LAPORAN PERBANDINGAN METODE SVM, DECISION TREE, DAN NAÏVE BAYES DALAM PREDIKSI STROKE

Oleh:

Lailatul Ahmada : 5003201054
Siti Asmaul Kusnah : 5003201036

Departemen Statistika

Fakultas Sains dan Analitika Data

Institut Teknologi Sepuluh Nopember Surabaya

2023

A. SUMBER DATA

Analisis menggunakan stroke dataset yang bersumber dari website Kaggle dengan link https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset. Data diakses pada tanggal 8 Mei 2023. Dataset terdiri dari 12 atribut dengan 3000 observasi. Analisis dilakukan dari preprocessing hingga analisis dengan menggunakan metode *Support Vector Machine, Decision Tree*, dan *Naïve Bayes*. Berikut adalah atribut pada stroke dataset.

Atribut	Type Data	Keterangan
id	Kontinue	Unique identifier
gender	Kategori	Male, Female, Other
age	Kontinue	Umur pasien
hypertension	Kategori	0: pasien tidak mempunyai hipertensi 1: pasien mempunyai hipertensi
heart_disease	Kategori	0: pasien tidak mempunyai penyakit jantung 1: pasien mempunyai penyakit jantung
ever_married	Kategori	No, Yes
work_type	Kategori	children, Govt_jov, Never_worked, Private, Self-employed
Residence_type	Kategori	Rural, Urban
avg_glucose_level	Kontinue	Rata-rata kadar glukosa dalam darah
bmi	Kontonue	Indeks masa tubuh
smoking_status	Kategori	Formely smoked, never smoked, smokes, Unknown
stroke	Kategori	0: pasien tidak mempunyai stroke 1: pasien mempunyai stroke

B. PREPROCESSING

➢ Meng-import data

```
df=pd.read_csv("D:/smt6/datmin/healthcare-dataset-stroke-data.csv")
df.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked

Info data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 3000 non-null   object
 1   gender             3000 non-null   object
 2   age                3000 non-null   int32
 3   hypertension       3000 non-null   object
 4   heart_disease      3000 non-null   object
 5   ever_married       3000 non-null   object
 6   work_type          3000 non-null   object
 7   Residence_type     3000 non-null   object
 8   avg_glucose_level  3000 non-null   float64
 9   bmi                3000 non-null   float64
 10  smoking_status     3000 non-null   object
 11  stroke             3000 non-null   object
dtypes: float64(2), int32(1), object(9)
memory usage: 269.7+ KB
```

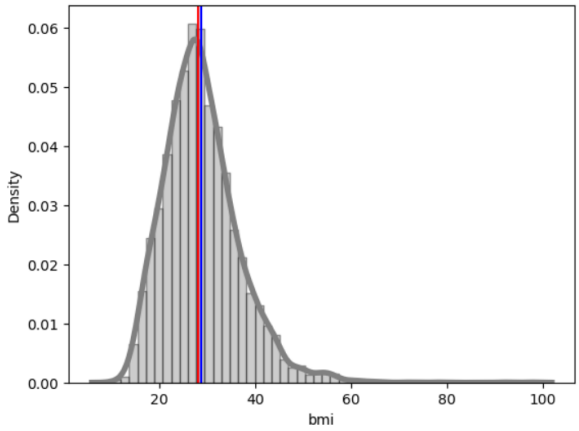
- **Cek missing value**

  Terdapat 139 data *missing* pada variabel bmi. Data missing tersebut kemudian diatasi dengan mengganti data *missing* dengan median data karena median bersifat robust.

```
df.isnull().sum()
```

```
id                   0
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
Residence_type       0
avg_glucose_level    0
bmi                139
smoking_status       0
stroke               0
dtype: int64
```

Density plot of bmi
Red(median) blue (mean)



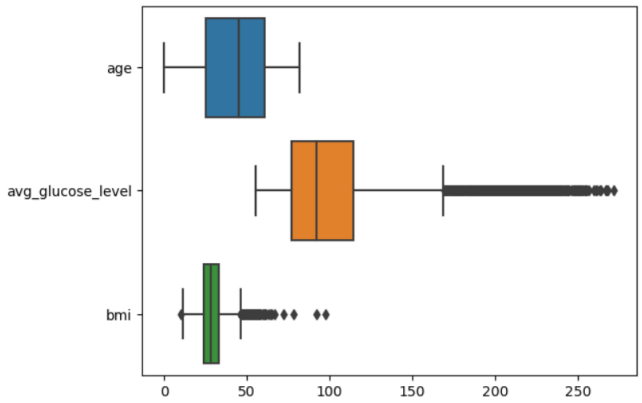
Mengatasi *missing value* dengan mengganti data missing dengan median data.

```
: df['bmi']=df['bmi'].fillna(df['bmi'].median())

: df.isnull().sum()

: id                   0
  gender               0
  age                  0
  hypertension         0
  heart_disease        0
  ever_married         0
  work_type            0
  Residence_type       0
  avg_glucose_level    0
  bmi                  0
  smoking_status       0
  stroke               0
  dtype: int64
```

- **Cek outlier untuk data numerik (variabel age, avg_glucose_level, dan bmi)**
  Outlier tidak diatasi karena kemungkinan outlier merupakan data yang penting sehingga tidak perlu dilakukan penghapusan outlier.



- **Encoding data pada variabel kategori**
  Encoding data dilakukan untuk mengubah data menjadi numerik sehingga memudahkan analisis:
  - Gender =male:0, female:1, other:2
  - Ever married = yes:0 & no:1
  - Work_type = private:0, self-employed:1, children:2, 'Govt_job':3,'Never_worked':4
  - Residence_type = 'Urban':0,'Rural':1
  - Smoking_statis= 'never smoked':0,'Unknown':1,'formerly smoked':2,'smokes':3

id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke	gender	ever_married	work_type	Residence_type	smoking_status
)046	67	0	1	228.69	36.6	1	1	1	2	1	1
1676	61	0	0	202.21	28.1	1	0	1	3	0	2
1112	80	0	1	105.92	32.5	1	1	1	2	0	2
)182	49	0	0	171.23	34.4	1	0	1	2	1	3
665	79	1	0	174.12	24.0	1	0	1	3	0	2
...	...	...	...	...	...	...	...	...	...	...	...
234	80	1	0	83.75	28.1	0	0	1	2	1	2
873	81	0	0	125.20	40.0	0	0	1	3	1	2
)723	35	0	0	82.99	30.6	0	0	1	3	0	2
544	51	0	0	166.29	25.6	0	1	1	2	0	1
679	44	0	0	85.28	26.2	0	0	1	0	1	0

- **Transformasi data pada variabel numerik yang memiliki satuan yang berbeda untuk menyamakan skala data.**

Dengan metode min max

```
df['age']=(df['age']-df['age'].min())/(df['age'].max()-df['age'].min())
df['avg_glucose_level']=(df['avg_glucose_level']-df['avg_glucose_level'].min())/(df['avg_glucose_level'].max()-df['a
df['bmi']=(df['bmi']-df['bmi'].min())/(df['bmi'].max()-df['bmi'].min())
df
```

id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke	gender	ever_married	work_type	Residence_type	smok
046	0.817073	0	1	0.801265	0.301260	1	1	1	2	1	
676	0.743902	0	0	0.679023	0.203895	1	0	1	3	0	
1112	0.975610	0	1	0.234512	0.254296	1	1	1	2	0	
182	0.597561	0	0	0.536008	0.276060	1	0	1	2	1	
665	0.963415	1	0	0.549349	0.156930	1	0	1	3	0	
...	...	...	...	...	...	...	...	...	...	...	
234	0.975610	1	0	0.132167	0.203895	0	0	1	2	1	
873	0.987805	0	0	0.323516	0.340206	0	0	1	3	1	
723	0.426829	0	0	0.128658	0.232532	0	0	1	3	0	
544	0.621951	0	0	0.513203	0.175258	0	1	1	2	0	
679	0.536585	0	0	0.139230	0.182131	0	0	1	0	1	

```
df.dtypes
```

```
id                  object
age                float64
hypertension        object
heart_disease       object
Residence_type      object
avg_glucose_level  float64
bmi                float64
stroke              object
gender               int32
ever_married         int32
work_type            int32
smoking_status       int32
dtype: object
```

- **Feature Selection**

  Melakukan *feature selection* untuk memilih variabel yang paling optimal.

```
#Target: kategori
#Features: kategori
#Chi-Square
from sklearn.feature_selection import SelectKBest, f_classif, chi2
```

Chi-Square

```
feature1 = df[['hypertension','heart_disease','gender','ever_married','work_type','Residence_type','smoking_status']]
target1 = df['stroke']

#feature selection using chi2
bestfeatures = SelectKBest(score_func=chi2, k=3)
fit = bestfeatures.fit(feature1,target1)
#create df for scores
dfscores = pd.DataFrame(fit.scores_)
#create df for column names
dfcolumns = pd.DataFrame(feature1.columns)

#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
#naming the dataframe columns
featureScores.columns = ['Selected_columns','Score_chi2']
#print 5 best features
print(featureScores.nlargest(3,'Score_chi2'))
```

```
  Selected_columns  Score_chi2
0     hypertension   66.253320
1    heart_disease   65.341053
3     ever_married   18.613573
```

Metode Chi-Square digunakan untuk memilih target merupakan data kategori dan features merupakan data kategori. Berdasarkan output tersebut dipilih tiga variabel yang optimal yaitu hypertension, heart_disease, dan ever_married.

anova

```
y=df['stroke']
xan=df[['age','avg_glucose_level','bmi']]
```

Sedangkan untuk memilih feature dari jenis data kontinu atau numerik digunakan metode ANOVA.

```
target2= df['stroke']
feature2=df[['age','avg_glucose_level','bmi']]
feature2
```

	age	avg_glucose_level	bmi
0	0.817073	0.801173	0.301260
1	0.743902	0.678875	0.205040
2	0.975610	0.234159	0.254296
3	0.597561	0.535793	0.276060
4	0.963415	0.549141	0.156930
...	...	...	...
2995	0.951220	0.137678	0.180985
2996	0.426829	0.117957	0.211913

```
#from sklearn.feature_selection import SelectKBest
#from sklearn.feature_selection import f_classif

#feature selection using f_classif
fs = SelectKBest(score_func=f_classif, k=2)
fit = fs.fit(feature2,target2)

#create df for scores
dfscores1 = pd.DataFrame(fit.scores_)

#create df for column names
dfcolumns1 = pd.DataFrame(feature2.columns)

#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns1,dfscores1],axis=1)

#naming the dataframe columns
featureScores.columns = ['Selected_columns','Score_ANOVA']

#print 3 best features
print(featureScores.nlargest(2,'Score_ANOVA'))
#k bisa diubah2 sebanyak feature yang mau diambil
```

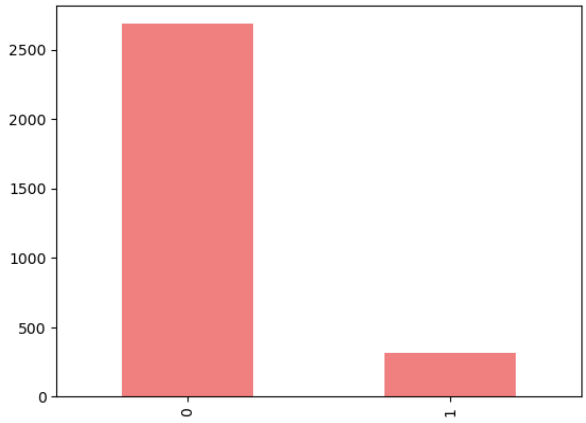
```
    Selected_columns  Score_ANOVA
0                age   316.413516
1  avg_glucose_level    77.117372
```

Berdasarkan output tersebut variabel yang terbaik adalah age dan avg_glucose_level karena memiliki score ANOVA tertinggi.

C. Visualisasi Data

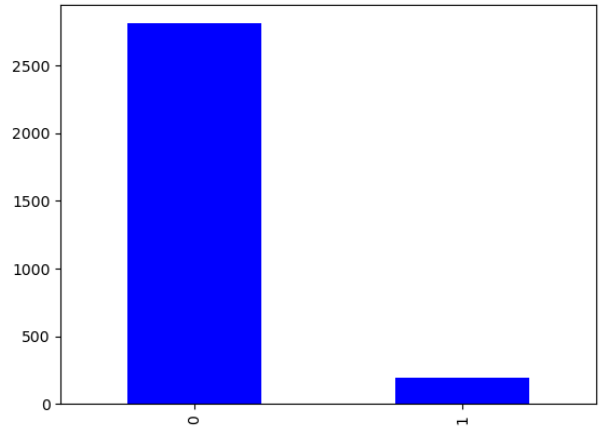
- **Data Kategori**

Variabel Hypertension



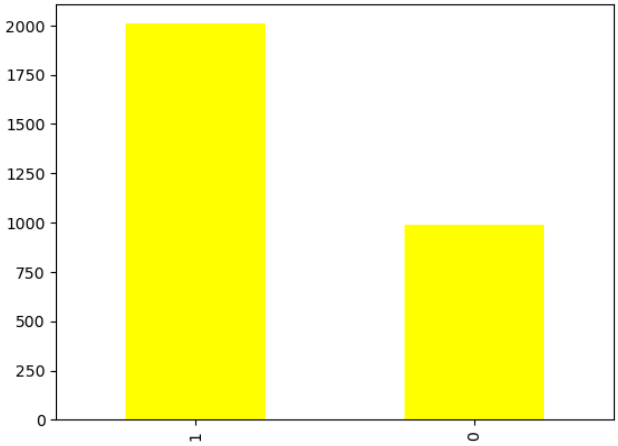
Berdasarkan bar chart variabel hypertension menunjukkan bahwa pasien yang tidak mempunyai hipertensi (kode 0) lebih banyak daripada pasien yang mempunyai hipertensi (kode 1).

Variabel Heart_disease



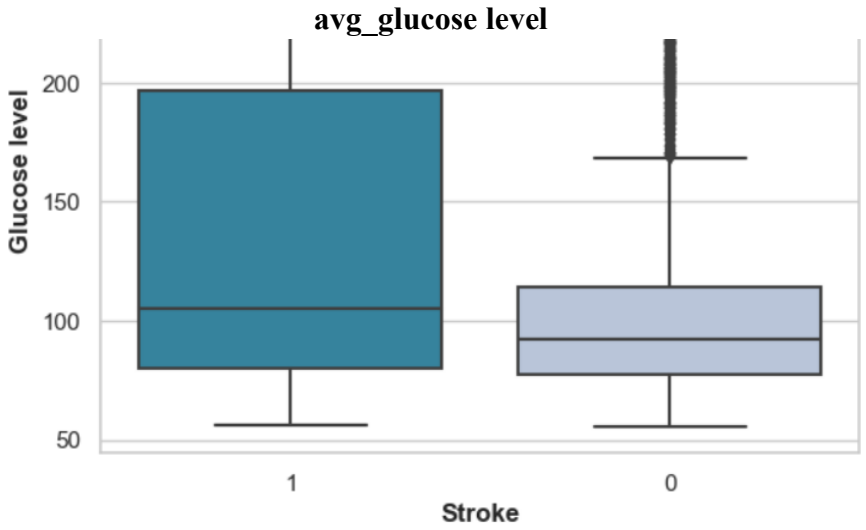
Berdasarkan bar chart variabel heart_disease menunjukkan bahwa pasien yang tidak memiliki penyakit jantung lebih tinggi (kode 0) daripada pasien yang mempunyai penyakit jantung (kode 1).

Variabel ever_married

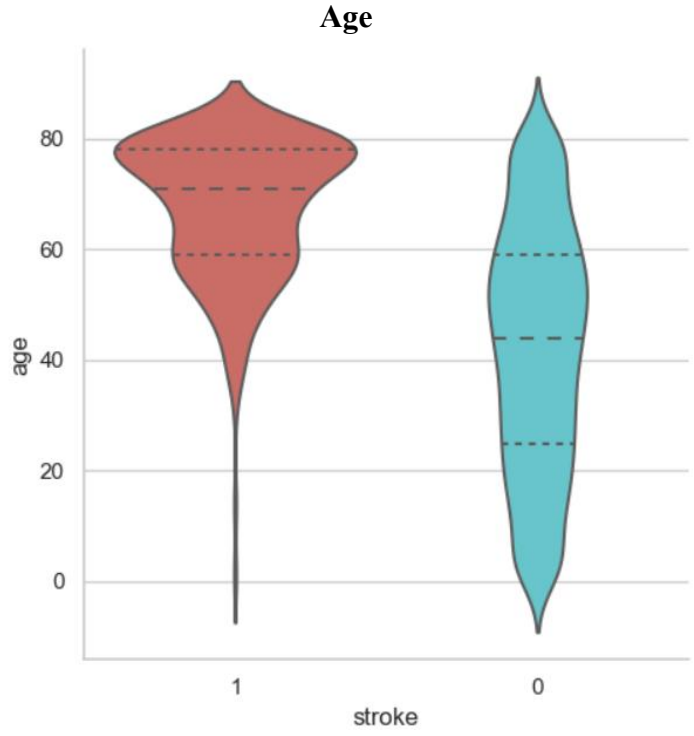


Berdasarkan bar chart variabel ever_married menunjukkan bahwa pasien yang pernah menikah lebih tinggi (kode 1) daripada pasien yang belum menikah (kode 0).

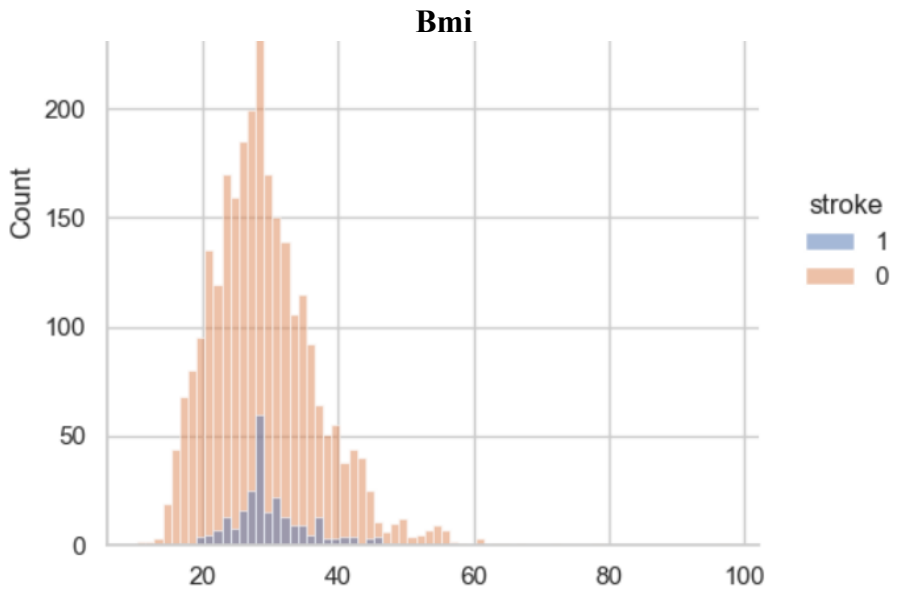
- **Data Kontinu**



Gambar diatas adalah boxplot nilai avg glucosa level di masing masing pasien yang mengidap stroke dan tidak mengidap stroke. Pada boxplot yang pertama dimana pasien mengidap stroke memiliki nilai median avg glucosa level kisaran 100-110 , dan tanpa memiliki outliers sama sekali hal itu menunjukkan bahwa pasiean yang mengidap stroke memiiki avg glucosa level yang tak jauh dari 80-200. Sedangkan pada boxplot sebalah kanan yaitu pasien tidak stroke memiliki nilai median avg glucosa level kisaran 80-90 dan memiliki outliers/ data pencilan yang bernilai diatas 200. Hal tersebut menunjukkan bahwa pasien yang memiliki avg glucosa level tinggi (diatas 200) tidak dapat langsung disimpulkan sebagai pasien yang mengidap stroke



Gambar diatas merupakan violin plot dengan sumbu x pasien stroke dan pasien tidak stroke sedangkan pada sumbu y adalah umur dari pasienya. Dapat kita lihat bahwa rata rata penyakit stroke didierita pasien kisaran umur40 ketas lebih detailnya banyak diderita oleh pasien berumur 80 katas. Sedangkan pada violin plot sebelah kanan dimana menggambarkan umur pasien yang tidak mengidap stroke yang hampir sama saja jumlahnya ditiap jenjang umur.



Gambar diatas merupakan histogram dari Body Mass Index untuk pasien pengidap stroke dan yang tidak pengidap stroke. Dapat kita lihat bahwa BMI rentang 20-40 baik untuk pasien pengidap stroke maupun yang tidak memilki jumlah tertinggi. Atau dapat kita katakan nilai modus BMI baik pada pasien stroke maupun tidak berada pada kisaran 20-40.

D. SVM

- **Mendefinisikan *features* dan *target.***

```
y=df['stroke']
x=df[['hypertension','heart_disease','ever_married','age','avg_glucose_level']]
```

```
y=y.values
x=x.values
```

Berdasarkan feature selection dengan metode chisquare dan ANOVA, kami memutuskan menggunakna 5 prediktor , yaitu 3 kategorik (hypertension, heart_disease, dan ever_married) dan 2 numerik (age dan avg_glucose_level)

- **Mengubah feature dan target kedalam bentuk array.**

```
y=y.values
x=x.values
```

```
y
```

```
array(['1', '1', '1', ..., '0', '0', '0'], dtype=object)
```

```
x
```

```
array([[0.8170731707317073, '0', '1', ..., 1, 2, 1],
       [0.7439024390243902, '0', '0', ..., 1, 3, 2],
       [0.975609756097561, '0', '1', ..., 1, 2, 2],
       ...,
       [0.4268292682926829, '0', '0', ..., 1, 3, 2],
       [0.6219512195121951, '0', '0', ..., 1, 2, 1],
       [0.5365853658536586, '0', '0', ..., 1, 0, 0]], dtype=object)
```

- **Menentukan parameter yang sesuai**

  Penentuan parameter dilakukan untuk menentukan metode apa yang terbaik dengan melihat nilai akurasi yang paling tinggi.

```
#Crossvalidation
cv=StratifiedKFold(n_splits=2, random_state=None) #n splits terserah ngesplit train testing
ss=svm.SVC(class_weight=None,random_state=2) #the weight is unknown
model=ss.fit(x,y)
```

```
parameter={
    'kernel':('linear', 'poly', 'rbf'),
    'C':[0.0001, 0.001, 0.01,0.1, 1, 10, 100, 1000, 10000],
    'gamma':[0.0001, 0.001, 0.01,0.1, 1, 10, 100, 1000, 10000],
}
grid=GridSearchCV(model, parameter, cv=cv, n_jobs=1, scoring='accuracy') #untuk mencari paraeter yang terbaik yang mana
grid
```

```
GridSearchCV(cv=StratifiedKFold(n_splits=2, random_state=None, shuffle=False),
             estimator=SVC(random_state=2), n_jobs=1,
             param_grid={'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000,
                               10000],
                         'gamma': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000,
                                   10000],
                         'kernel': ('linear', 'poly', 'rbf')},
             scoring='accuracy')
```

```
#gridfit=grid.fit(x,y)
#gridfit.best_params_
```

Penentuan parameter dengan gridfit mengalami kendala dalam proses runing yang lama. Oleh karena itu, analisis dilakukan dengan metode kernel linier, rbf, dan poly dengan dilkaukan percobaan beberapa parameter

.

- **Analisis dengan metode kernel**

  Melakukan analisis menggunakan metode kernel untuk mencari metode terbaik yang memiliki hasil akurasi yang paling tinggi.

```
from sklearn import svm
svm1=svm.SVC(class_weight=None,C=100, gamma=100, kernel='rbf',
           random_state=100)
n=5
from sklearn.model_selection import StratifiedKFold
kf1=StratifiedKFold(n_splits=n, random_state=None)
```

```
svm1.fit(x,y)
y_pred1=svm1.predict(x)
y_pred1
```

```
#Membuat tempat
cm1=[]
total1=[]
ac1=[]
se1=[]
sp1=[]
```

```
for train_index1, test_index1 in kf1.split(x,y):
    x_train1, x_test1 = x[train_index1], x[test_index1]
    y_train1, y_test1 = y[train_index1], y[test_index1]
    svm1.fit(x_train1, y_train1)
    y_pred1 = svm1.predict(x_test1)
    cm1.append((confusion_matrix(y_test1, y_pred1)).astype(float))
for j in range (n):
    total.append(sum(sum(cm1[j])))
    ac1.append((cm1[j][0,0]+cm1[j][1,1])/total[j])
    se1.append(cm1[j][0,0]/(cm1[j][0,0]+cm1[j][0,1]))
    sp1.append(cm1[j][1,1]/(cm1[j][1,0]+cm1[j][1,1]))
akurasi=np.mean(ac1)
spesifisiti=np.mean(sp1)
sensitiviti=np.mean(se1)
print("Akurasi : ",akurasi)
print("Spesifisitas : ", spesifisiti)
print("Sensitivitas :", sensitiviti)
```

```
Akurasi :  0.8866666666666667
Spesifisitas :  0.06016326530612245
Sensitivitas : 0.9614710443821151
```

Kernel	C	Gamma	Akurasi
Linear	1	0.1	0,917
RBF	1	0.1	0.917
Polly	1	0.1	0.917
Linear	100	100	0.917
RBF	100	100	0.887
Linear	1	100	0.917
RBF	1	100	0.915
Polly	1	100	0.9106

Berdasarkan percobaan kombinasi 3 parameter diatas (kernel, cost, dan gamma) didapat akurasi yang paling besar sebesar 0,917 pada kernel linear dengan apapun nilai cost dan gamma-nya sedangkan kernel RBF dan Polly hanya pada nilai cost 1 dan gamma 0.1. Oleh karena itu metode kernel terbaik adalah metode kernel linear dengan perfomansi tiap fold sebagai berikut.

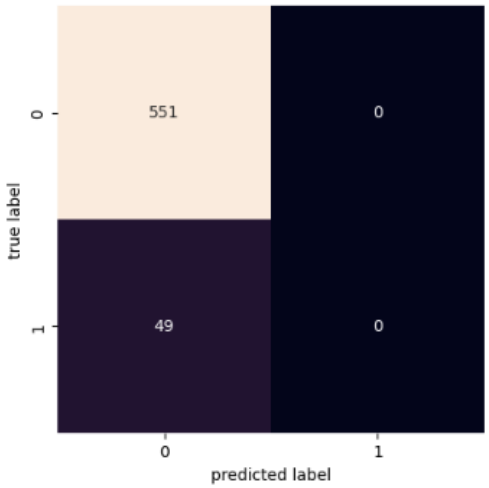
```
#performansi masing masing fold
df_k3=pd.DataFrame()
test=dict()
for j in range (3):
    test[j]=[]
for i in range (n):
    test[0].append(ac[i])
    test[1].append(sp[i])
    test[2].append(se[i])
for i in range (3):
    df_k3=pd.concat([df_k3,pd.DataFrame(test[i])],axis=1)
df_k3.columns=['Akurasi','Spesitifitas','Sensitivitas']
df_k3
```

	Akurasi	Spesitifitas	Sensitivitas
0	0.916667	0.0	1.0
1	0.916667	0.0	1.0
2	0.916667	0.0	1.0
3	0.916667	0.0	1.0
4	0.918333	0.0	1.0

- **Evaluasi Model SVM**

```
#karena akurasi kernel linear > RBF maka yang dieval linear
from sklearn.metrics import confusion_matrix
conf_matrix=confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, square=True, annot=True, fmt='d', cbar=False)
plt.ylabel('true label')
plt.xlabel('predicted label')
```

```
Text(0.5, 23.52222222222222, 'predicted label')
```



Berdasarkan plot matriks sebelumnya menunjukkan bahwa orang sebenarnya tidak memiliki riwayat penyakit stroke dan dikelompokkan ke dalam orang yang tidak mempunyai stroke sebanyak 551 sampel. Orang yang sebenarnya mempunyai penyakit stroke namun dikelompokkan tidak memiliki penyakit stroke sebanyak 49 sampel.

```
akurasi=accuracy_score(y_test, y_pred)
presisi=precision_score(y_test, y_pred, pos_label=0, average=None)
recalls=recall_score(y_test, y_pred, pos_label=0, average=None)
sensitivitas=conf_matrix[0,0]/(conf_matrix[0,0]+conf_matrix[0,1])
spesifisitas=conf_matrix[1,1]/(conf_matrix[1,0]+conf_matrix[1,1])
print('Akurasi', akurasi)
print('Spesifitas', spesifisitas)
print('Sensitivitas', sensitivitas)
print('Presisi', presisi)
print('Recalls', recalls)
```

```
Akurasi 0.9183333333333333
Spesifitas 0.0
Sensitivitas 1.0
Presisi [0.91833333 0.        ]
Recalls [1. 0.]
```

Akurasi	Spesifitas	Sensitivitas
0,917	0,0	1

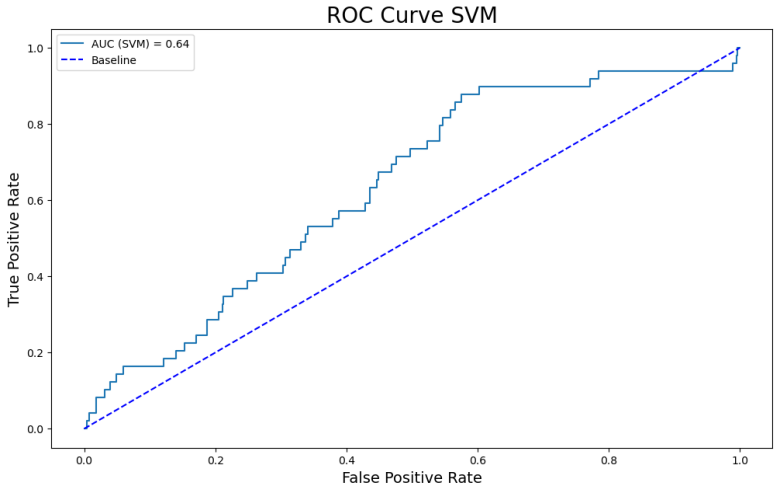
- **Membuat kurva ROC**

  Kurva ROC berfungsi untuk menyajikan ilustrasi performansi dari *binary classifier system* dalam prediksi.

```
from sklearn.svm import SVC
svc=SVC()
```

```
from sklearn import svm
model=SVC(probability=True).fit(x_train,y_train)
probs=model.predict_proba(x_test)[:,1]
```

```
from sklearn.metrics import roc_auc_score, roc_curve
import matplotlib.pyplot as plt
auc = roc_auc_score(y_test, probs)
fpr, tpr, threshold = roc_curve(y_test, probs)
plt.figure(figsize=(10, 5))
plt.plot(fpr, tpr, label=f'AUC (SVM) = {auc:.2f}')
plt.plot([0, 1], [0, 1], color='blue', linestyle='--', label='Baseline')
plt.title('ROC Curve', size=20)
plt.xlabel('False Positive Rate', size=14)
plt.ylabel('True Positive Rate', size=14)
plt.legend();
```



Kurva ROC yang semakin mendekat ke sudut kiri atas menunjukkan kinerja yang semakin baik. Berdasarkan kurva ROC tersebut didapat nilai AUC (SVM) sebesar 0,64 yang berarti model yang telah dibuat memiliki 64% area di bawah kurva sehingga model dapat dikatakan kurang baik.

E. DECISION TREE

- **Import data dan mendefinisikan data yang dianalisis menggunakan decision tree.**

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
```

```
data=pd.read_csv("dataweek11.csv",delimiter=',')
```

```
#Mendefiisikan data untuk decision tree dengan nama datatree
#diambil data kategorik
df3=data.drop(['id'], axis=1)
df3.dtypes
```

```
gender              object
age                float64
hypertension         int64
heart_disease        int64
ever_married        object
work_type           object
Residence_type      object
avg_glucose_level  float64
bmi                float64
smoking_status      object
stroke               int64
dtype: object
```

- **Melakukan coding features kategori menjadi angka**

```
#endcoding features kategorik
df3['gender']=df3['gender'].map({'Female':0,'Male':1,'Other':2})
df3['ever_married']=df3['ever_married'].map({'Yes':0,'No':1})
df3['work_type']=df3['work_type'].map({'Private':0,'Self-employed':1,'children':2,'Govt_job':3,'Never_worked':4})
df3['Residence_type']=df3['Residence_type'].map({'Urban':0,'Rural':1})
df3['smoking_status']=df3['smoking_status'].map({'never smoked':0,'Unknown':1,'formerly smoked':2,'smokes':3})
```

- **Mendefinisikan variabel target sebagai y3 dan features sebagai x3**

```
#Mendefinisikan target menjadi y3 dan features menjadi x3
y3 = df3['stroke']
x3= df3[['hypertension','ever_married','heart_disease','age','avg_glucose_level']]
x3
```

- **Membagi dataset menjadi training dan testing**

  Data dibagi menjadi data training sebesar 70% dan data testing sebesar 30%.

```
# Split dataset into training set and test set
x3_train, x3_test, y3_train, y3_test = train_test_split(x3, y3, test_size=0.3, random_state=500) # 70% training and 30% test
```

- **Membuat classifier tree**

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(x3_train,y3_train)

#Predict the response for test dataset
y3_pred = clf.predict(x3_test)
```

- **Mencari akurasi model**

```
# Model Accuracy
print("Accuracy:",metrics.accuracy_score(y3_test, y3_pred))
```

```
Accuracy: 0.8377777777777777
```

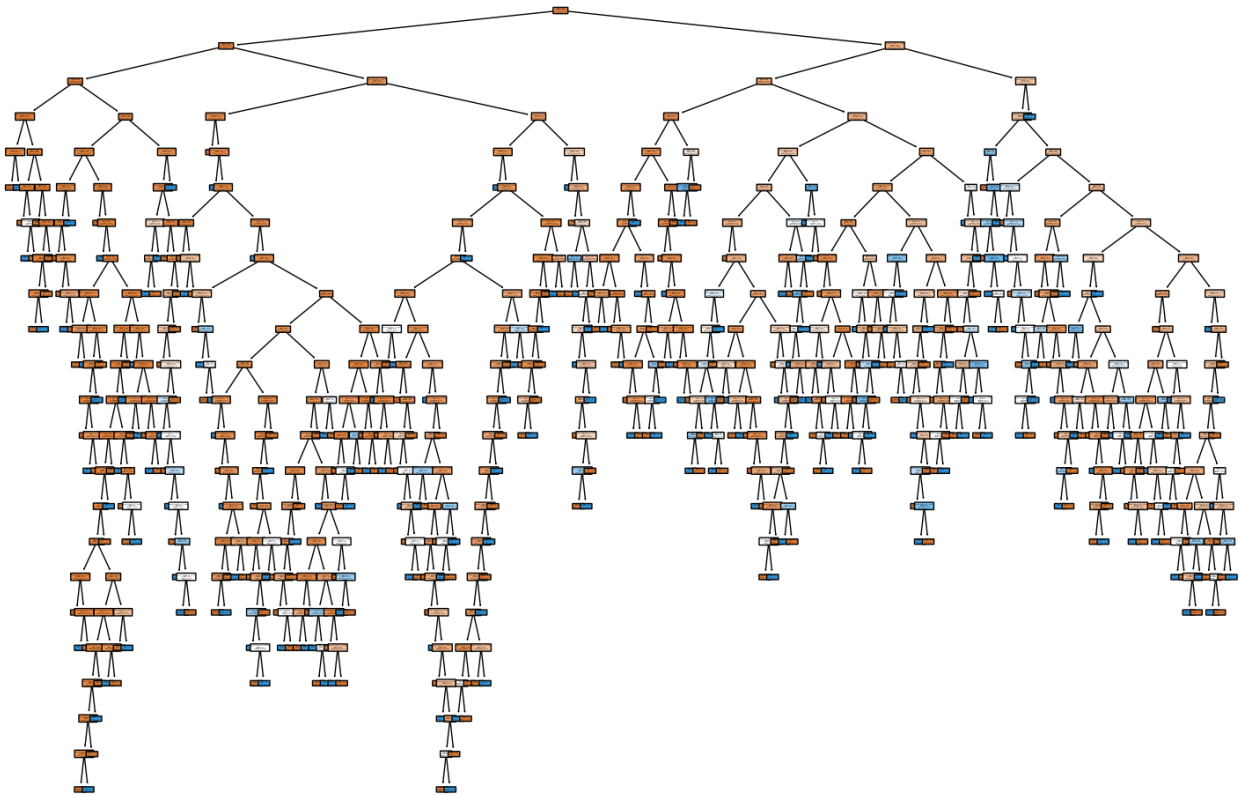
- **Membuat grafik decision tree**

```
from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
from sklearn import tree
```

```
dot_data = StringIO()
feature_cols=['hypertension','ever_married','heart_disease','age','avg_glucose_level']
tree.export_graphviz(clf, out_file="tkmaulala_tree.dot",
                filled=True, rounded=True,
                special_characters=True, feature_names = feature_cols,class_names=['Yes', 'No'])
```

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(18, 12))
plot_tree(clf,
          feature_names=['hypertension','ever_married','heart_disease','age','avg_glucose_level'],
          class_names=['Yes', 'No'],
          rounded=True, # Rounded node edges
          filled=True, # Adds color accoding to class
          proportion=True); # Displays the proportions of class samples instead of the whole number of samples
```



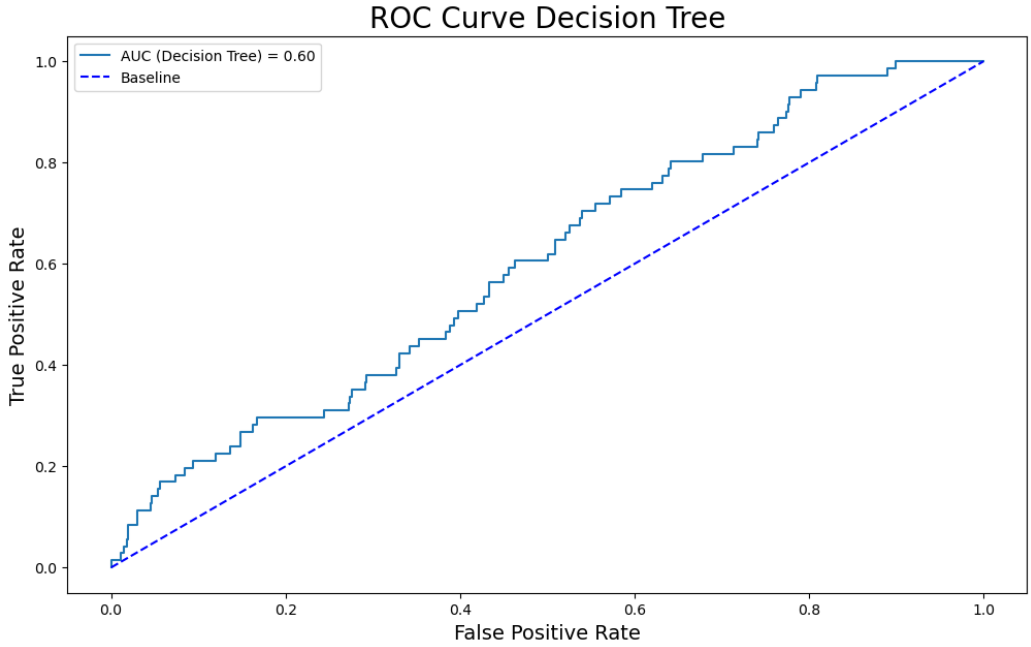
Decision node yang terbentuk tidak dapat dianalisis secara visual. Sebagai contoh untuk memahami decision tree tersebut dapat dilihat dari skema berikut:

```
from sklearn.tree import export_text
r = export_text(clf, feature_names=['hypertension','ever_married','heart_disease','age','avg_glucose_level'])
print(r)
```

```
|--- age <= 67.50
|   |--- age <= 53.50
|   |   |--- age <= 37.50
|   |   |   |--- avg_glucose_level <= 57.93
|   |   |   |   |--- avg_glucose_level <= 57.92
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- avg_glucose_level >  57.92
|   |   |   |   |   |--- class: 1
|   |   |   |--- avg_glucose_level >  57.93
|   |   |   |   |--- age <= 1.36
|   |   |   |   |   |--- age <= 1.28
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- age >  1.28
|   |   |   |   |   |   |--- avg_glucose_level <= 74.45
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- avg_glucose_level >  74.45
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- age >  1.36
|   |   |   |   |   |--- age <= 31.50
```

Pengklasifikasian berdasarkan riwayat stroke. Dapat diketahui bahwa age <= 67,50 sampai avg_glucose_level <=57,92 masuk dalam class 0 atau tidak mempunyai riwayat stroke.

- **Membuat Kurva ROC**



Kurva ROC yang semakin mendekat ke sudut kiri atas menunjukkan kinerja yang semakin baik. Berdasarkan kurva ROC tersebut didapat nilai AUC (Decision Tree) sebesar 0,6 yang berarti model yang telah dibuat memiliki 60% area di bawah kurva sehingga model dapat dikatakan kurang baik.

F. NAIVE BAYES

- **Naive bayes.**

  Naïve Bayes adalah teknik klasifikasi yang berdasarkan teorema Bayes dengan asumsi indepedensi antar prediktor. Secara sederhana, klasifikasi metode ini mengansumsikan bahwa keberadaan fitur tertentu tidak terkait dengan fitur lainnya. *Bayesian classification* adalah pengklasifikasian statistik yang dapat digunakan untuk memprediksi probabilitas keanggotaan suatu class.

- **Keunggulan NB**
  - Mudah dan cepat untuk memprediksi kelas dari suatu kumpulan data besar.
  - Jika asumsi independensi berlaku, pengklasifikasi Naïve Bayes berperforma lebih baik dibandingkan dengan model lain seperti regresi logistik dan memerlukan data training yang lebih sedikit.
  - Berkinerja baik dalam kasus variabel masukan kategorikal dibandingkan dengan variabel numerik. Untuk variabel numerik, diasumsikan berdistribusi normal.
- **Kelemahan NB**
  - Jika variabel kategorikal memiliki kategori yang tidak diamati dalam kumpulan data pelatihan, maka model akan menetapkan probabilitas 0 (nol) dan tidak akan dapat membuat prediksi.
  - Batasan lain adalah asumsi predictor independent. Dalam kehidupan nyata, hamper tidak mungkin kita mendapatkan seperangkat predictor yang sepenuhnya independent.
- **Teorema Bayes**

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)}$$

  X: data dengan kelas yang tidak ketahui
  H: data hipotesis yang merupakan class tertentu
  P(H): probablitas hipotesis H
  P (X): probabilitas X
  P(H|H): probablitas hipotesis H berdasarkan kondisi x
  P(X|H): probablitas hipotesis X berdasarkan kondisi H

- **Data**

  Terdapat 5 variabel dengan 3 variabel kategorik dan 2 varibael numerik yang sudah distandarisasi /normalisasi.

	hypertension	heart_disease	ever_married	age	avg_glucose_level
0	0	1	1	0.817073	0.801173
1	0	0	1	0.743902	0.678875
2	0	1	1	0.975610	0.234159
3	0	0	1	0.597561	0.535793
4	1	0	1	0.963415	0.549141
...	...	...	...	...	...
2995	0	0	1	0.951220	0.137678
2996	0	0	1	0.426829	0.117957
2997	0	1	1	0.865854	0.691668
2998	0	0	1	0.695122	0.185387
2999	0	0	0	0.146341	0.254526

- **Mendefiniskan target (y) dan feature (x) dan menentukan data training dan testing**

```
#definisi x dan y
y=df['stroke']
x=df.drop(['id','stroke'],axis=1)
```

```
#split data
x_train, x_test, y_train, y_test= train_test_split(x,y,test_size=0.25, random_state=123)
```

Pada pemodelan Naive Bayes, data dibagi menjadi dua yaitu data testing sebanyak 20%, serta data training sebanyak 70%. Data training digunakan untuk membangun algoritma klasifikasi dengan cara menghitung probabilitas bersyarat Sedangkan, data testing digunakan untuk membuat keputusan stroke tidaknya dari algoritma yang dibangun oleh data training.

X_test (600 data)

	hypertension	heart_disease	ever_married	age	avg_glucose_level
1690	0	0	1	0.756098	0.033900
1798	0	0	1	0.682927	0.782422
417	0	0	1	0.451220	0.686265
1599	0	0	1	0.829268	0.120820
1206	0	0	0	0.000000	0.051450
...	...	...	...	...	...
813	0	0	0	0.500000	0.239054
174	0	0	1	0.951220	0.025171
2874	0	0	0	0.292683	0.137863
243	1	1	1	0.829268	0.888093
447	0	0	1	0.792683	0.226261

X_train (2400 data)

	hypertension	heart_disease	ever_married	age	avg_glucose_level
497	0	0	0	0.170732	0.343202
21	1	0	1	0.634146	0.822418
1710	0	0	1	0.548780	0.013994
2323	0	0	0	0.219512	0.083456
1516	1	0	1	0.646341	0.108350
...	...	...	...	...	...
1147	0	0	1	0.451220	0.107149
2154	0	0	0	0.085366	0.031683
1766	0	0	1	0.317073	0.186357
1122	0	0	0	0.231707	0.192222
1346	0	0	1	0.597561	0.668483

2400 rows × 5 columns

Y_test

```
1690    0
1798    0
417     0
1599    0
1206    0
       ..
813     0
174     1
2874    0
243     1
447     0
Name: stroke, Length: 600
```

Y train

```
497     0
21      1
1710    0
2323    0
1516    0
       ..
1147    0
2154    0
1766    0
1122    0
1346    0
Name: stroke, Length: 2400,
```

- **Akurasi model training**

Kebaikan pemodelan dinilai dari seberapa baik model tersebut menghasilkan keputusan data testing (uji coba) yang sesuai dengan kelas (target) sebenarnya. Hal ini dapat dievaluasi dari nilai akurasi model. Pada pemodelan Naive Bayes, diperoleh nilai akurasi sebesar 0,832083. Artinya, model Naive Bayes dengan melibatkan 11 variabel tersebut dapat memprediksi riwayat stroke seseorang dengan ketepatan klasifikasi 83,21%. Selain itu, disajikan matriks konfusi dari pemodelan Naive Bayes sebagai berikut.

```
from sklearn.naive_bayes import GaussianNB
nb=GaussianNB()
#fit train set pake gaussian naive bayes
nb.fit(x_train,y_train)
```

```
GaussianNB()
```

```
#compute the  accurancy of train set
nb.score(x_train,y_train)
```

```
0.8320833333333333
```

- **Confussion matrix**

```
from sklearn.metrics import confusion_matrix, accuracy_score
#create an empty dictionary for confussion metrices
conf_matrix={}
#add new keys and assign the confussion matrices results
classif_name=['nb']
for name in classif_name:
    conf_matrix[name]=pd.DataFrame(data=confusion_matrix(y_test,y_predict[name]),
                                   columns=df['stroke'].unique(),
                                   index=df['stroke'].unique())
```

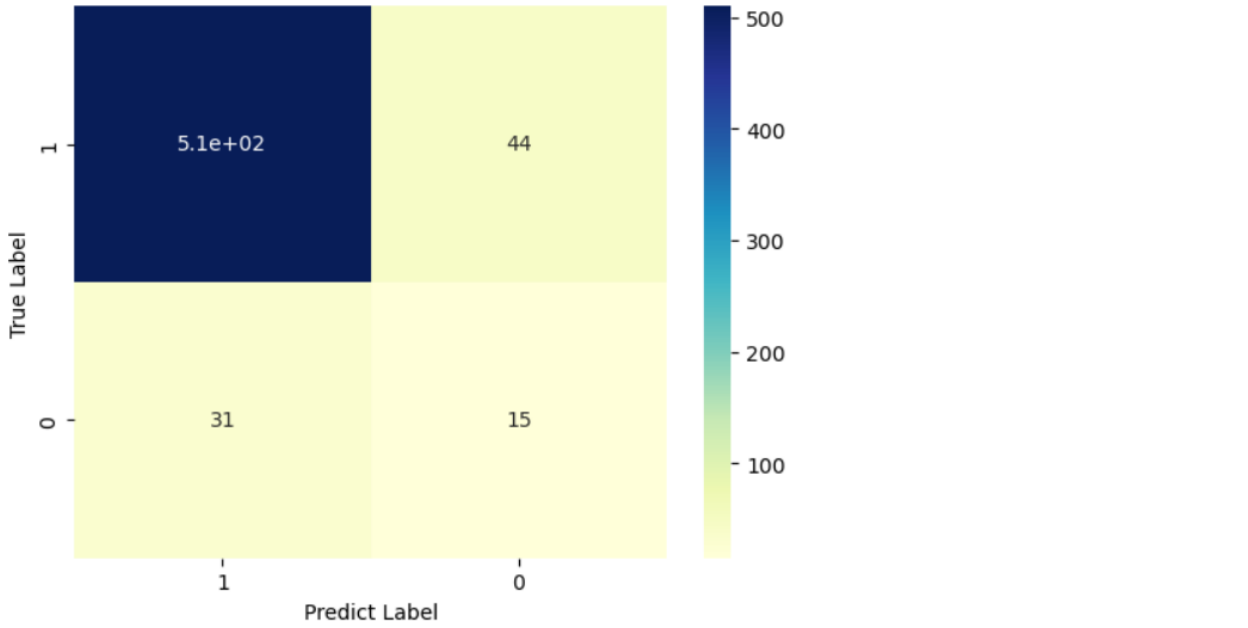
```
conf_matrix['nb']
```

	1	0
1	510	44
0	31	15

Plot matriks konfusi disamping menunjukkan bahwa sampel pasien (testing) yang sebenarnya stroke (1) dan diklasifikasikan model sebagai stroke(1) sebanyak 510 sampel. Sedangkan, sampel yang sebenarnya tidak stroke (0) dan diklasifikasikan model sebagai tidak stroke(0) sebanyak 31 sampel. Sementara itu, kesalahan algoritma Naive Bayes dalam mengklasifikasikan riwayat penyakit stroke sebanyak 59 sampel.

- **Dibuat heatmap mempermudah visualisasi**

```
#Create heatmap for NB confusion NB
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(conf_matrix['nb'], annot=True, cmap='YlGnBu'). set(ylabel='True Label', xlabel='Predict Label')
plt.show()
```



- **Probabilitas per observasi**

Probabilitas keputusan pasien stroke tidaknya yang dihasilkan menggunakan metode Naive Bayes yaitu sebagai berikut.

```
nb.predict_proba(x_test) #probabilotas per observasi
```

```
array([[9.36341707e-01, 6.36582933e-02],
       [6.69682293e-01, 3.30317707e-01],
       [9.79958169e-01, 2.00418313e-02],
       ...,
       [9.99981256e-01, 1.87436094e-05],
       [2.26006120e-05, 9.99977399e-01],
       [9.18691626e-01, 8.13083740e-02]])
```

Pada output di atas, terlihat bahwa pada sampel ke-1 diklasifikasikan sebagai pasien yang tidak stroke dikarenakan probabilitas pada kelas 0 = 0,93641 lebih besar daripada probabilitas pada kelas 1 = 0,636582.

- **Kurva ROC**

Kurva ROC Naive Bayes

```
#Model akurasi
print('Akurasi Metode Naive Bayes:',metrics.accuracy_score(y_testbayes,nb.predict(x_testbayes)))
```

```
Akurasi Metode Naive Bayes: 0.88
```

```
C:\Users\DELL\anaconda3\lib\site-packages\sklearn\base.py:566: FutureWarning: Arrays of bytes/str
imal numbers if dtype='numeric'. This behavior is deprecated in 0.24 and will be removed in 1.1 (
ert your data to numeric values explicitly instead.
  X = check_array(X, **check_params)
```

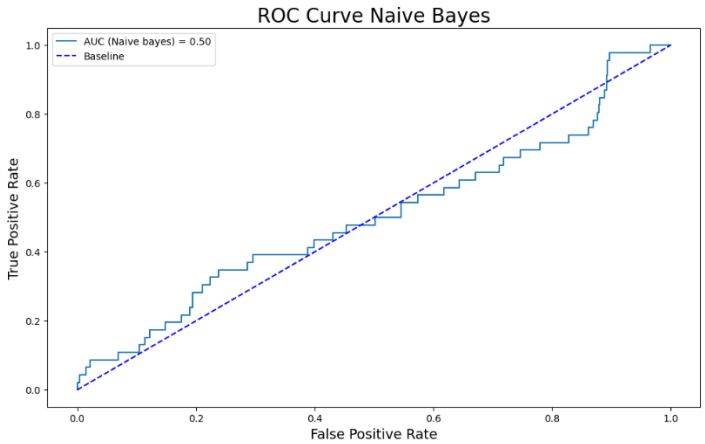
```
from sklearn.svm import SVC
svc = SVC()
```

```
from sklearn import svm
modelbayes = SVC(probability = True).fit(x_trainbayes,y_trainbayes)
probsbayes = modelbayes.predict_proba(x_testbayes)[:,1]
```

```
from sklearn.metrics import roc_auc_score, roc_curve
import matplotlib.pyplot as plt

auc = roc_auc_score(y_testbayes, probsbayes)
fpr, tpr, threshold = roc_curve(y_testbayes, probsbayes)

plt.figure(figsize=(12, 7))
plt.plot(fpr, tpr, label=f'AUC (Naive bayes) = {auc:.2f}')
plt.plot([0, 1], [0, 1], color='blue', linestyle='--', label='Baseline')
plt.title('ROC Curve Naive Bayes', size=20)
plt.xlabel('False Positive Rate', size=14)
plt.ylabel('True Positive Rate', size=14)
plt.legend();
```



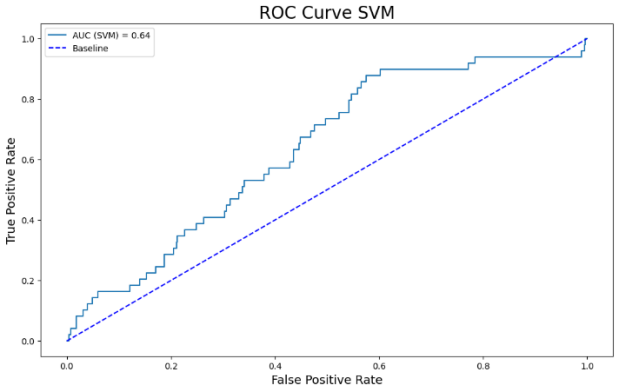
Berdasarkan kurva ROC tersebut, nilai AUC sebesar 0,5 yang berarti model yang telah dibuat memiliki 50% area di bawah kurva sehingga model dapat dikatakan tidak baik.

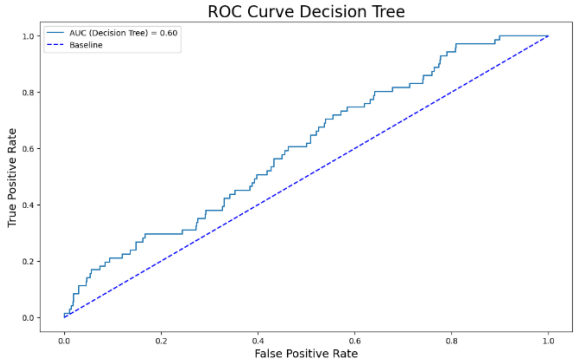
G. PERBANDINGAN METODE

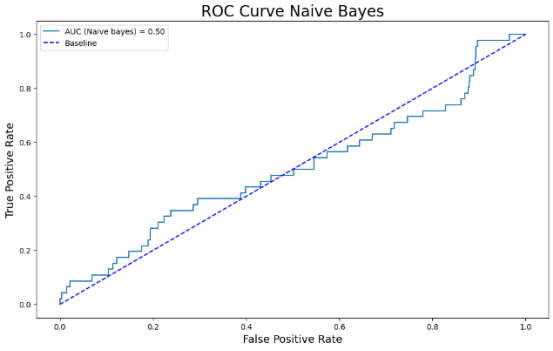
Metode SVM, *Decision Tree*, dan *Naive Bayes* dapat digunakan untuk mengklasifikasikan data pada kelas tertentu berdasarkan fitur/variabel lain yang diduga mempengaruhi. Perbandingan kebaikan model dalam mengklasifikasikan riwayat stroke seseorang dari ketiga metode yaitu sebagai berikut.

Metode	Akurasi	Spesifisitas	Sensitivitas	Presisi	F1 Score	AUC
SVM	0,918	0,000	1,000	[0,918;0,000]		0,640
Decision Tree	0,840	0,889	0,268	[0,934;0,171]	[0,911; 0,209]	0,600
Naïve Bayes	0,88	0,920	0,330	[0,940;0,250]	[0,930;0,290]	0,500

Berdasarkan tabel sebelumnya, diketahui bahwa metode SVM merupakan metode terbaik dalam mengelompokkan riwayat penyakit stroke seseorang. Hal tersebut berdasarkan nilai akurasi model SVM paling tinggi dan nilai AUC juga menunjukkkan nilia yang paling tinggi sehingga metode SVM merupakan metode terbaik dalam analisis stroke dataset. Nilai akurasi SVM sebesar 0,918 menunjukkan bahwa metode SVM dapat meklasifikasikan riwayat stroke pasien sebesar 91,8% dari keseluruhan sampel. Berikut perbandingan ketiga metode berdasarkan visualisasi kurva ROC.







Berdasarkan visualisasi ketiga kurva tersebut, terlihat bahwa kurva ROC SVM memiliki garis yang lebih dekat ke kiri atas sehingga meunjuukan SVM adalah model yang terbaik.