekayasa fitur lanjutan, dan penyetelan parameter hiper.

3. Phase 5 — Feature Engineering (Revisited)

Seperti disebutkan dalam Fase 3, setelah pemilihan model, Rekayasa Fitur harus dibahas lebih lanjut. Oleh karena itu kami akan menganalisis model yang dipilih yaitu Regresi Logistik, dan bagaimana pengaruh pentingnya fitur terhadapnya.

Scikit Learn menyediakan metode berguna yang dapat digunakan untuk melakukan pemilihan fitur dan mengetahui pentingnya fitur yang memengaruhi model.

1. `Univariate Feature Selection` : Uji statistik dapat digunakan untuk memilih fitur-fitur yang memiliki hubungan paling kuat dengan variabel keluaran.
2. `Recursive Feature Elimination` : Penghapusan Fitur Rekursif (atau RFE) bekerja dengan menghapus atribut secara rekursif dan membangun model pada atribut yang tersisa. Ia menggunakan akurasi model untuk mengidentifikasi atribut mana (dan kombinasi atribut) yang berkontribusi paling besar dalam memprediksi atribut target.
3. `Principal Component Analysis` : Analisis Komponen Utama (atau PCA) menggunakan aljabar linier untuk mengubah kumpulan data menjadi bentuk terkompresi. Umumnya hal ini disebut dengan teknik reduksi data. Properti PCA adalah Anda dapat memilih jumlah dimensi atau komponen utama dalam hasil transformasi.
4. `Feature Importance` : Pohon keputusan yang dikantongi seperti Random Forest dan Extra Trees dapat digunakan untuk memperkirakan pentingnya fitur.

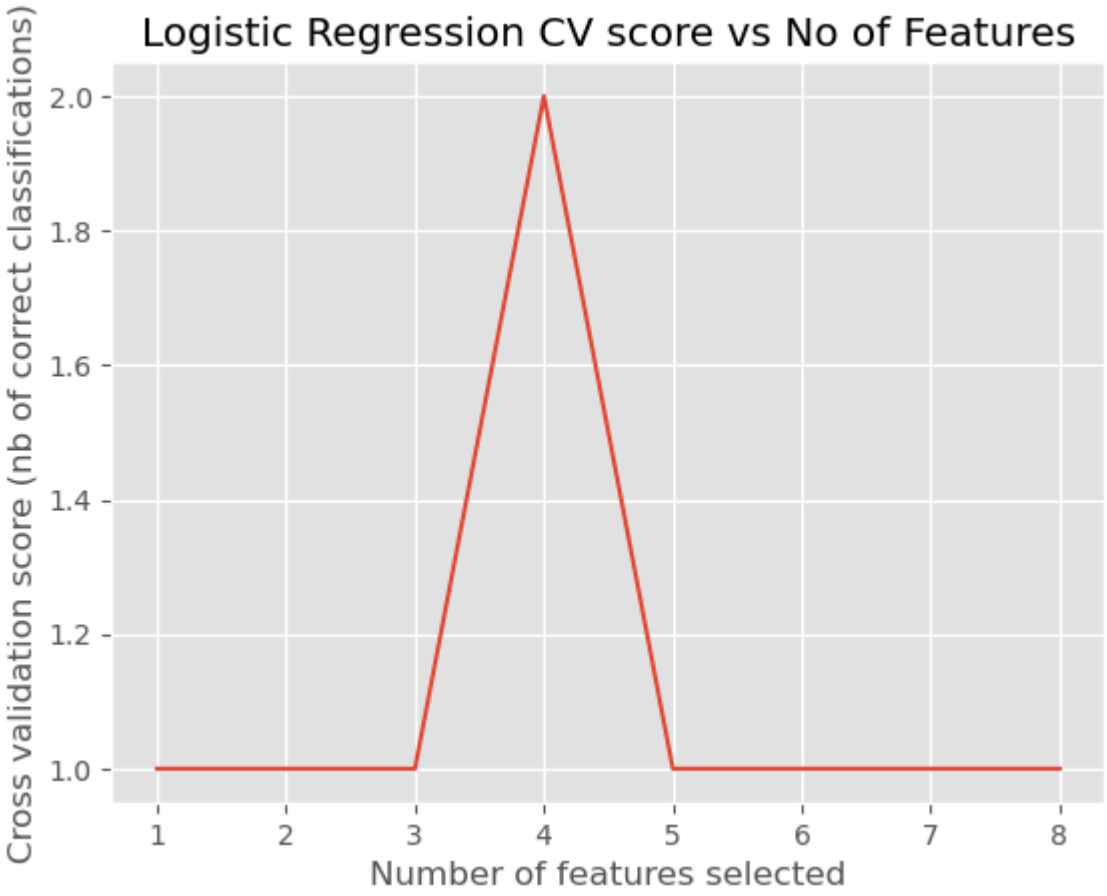
Dalam analysis ini kita akan menggunakan Penghapusan Fitur Rekursif sebagai metode pemilihan fitur.

Pertama kita mengimpor `RFECV`, yang dilengkapi dengan fitur validasi silang bawaan. Sama seperti model pengklasifikasi, RFECV memiliki metode `fit()` yang menerima fitur dan respons/target.

3.1 Logistic Regression — Feature Selection

In [42]:
```
from sklearn.feature_selection import RFECV
```

In [43]:
```
logreg_model = LogisticRegression(solver='lbfgs', max_iter=4000)

rfecv = RFECV(estimator=logreg_model, step=1, cv=strat_k_fold, scoring='accu
rfecv.fit(X, y)

plt.figure()
plt.title('Logistic Regression CV score vs No of Features')
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (nb of correct classifications)")
plt.plot(range(1, len(rfecv.ranking_) + 1), rfecv.ranking_)
plt.show()
```



Dengan melihat plotnya kita dapat melihat bahwa `memasukkan 4 fitur` ke dalam model memberikan `skor akurasi terbaik`. RFECV memaparkan support_ yang merupakan atribut lain untuk mengetahui fitur yang paling berkontribusi dalam prediksi. Untuk mengetahui fitur apa saja yang dipilih kita dapat menggunakan kode berikut.

In [44]:
```
feature_importance = list(zip(feature_names, rfecv.support_))

new_features = []

for key,value in enumerate(feature_importance):
    if(value[1]) == True:
        new_features.append(value[0])

print(new_features)
```

```
['Pregnancies', 'Glucose', 'BloodPressure', 'Insulin', 'BMI', 'DiabetesPedig
reeFunction', 'Age']
```

Kita dapat melihat bahwa fitur yang diberikan adalah yang paling cocok untuk memprediksi kelas respons. Kita dapat melakukan perbandingan model dengan fitur asli dan fitur RFECV yang dipilih untuk melihat apakah ada peningkatan pada skor akurasi.

In [45]:
```
# Calculate accuracy scores

X_new = diabetes_mod[new_features]
```

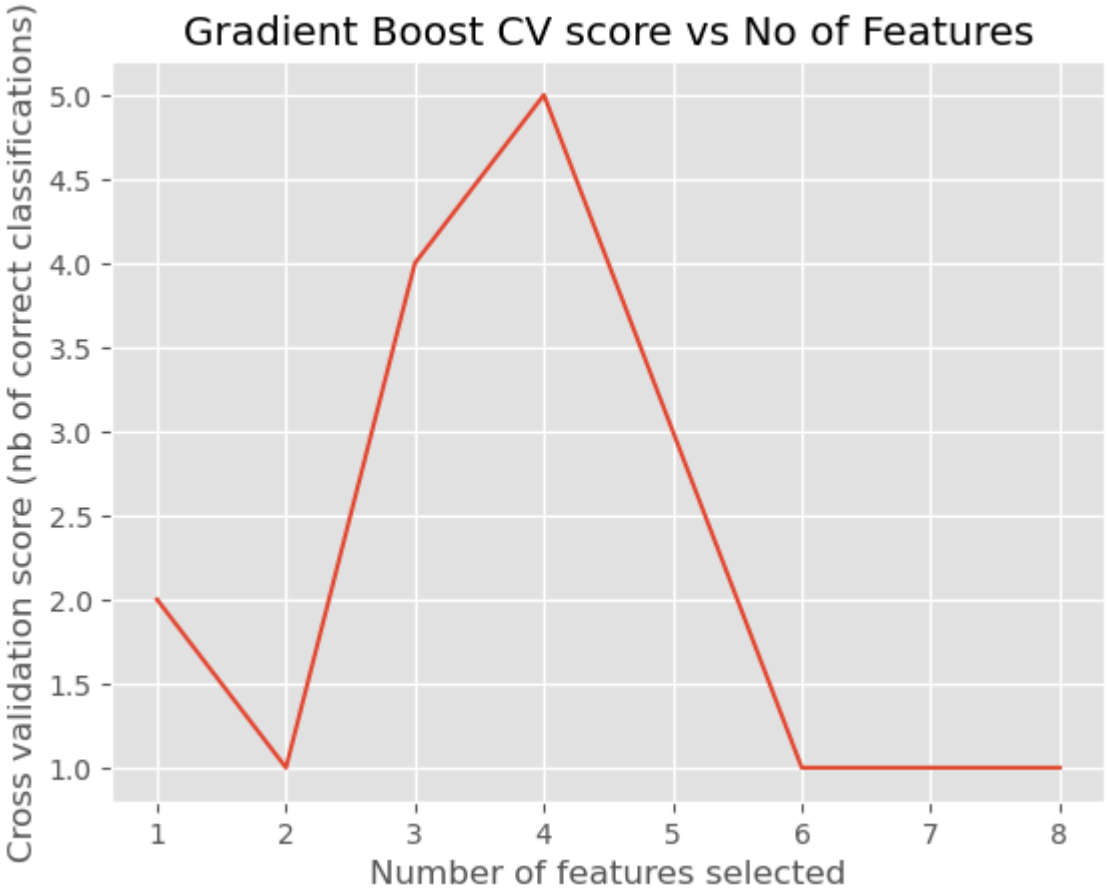
```
initial_score = cross_val_score(logreg_model, X, y, cv=strat_k_fold, scoring
print("Initial accuracy : {} ".format(initial_score))

fe_score = cross_val_score(logreg_model, X_new, y, cv=strat_k_fold, scoring=
print("Accuracy after Feature Selection : {} ".format(fe_score))
```

```
Initial accuracy : 0.7763508371385084
Accuracy after Feature Selection : 0.7763508371385084
```

Dengan mengamati akurasi, ada sedikit peningkatan akurasi setelah memasukkan fitur yang dipilih ke model.

3.1.1 Gradien Boost

In [46]:
```
gb_model = GradientBoostingClassifier()

gb_rfecv = RFECV(estimator=gb_model, step=1, cv=strat_k_fold, scoring='accur
gb_rfecv.fit(X, y)

plt.figure()
plt.title('Gradient Boost CV score vs No of Features')
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (nb of correct classifications)")
plt.plot(range(1, len(gb_rfecv.ranking_) + 1), gb_rfecv.ranking_)
plt.show()
```



In [47]:
```
feature_importance = list(zip(feature_names, gb_rfecv.support_))

new_features = []

for key,value in enumerate(feature_importance):
    if(value[1]) == True:
        new_features.append(value[0])

print(new_features)
```

```
['Glucose', 'BMI', 'DiabetesPedigreeFunction', 'Age']
```

In [48]:
```
X_new_gb = diabetes_mod[new_features]

initial_score = cross_val_score(gb_model, X, y, cv=strat_k_fold, scoring='ac
print("Initial accuracy : {} ".format(initial_score))

fe_score = cross_val_score(gb_model, X_new_gb, y, cv=strat_k_fold, scoring='
print("Accuracy after Feature Selection : {} ".format(fe_score))
```

```
Initial accuracy : 0.7484969558599696
Accuracy after Feature Selection : 0.7583333333333333
```

> Namun Regresi Logistik lebih akurat daripada Peningkatan Gradien. Jadi kita akan menggunakan Regresi Logistik untuk tahap penyetelan parameter.

3.2 Phase 6 - Model Parameter Tuning

Scikit Learn menyediakan model dengan parameter default yang masuk akal yang memberikan skor akurasi yang layak. Ini juga memberi pengguna opsi untuk mengubah parameter untuk lebih meningkatkan akurasi.

Dari pengklasifikasi, kami akan memilih Regresi Logistik untuk penyesuaian, di mana kami mengubah parameter model agar dapat meningkatkan keakuratan model untuk kumpulan data tertentu.

Daripada harus mencari parameter optimal secara manual, kita dapat dengan mudah melakukan pencarian menyeluruh menggunakan GridSearchCV , yang melakukan “pencarian menyeluruh atas nilai parameter tertentu untuk suatu estimator” .

> `Penting !!!` : Saat menggunakan GridSearchCV, ada beberapa model yang memiliki parameter yang tidak berfungsi satu sama lain. Karena GridSearchCV menggunakan kombinasi semua parameter yang diberikan, jika dua parameter tidak berfungsi satu sama lain, kita tidak akan dapat menjalankan GridSearchCV.

> Jika hal tersebut terjadi, daftar grid parameter dapat diberikan untuk mengatasi masalah yang ada. **Disarankan agar Anda membaca dokumen kelas yang ingin Anda sesuaikan, untuk mengetahui cara kerja parameter satu sama lain**.

> Dokumen kelas Regresi Logistik dapat ditemukan di sini .

In [49]:
```
from sklearn.model_selection import GridSearchCV
```

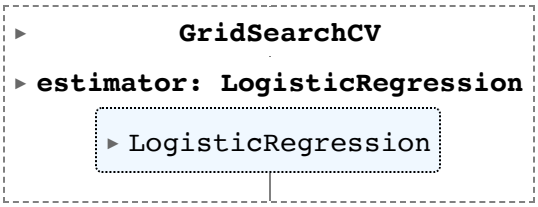
In [50]:
```
# Specify parameters
c_values = list(np.arange(1, 10))

param_grid = [
    {'C': c_values, 'penalty': ['l1'], 'solver' : ['liblinear'], 'multi_clas
    {'C': c_values, 'penalty': ['l2'], 'solver' : ['liblinear', 'newton-cg',
]
```

- Kemudian kami memasukkan data ke GridSearchCV, yang melakukan validasi K-fold cross pada data untuk kombinasi parameter tertentu. Mungkin perlu waktu agak lama untuk menyelesaikannya.

%pip install scikit-optimize==0.8.1 %pip install scikit-learn==0.22.2

In [51]:
```
grid = GridSearchCV(LogisticRegression(), param_grid, cv=strat_k_fold, scori
grid.fit(X_new, y)
```

Out[51]:



- Setelah pelatihan dan scoring selesai, GridSearchCV memberikan beberapa atribut yang berguna untuk mencari parameter dan estimator terbaik.

In [52]:
```
print(grid.best_params_)
print(grid.best_estimator_)
```

```
{'C': 9, 'multi_class': 'ovr', 'penalty': 'l1', 'solver': 'liblinear'}
LogisticRegression(C=9, multi_class='ovr', penalty='l1', solver='liblinear')
```

- Kita dapat mengamati bahwa parameter hiper terbaik adalah sebagai berikut.
- Kita dapat memasukkan parameter terbaik ke model Regresi Logistik dan mengamati apakah akurasinya meningkat.

In [53]:
```
logreg_new = LogisticRegression(C=1, multi_class='ovr', penalty='l2', solver
```

In [54]:
```
initial_score = cross_val_score(logreg_new, X_new, y, cv=strat_k_fold, scori
print("Final accuracy : {} ".format(initial_score))
```

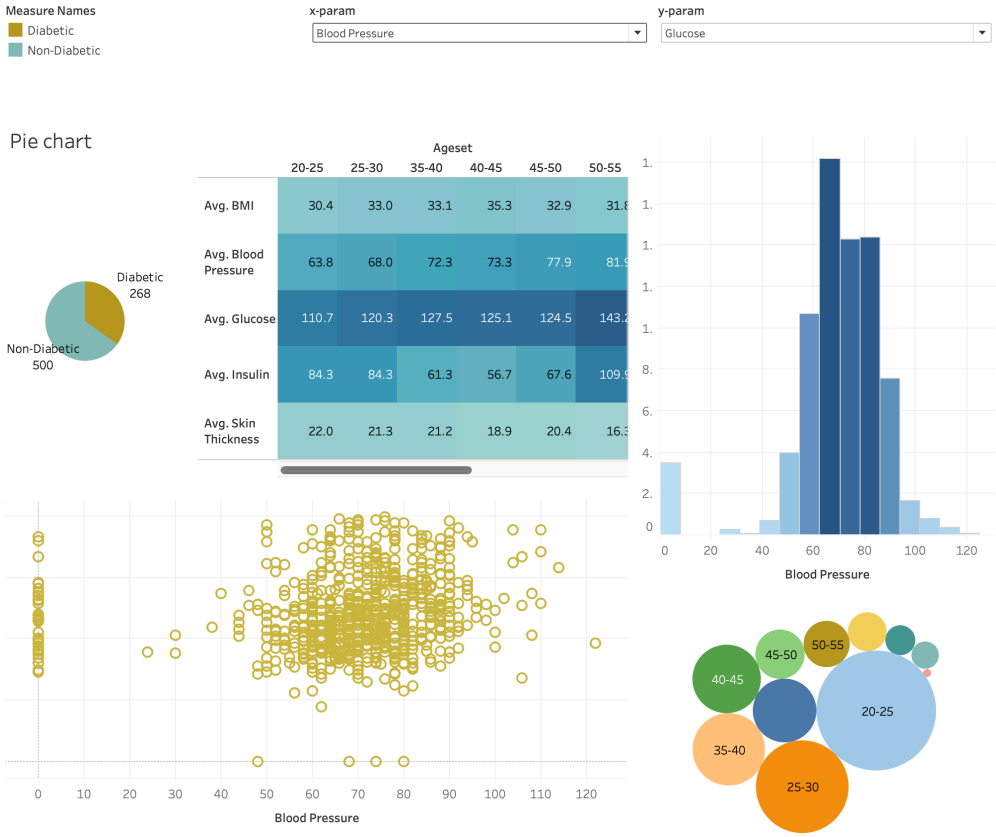
```
Final accuracy : 0.765220700152207
```

- Kita dapat menyimpulkan bahwa penyetelan parameter hiper tidak meningkatkan akurasinya. Mungkin parameter hiper yang kami pilih tidak bersifat indikatif. Namun Anda dipersilakan untuk mencoba menambahkan lebih banyak kombinasi parameter.

> Aspek terpenting adalah prosedur melakukan penyetelan parameter hiper, bukan hasil itu sendiri. Dalam kebanyakan kasus, penyetelan parameter hiper meningkatkan akurasi.

Dashboard Diabetes Prediction

Dibuat secara terpisah pada tableau public



In [ ]:
```
!jupyter nbconvert --to webpdf --allow-chromium-download NYCTreeCensus.ipynb
```