UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS
MATEMATIKA
DAN ILMU
PENGETAHUAN
ALAM

PROYEK AKHIR SAINS DATA

Penerapan Algoritma Klasifikasi dalam Machine Learning : Mendeteksi Penyakit Jantung

Dosen Fasilitator: Devvi Sarwinda, M.Kom.

Anggota Kelompok 1:

Bisma Rohpanca Joyosumarto	2106635581
Giovany Valencia Rywandi	2106652120
M.Irfansyah	2106701255
Romauli Graciella Debora	2106722575

Daftar Isi

01 Pendahuluan

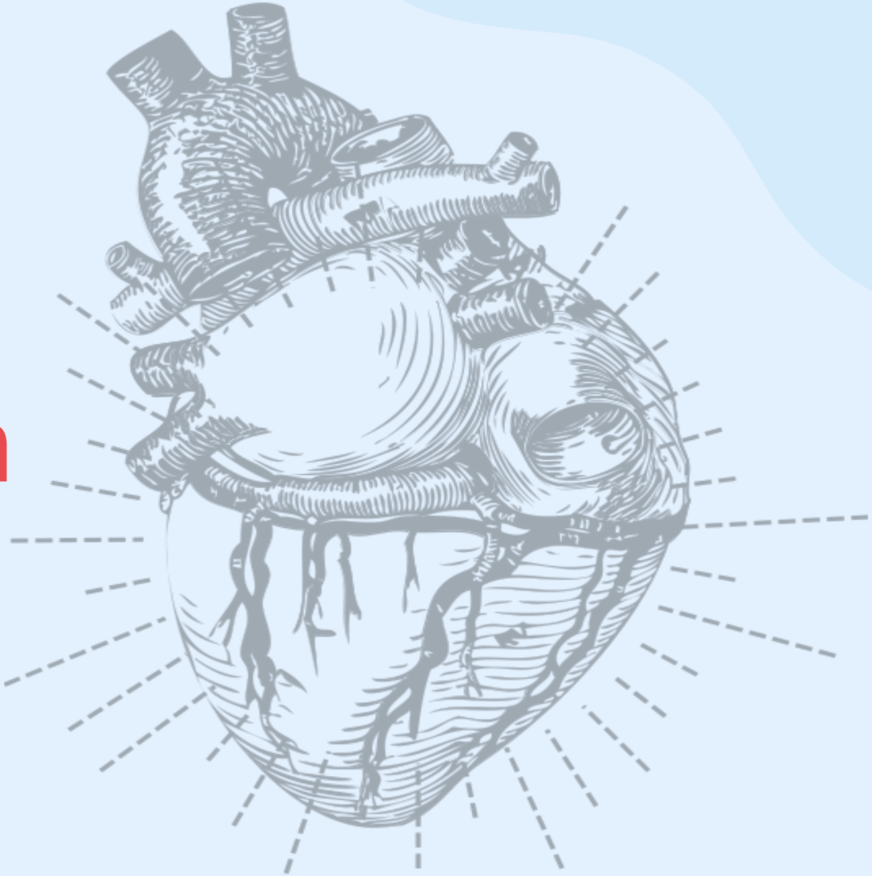
02 Exploratory Data Analysis

03 Pemodelan

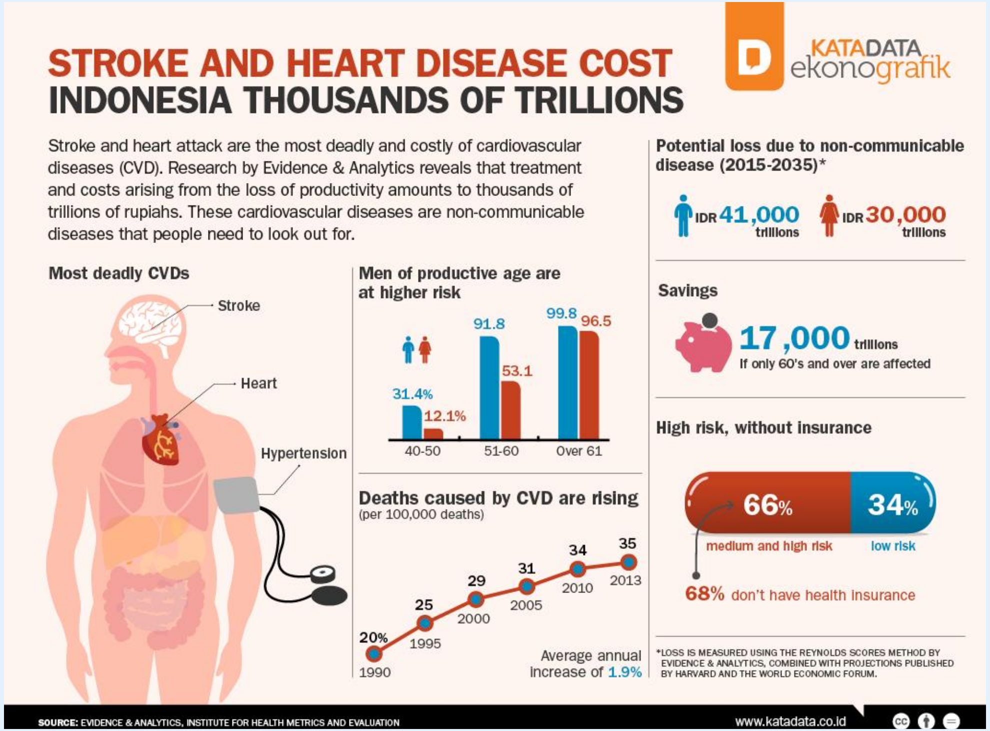
04 Kesimpulan dan Saran

01

Pendahuluan



Penyakit yang Mengancam Dunia



KATADATA ekonografik

STROKE AND HEART DISEASE COST INDONESIA THOUSANDS OF TRILLIONS

Stroke and heart attack are the most deadly and costly of cardiovascular diseases (CVD). Research by Evidence & Analytics reveals that treatment and costs arising from the loss of productivity amounts to thousands of trillions of rupiahs. These cardiovascular diseases are non-communicable diseases that people need to look out for.

Most deadly CVDs
Stroke
Heart
Hypertension
Men of productive age are at higher risk
31.4%
12.1%
91.8
53.1
99.8
96.5
40-50
51-60
Over 61
Deaths caused by CVD are rising
(per 100,000 deaths)
20%
1990
25
1995
29
2000
31
2005
34
2010
35
2013
Average annual increase of 1.9%
Potential loss due to non-communicable disease (2015-2035)*
IDR 41,000 trillions
IDR 30,000 trillions
Savings
17,000 trillions
If only 60's and over are affected
High risk, without insurance
66%
34%
medium and high risk
low risk
68% don't have health insurance

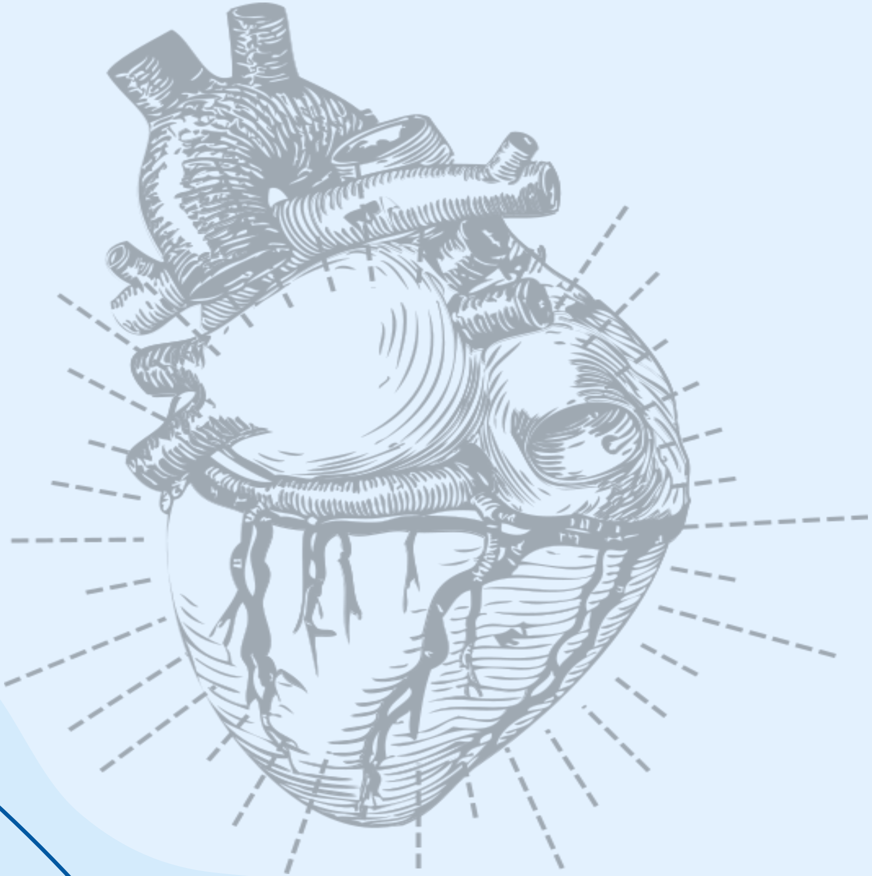
*LOSS IS MEASURED USING THE REYNOLDS SCORES METHOD BY EVIDENCE & ANALYTICS, COMBINED WITH PROJECTIONS PUBLISHED BY HARVARD AND THE WORLD ECONOMIC FORUM.

SOURCE: EVIDENCE & ANALYTICS, INSTITUTE FOR HEALTH METRICS AND EVALUATION

www.katadata.co.id

Berdasarkan **Data WHO**, **>17 Juta** orang **meninggal dunia karena jantung** dan pembuluh darah.

Data Riset Kesehatan Dasar (2018) mencatat **15** dari **1000** orang di Indonesia **menderita penyakit jantung.**

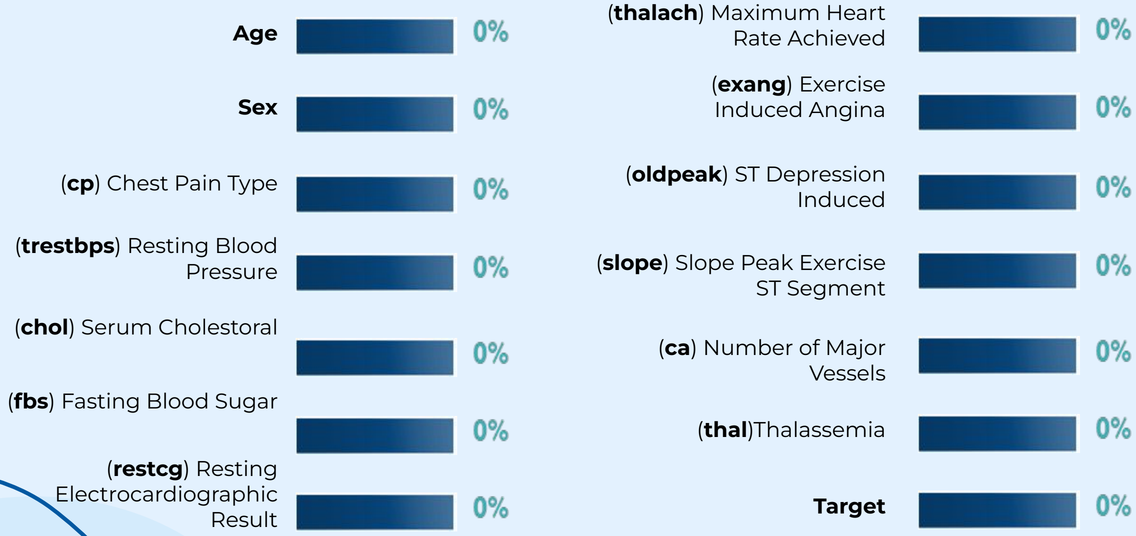


02

Exploratory Data Analysis

Dataset Overview

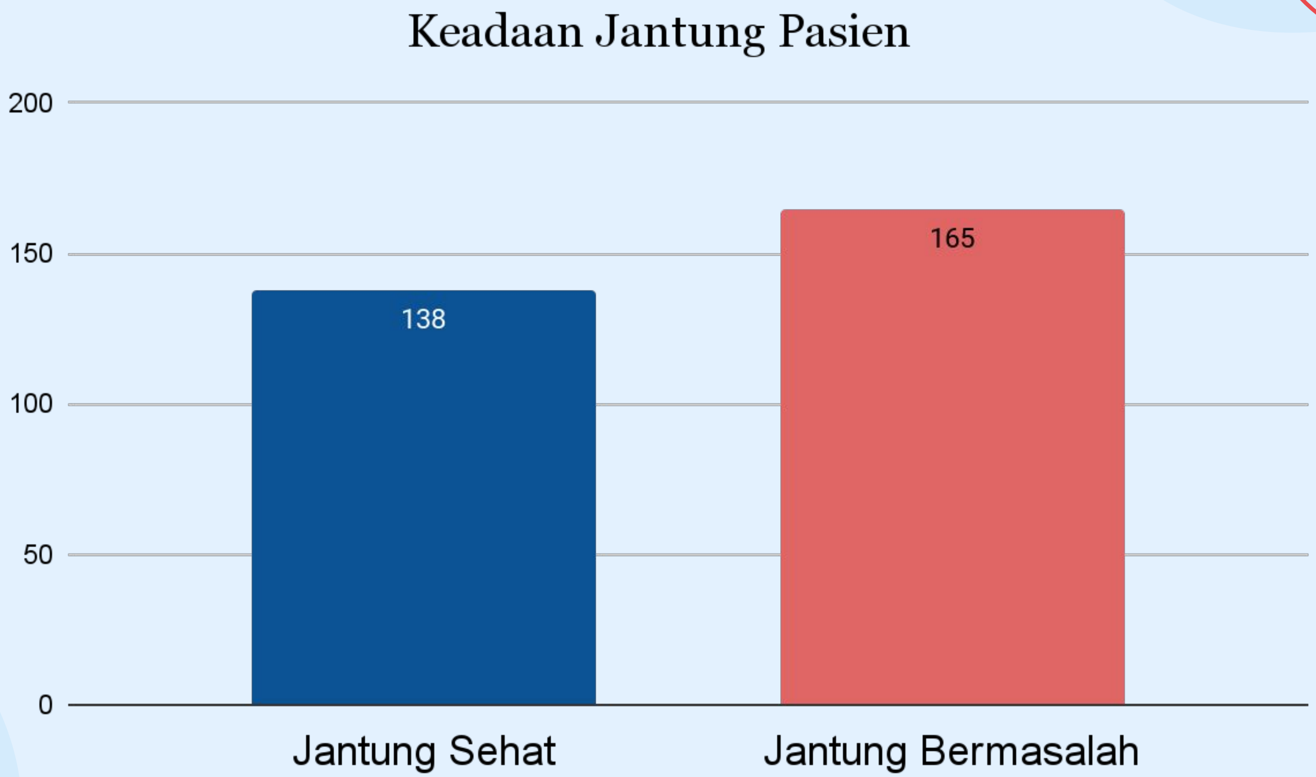
Persentase Missing Value



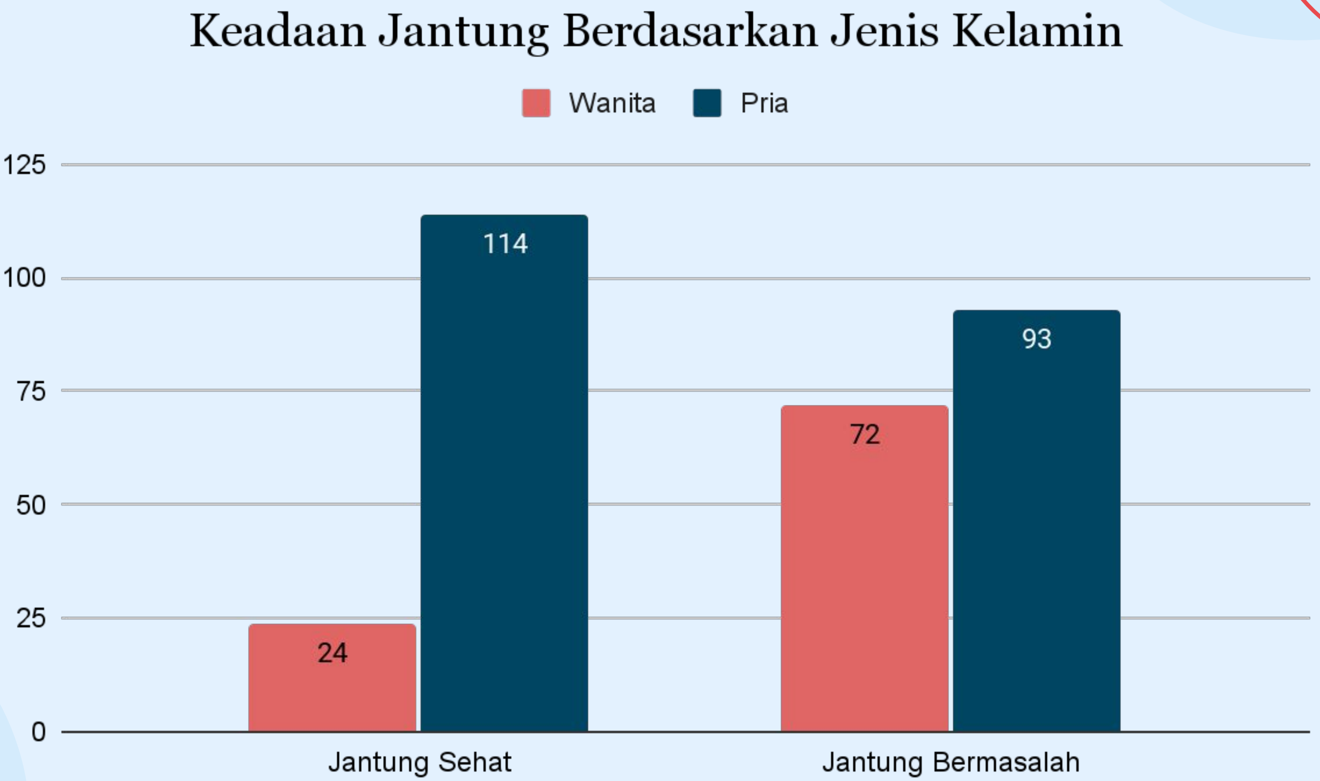
Data Type

Variable	Type	Variable	Type
Age	int	thalach	int
Sex	int	exang	int
cp	int	oldpeak	float
trestbps	int	slope	int
chol	int	ca	int
fbs	int	thal	int
restcg	int	Target	int

Keadaan Jantung Pasien



Keadaan Jantung Pasien

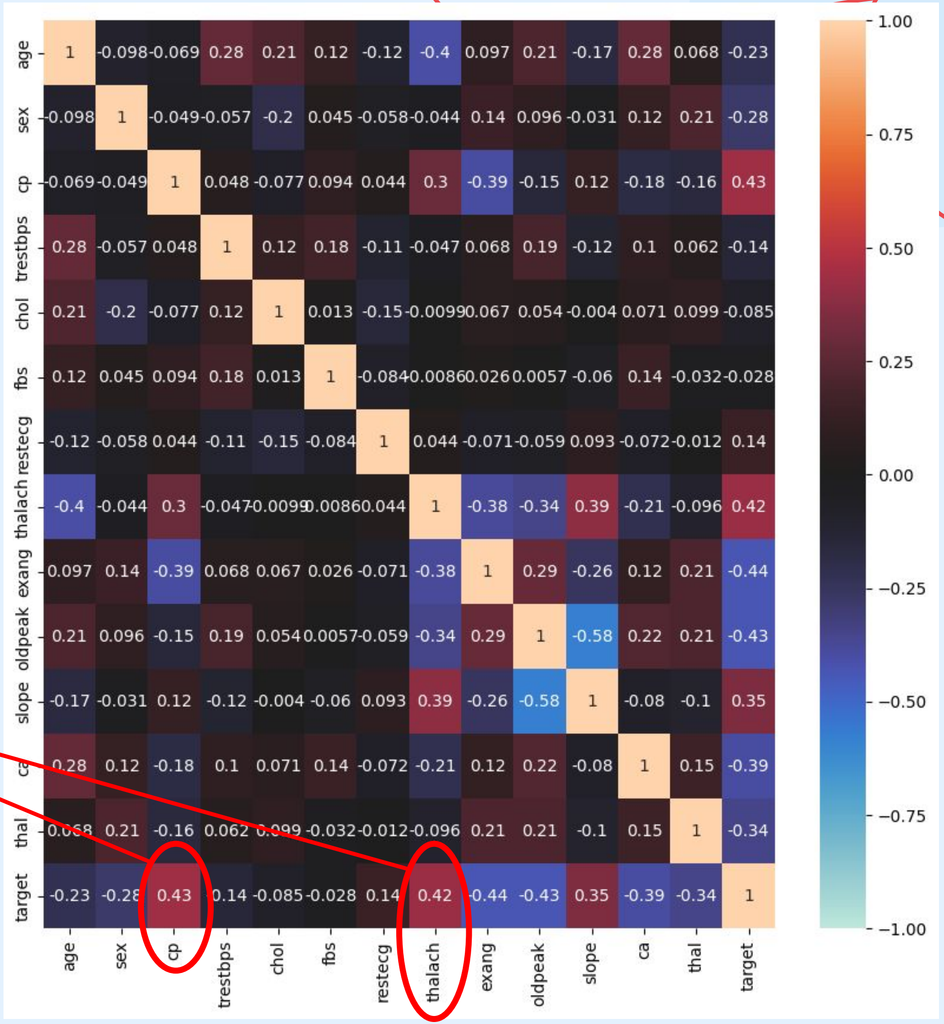


HEATMAP: Hubungan Antar Variabel

cp: Jenis Nyeri Dada

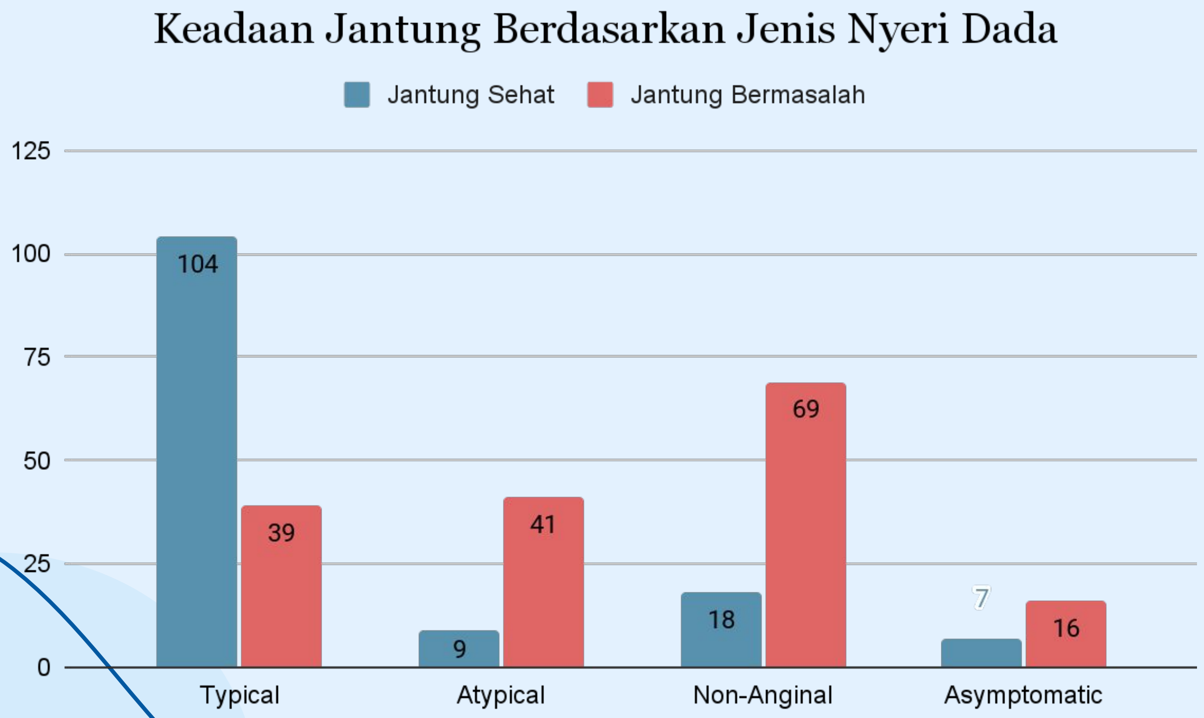
thalach: Tingkat Detak Jantung Maksimum

Variabel yang paling berpengaruh terhadap target.



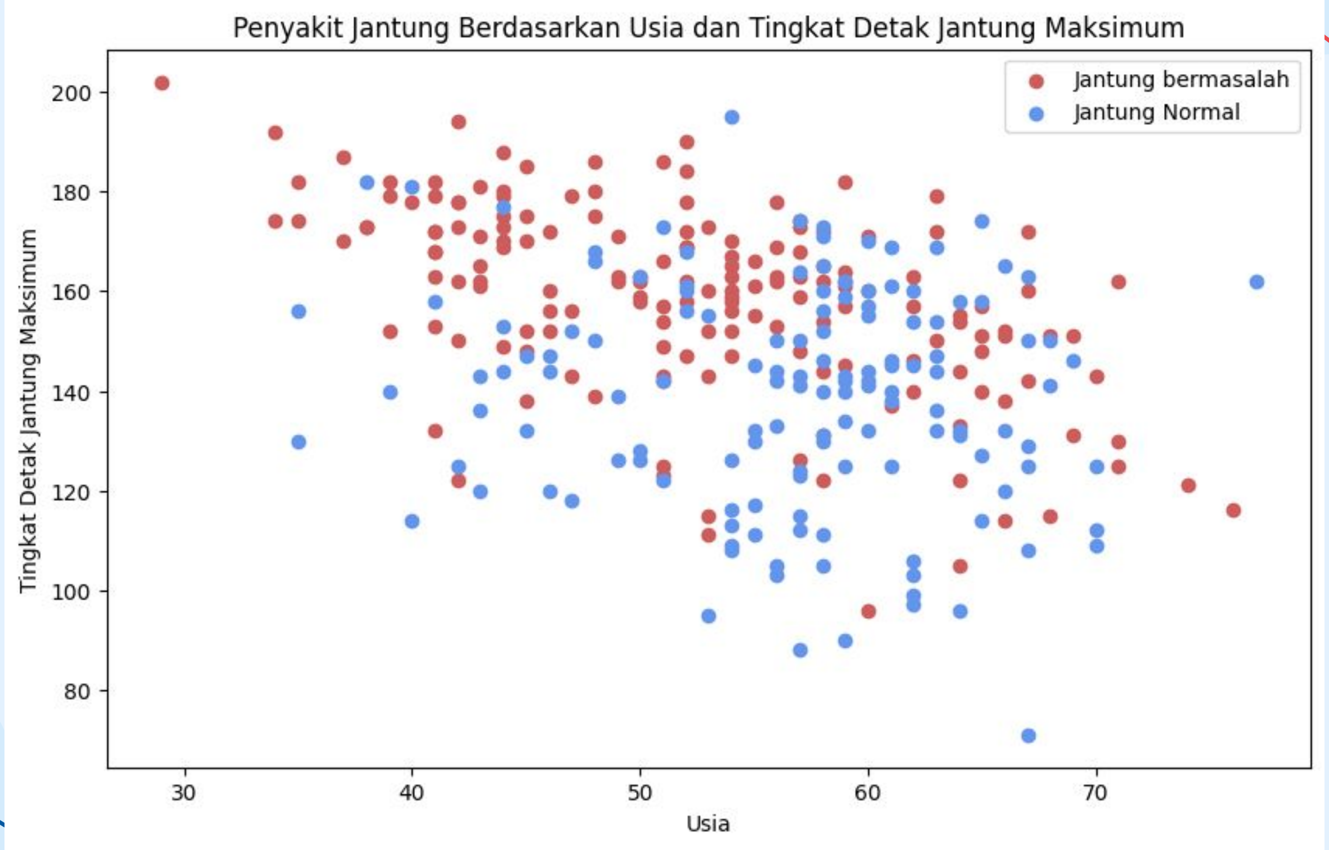
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1	-0.098	-0.069	0.28	0.21	0.12	-0.12	-0.4	0.097	0.21	-0.17	0.28	0.068	-0.23
sex	-0.098	1	-0.049	-0.057	-0.2	0.045	-0.058	-0.044	0.14	0.096	-0.031	0.12	0.21	-0.28
cp	-0.069	-0.049	1	0.048	-0.077	0.094	0.044	0.3	-0.39	-0.15	0.12	-0.18	-0.16	0.43
trestbps	0.28	-0.057	0.048	1	0.12	0.18	-0.11	-0.047	0.068	0.19	-0.12	0.1	0.062	-0.14
chol	0.21	-0.2	-0.077	0.12	1	0.013	-0.15	-0.0099	0.067	0.054	-0.004	0.071	0.099	-0.085
fbs	0.12	0.045	0.094	0.18	0.013	1	-0.084	-0.0086	0.026	0.0057	-0.06	0.14	-0.032	-0.028
restecg	-0.12	-0.058	0.044	-0.11	-0.15	-0.084	1	0.044	-0.071	-0.059	0.093	-0.072	-0.012	0.14
thalach	-0.4	-0.044	0.3	-0.047	-0.0099	-0.0086	0.044	1	-0.38	-0.34	0.39	-0.21	-0.096	0.42
exang	0.097	0.14	-0.39	0.068	0.067	0.026	-0.071	-0.38	1	0.29	-0.26	0.12	0.21	-0.44
oldpeak	0.21	0.096	-0.15	0.19	0.054	0.0057	-0.059	-0.34	0.29	1	-0.58	0.22	0.21	-0.43
slope	-0.17	-0.031	0.12	-0.12	-0.004	-0.06	0.093	0.39	-0.26	-0.58	1	-0.08	-0.1	0.35
ca	0.28	0.12	-0.18	0.1	0.071	0.14	-0.072	-0.21	0.12	0.22	-0.08	1	0.15	-0.39
thal	0.068	0.21	-0.16	0.062	0.099	-0.032	-0.012	-0.096	0.21	0.21	-0.1	0.15	1	-0.34
target	-0.23	-0.28	0.43	-0.14	-0.085	-0.028	0.14	0.42	-0.44	-0.43	0.35	-0.39	-0.34	1

Keadaan Jantung Berdasarkan Nyeri Dada



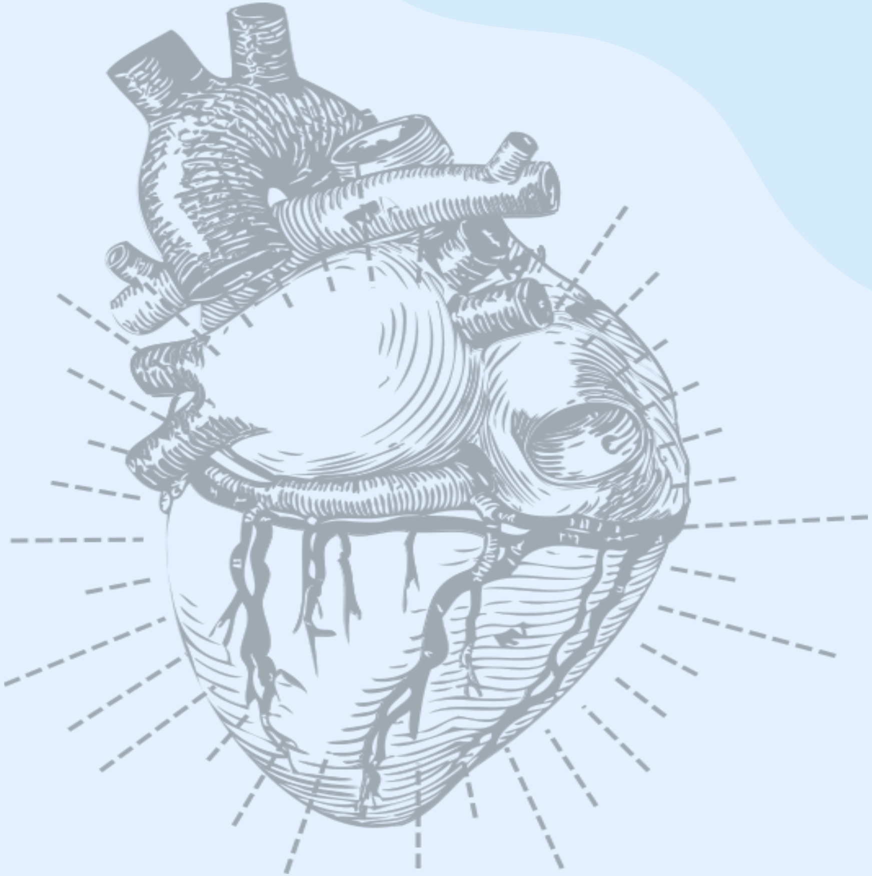
Typical Angina: Nyeri dada karena aliran darah yang berkurang ke jantung.
Atypical Angina: Nyeri dada tidak disebabkan jantung.
Non-Anginal: Nyeri dada karena spasme kerongkongan, tidak terkait jantung.
Asymptomatic: Tidak mengalami nyeri dada.

Korelasi Usia dan Tingkat Detak Jantung Maksimum

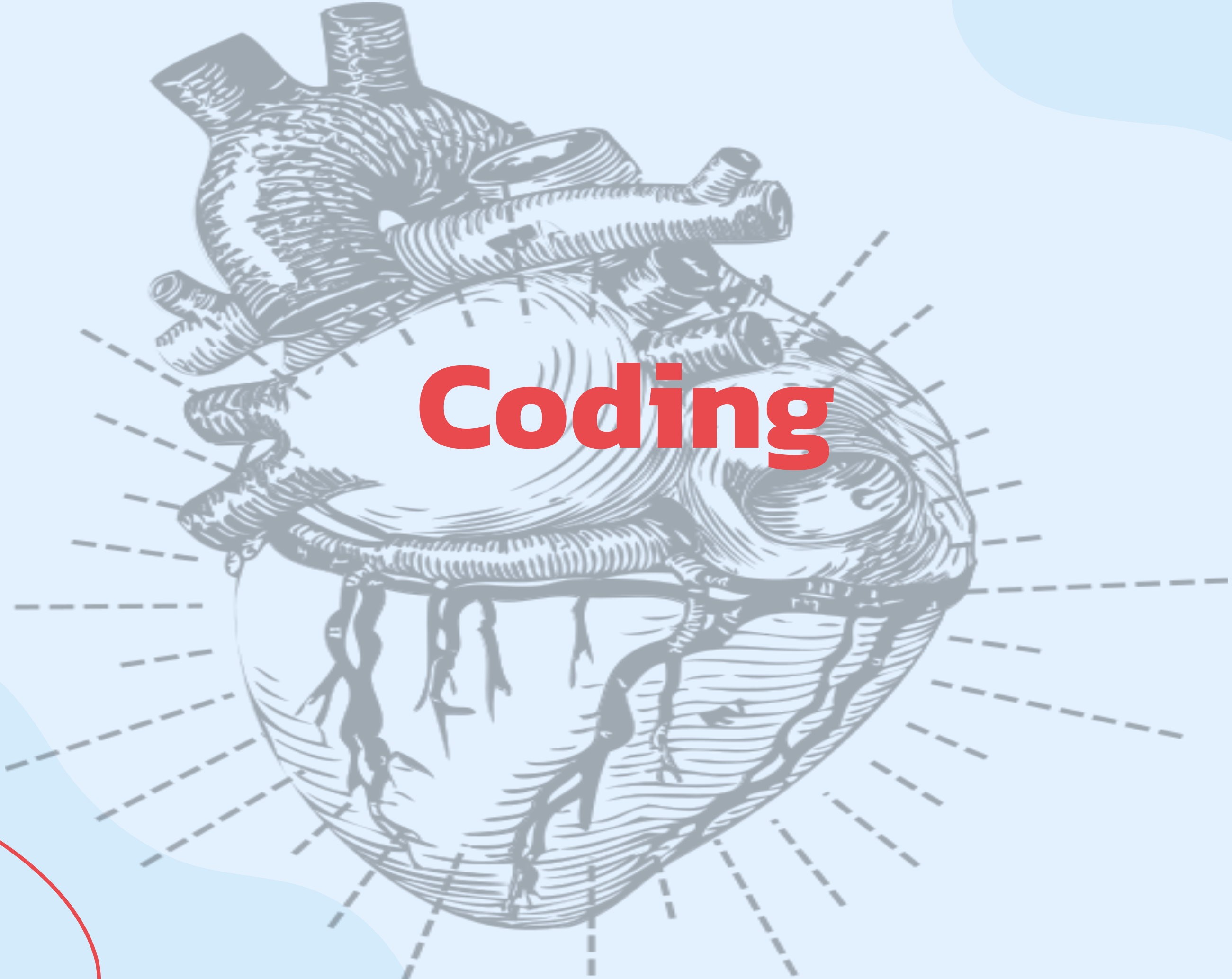


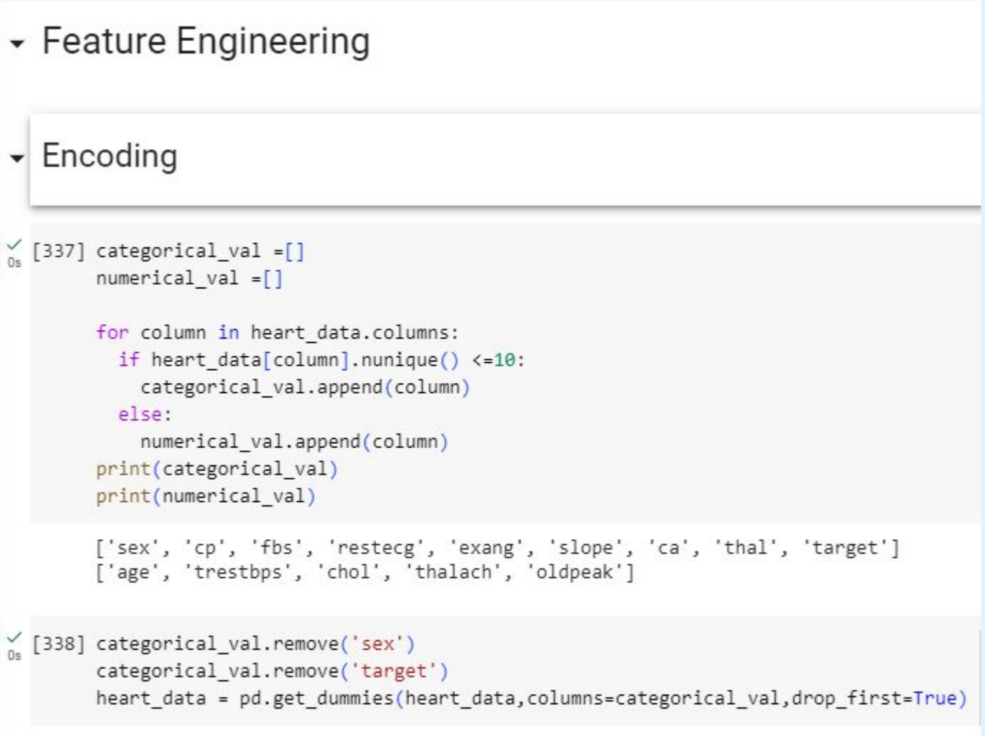
03

Pemodelan



Coding





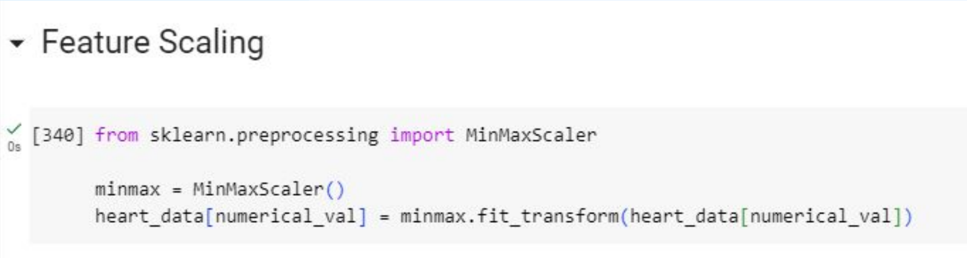
Feature Engineering

Encoding

```
[337] categorical_val =[]
      numerical_val =[]

      for column in heart_data.columns:
        if heart_data[column].nunique() <=10:
          categorical_val.append(column)
        else:
          numerical_val.append(column)
      print(categorical_val)
      print(numerical_val)
```

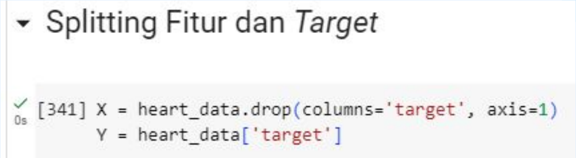
```
      ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']
      ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

```
[338] categorical_val.remove('sex')
      categorical_val.remove('target')
      heart_data = pd.get_dummies(heart_data,columns=categorical_val,drop_first=True)
```



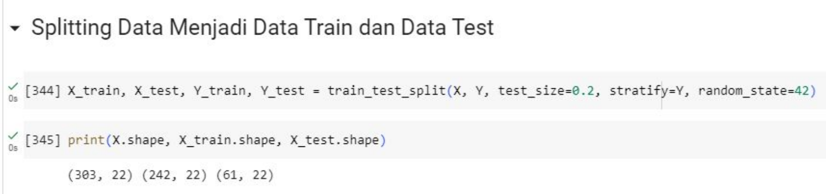
Feature Scaling

```
[340] from sklearn.preprocessing import MinMaxScaler

      minmax = MinMaxScaler()
      heart_data[numerical_val] = minmax.fit_transform(heart_data[numerical_val])
```



Splitting Fitur dan *Target*

```
[341] X = heart_data.drop(columns='target', axis=1)
      Y = heart_data['target']
```

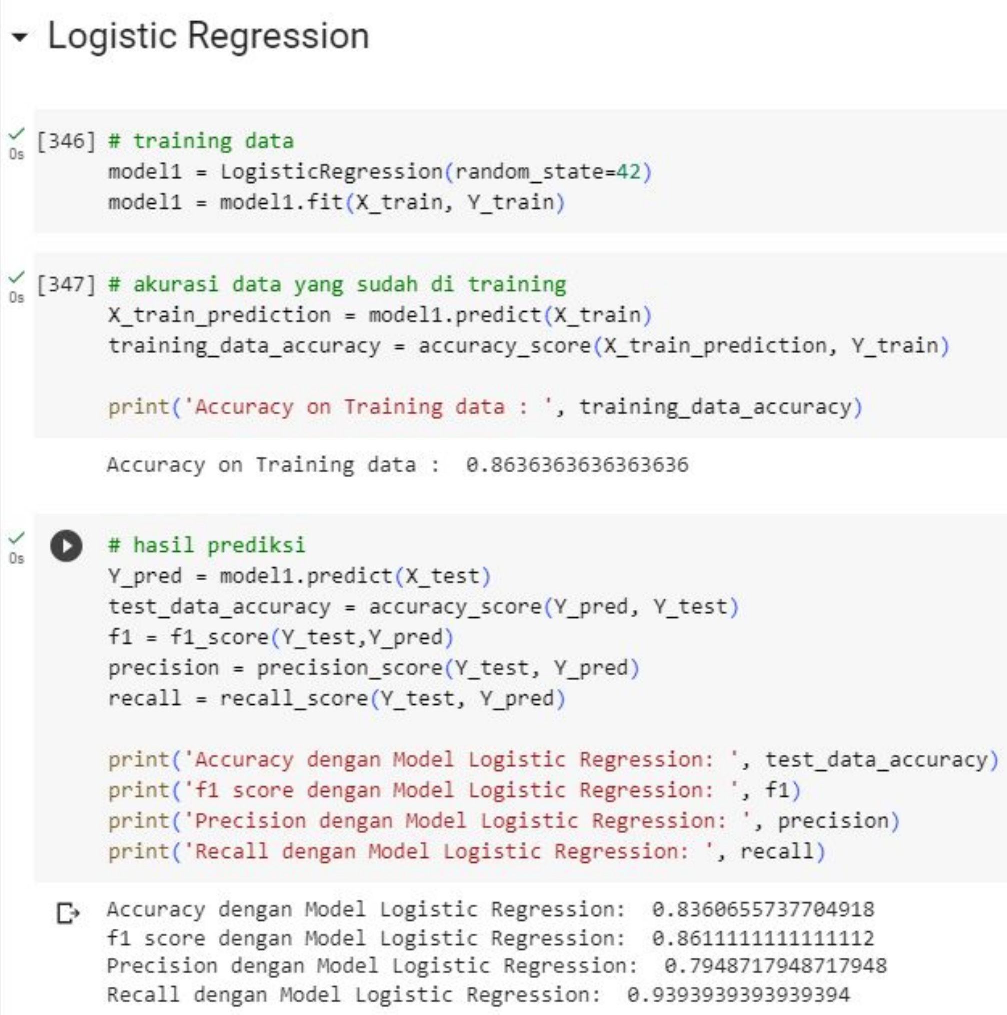


Splitting Data Menjadi Data Train dan Data Test

```
[344] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=42)
```

```
[345] print(X.shape, X_train.shape, X_test.shape)
```

```
      (303, 22) (242, 22) (61, 22)
```



Logistic Regression

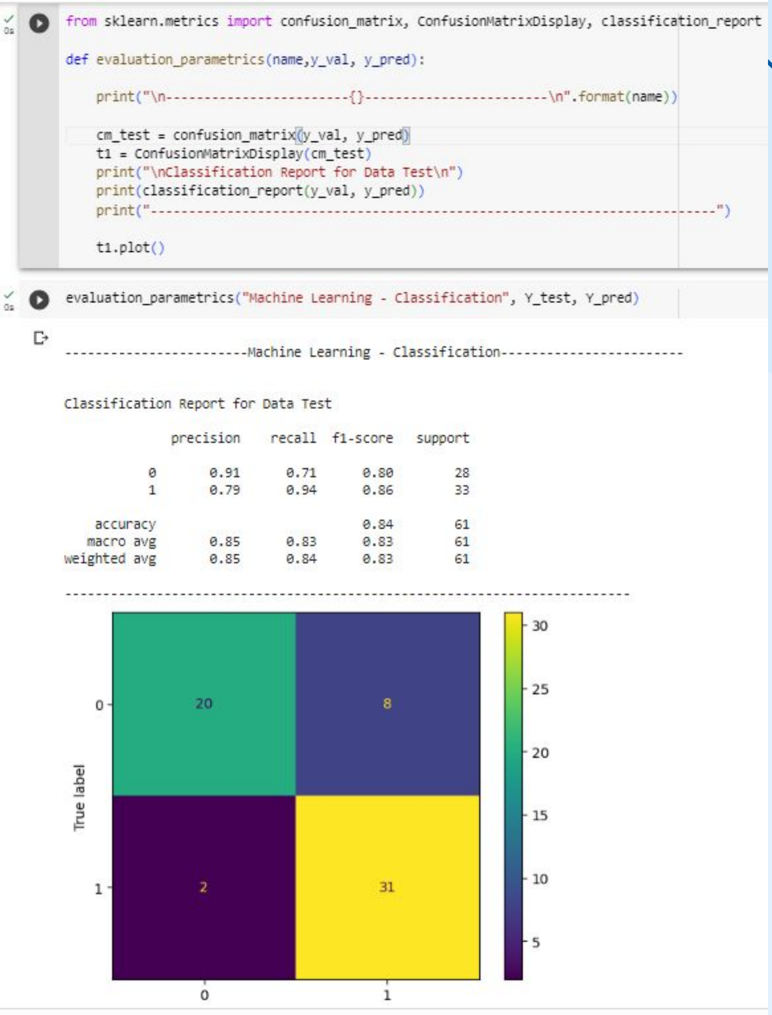
```
[346] # training data
      model1 = LogisticRegression(random_state=42)
      model1 = model1.fit(X_train, Y_train)
```

```
[347] # akurasi data yang sudah di training
      X_train_prediction = model1.predict(X_train)
      training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

      print('Accuracy on Training data : ', training_data_accuracy)
```

```
Accuracy on Training data :  0.8636363636363636
```

```
# hasil prediksi
Y_pred = model1.predict(X_test)
test_data_accuracy = accuracy_score(Y_pred, Y_test)
f1 = f1_score(Y_test,Y_pred)
precision = precision_score(Y_test, Y_pred)
recall = recall_score(Y_test, Y_pred)

print('Accuracy dengan Model Logistic Regression: ', test_data_accuracy)
print('f1 score dengan Model Logistic Regression: ', f1)
print('Precision dengan Model Logistic Regression: ', precision)
print('Recall dengan Model Logistic Regression: ', recall)
```

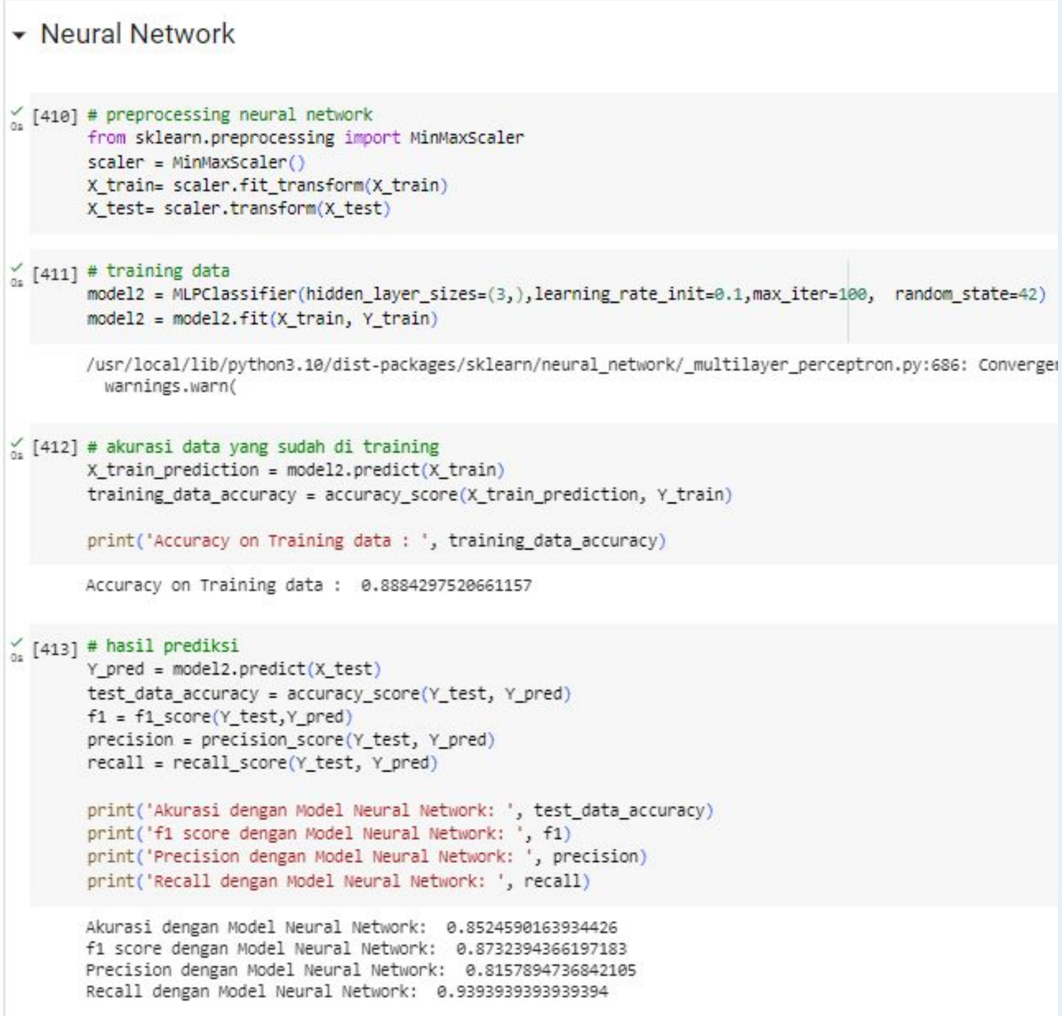
```
Accuracy dengan Model Logistic Regression:  0.8360655737704918
f1 score dengan Model Logistic Regression:  0.8611111111111112
Precision dengan Model Logistic Regression:  0.7948717948717948
Recall dengan Model Logistic Regression:  0.9393939393939394
```



```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report

def evaluation_parametrics(name,y_val, y_pred):

    print("\n----------------------{}----------------------\n".format(name))

    cm_test = confusion_matrix(y_val, y_pred)
    t1 = ConfusionMatrixDisplay(cm_test)
    print("\nClassification Report for Data Test\n")
    print(classification_report(y_val, y_pred))
    print("--------------------------------------------------------------")

    t1.plot()
```

```
evaluation_parametrics("Machine Learning - Classification", Y_test, Y_pred)
```

```
----------------------Machine Learning - Classification----------------------


Classification Report for Data Test

              precision    recall  f1-score   support

           0       0.91      0.71      0.80        28
           1       0.79      0.94      0.86        33

    accuracy                           0.84        61
   macro avg       0.85      0.83      0.83        61
weighted avg       0.85      0.84      0.83        61

--------------------------------------------------------------
```



Neural Network

```
[410] # preprocessing neural network
      from sklearn.preprocessing import MinMaxScaler
      scaler = MinMaxScaler()
      X_train= scaler.fit_transform(X_train)
      X_test= scaler.transform(X_test)
```

```
[411] # training data
      model2 = MLPClassifier(hidden_layer_sizes=(3,),learning_rate_init=0.1,max_iter=100,  random_state=42)
      model2 = model2.fit(X_train, Y_train)
```

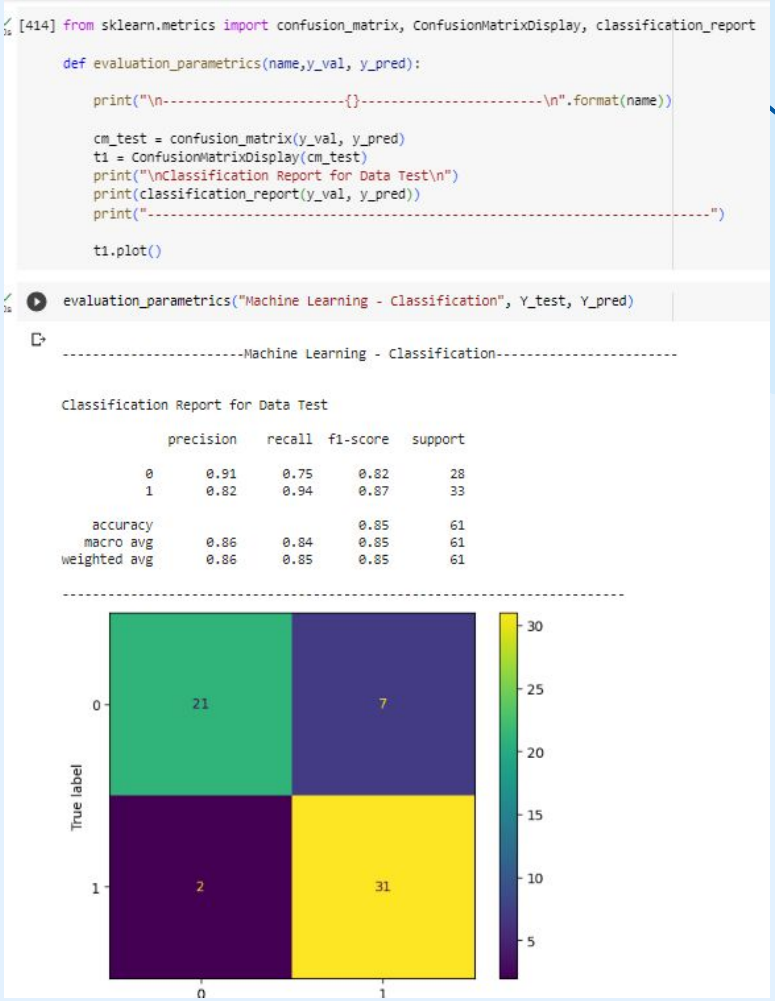
```
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: Converge
  warnings.warn(
```

```
[412] # akurasi data yang sudah di training
      X_train_prediction = model2.predict(X_train)
      training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

      print('Accuracy on Training data : ', training_data_accuracy)
```

```
Accuracy on Training data :  0.8884297520661157
```

```
[413] # hasil prediksi
      Y_pred = model2.predict(X_test)
      test_data_accuracy = accuracy_score(Y_test, Y_pred)
      f1 = f1_score(Y_test,Y_pred)
      precision = precision_score(Y_test, Y_pred)
      recall = recall_score(Y_test, Y_pred)

      print('Akurasi dengan Model Neural Network: ', test_data_accuracy)
      print('f1 score dengan Model Neural Network: ', f1)
      print('Precision dengan Model Neural Network: ', precision)
      print('Recall dengan Model Neural Network: ', recall)
```

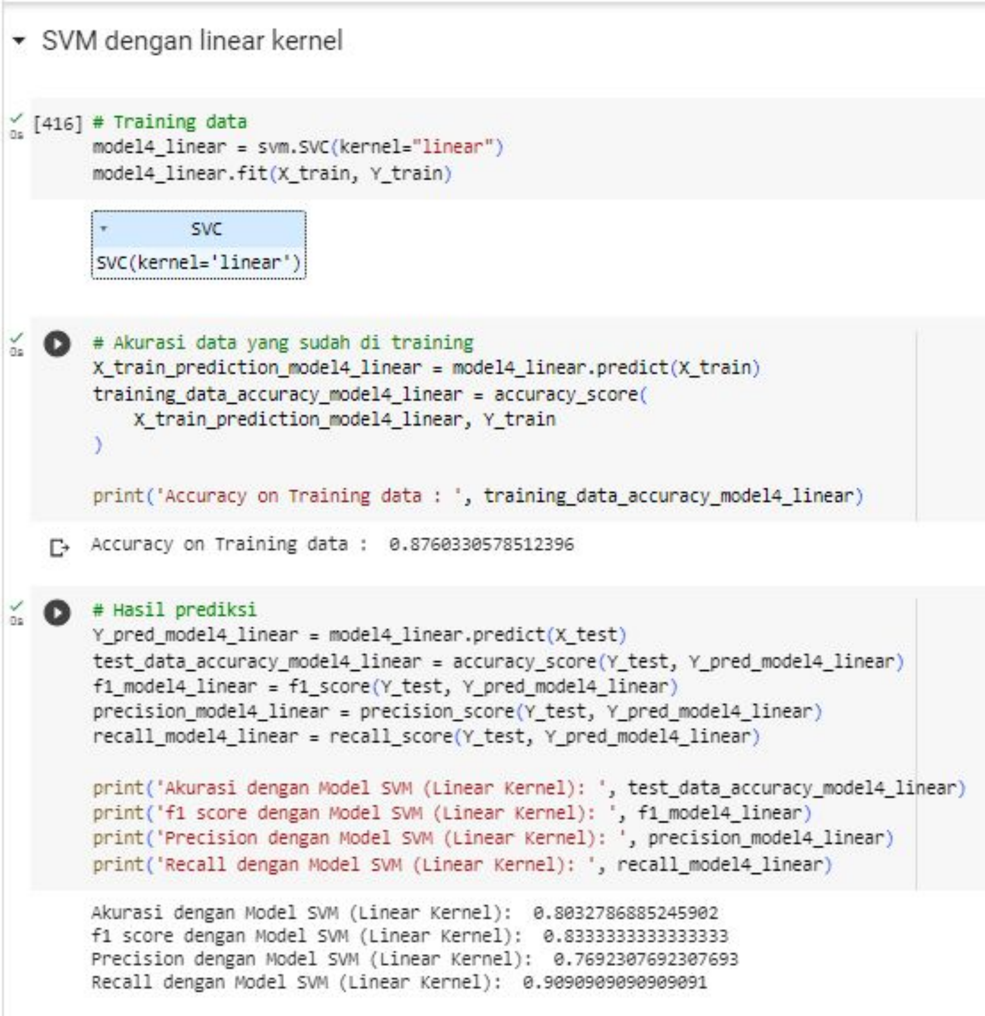
```
Akurasi dengan Model Neural Network:  0.8524590163934426
f1 score dengan Model Neural Network:  0.8732394366197183
Precision dengan Model Neural Network:  0.8157894736842105
Recall dengan Model Neural Network:  0.9393939393939394
```



```
[414] from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report

      def evaluation_parametrics(name,y_val, y_pred):

          print("\n----------------------{}----------------------\n".format(name))

          cm_test = confusion_matrix(y_val, y_pred)
          t1 = ConfusionMatrixDisplay(cm_test)
          print("\nClassification Report for Data Test\n")
          print(classification_report(y_val, y_pred))
          print("---------------------------------------------------------------------------")

          t1.plot()
```

```
evaluation_parametrics("Machine Learning - Classification", Y_test, Y_pred)
```

```
----------------------Machine Learning - Classification----------------------


Classification Report for Data Test

              precision    recall  f1-score   support

           0       0.91      0.75      0.82        28
           1       0.82      0.94      0.87        33

    accuracy                           0.85        61
   macro avg       0.86      0.84      0.85        61
weighted avg       0.86      0.85      0.85        61

---------------------------------------------------------------------
```



SVM dengan linear kernel

```
[416] # Training data
      model4_linear = svm.SVC(kernel="linear")
      model4_linear.fit(X_train, Y_train)
```

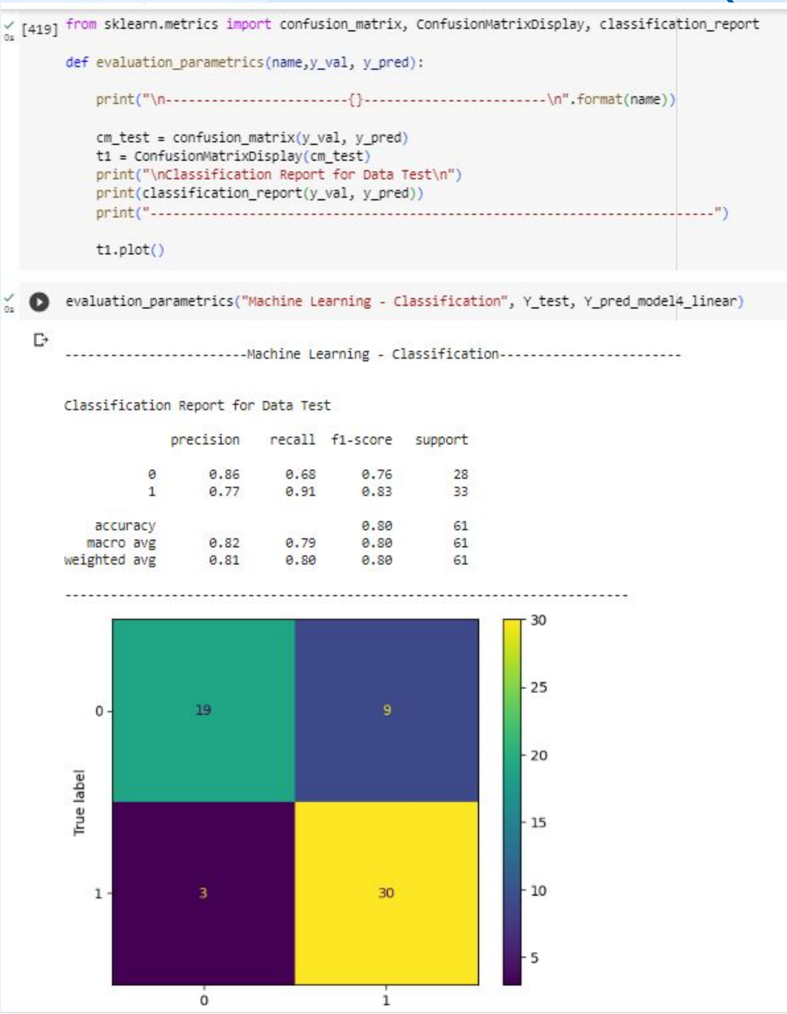
```
SVC
SVC(kernel='linear')
```

```
# Akurasi data yang sudah di training
X_train_prediction_model4_linear = model4_linear.predict(X_train)
training_data_accuracy_model4_linear = accuracy_score(
    X_train_prediction_model4_linear, Y_train
)

print('Accuracy on Training data : ', training_data_accuracy_model4_linear)
```

```
Accuracy on Training data :  0.8760330578512396
```

```
# Hasil prediksi
Y_pred_model4_linear = model4_linear.predict(X_test)
test_data_accuracy_model4_linear = accuracy_score(Y_test, Y_pred_model4_linear)
f1_model4_linear = f1_score(Y_test, Y_pred_model4_linear)
precision_model4_linear = precision_score(Y_test, Y_pred_model4_linear)
recall_model4_linear = recall_score(Y_test, Y_pred_model4_linear)

print('Akurasi dengan Model SVM (Linear Kernel): ', test_data_accuracy_model4_linear)
print('f1 score dengan Model SVM (Linear Kernel): ', f1_model4_linear)
print('Precision dengan Model SVM (Linear Kernel): ', precision_model4_linear)
print('Recall dengan Model SVM (Linear Kernel): ', recall_model4_linear)
```

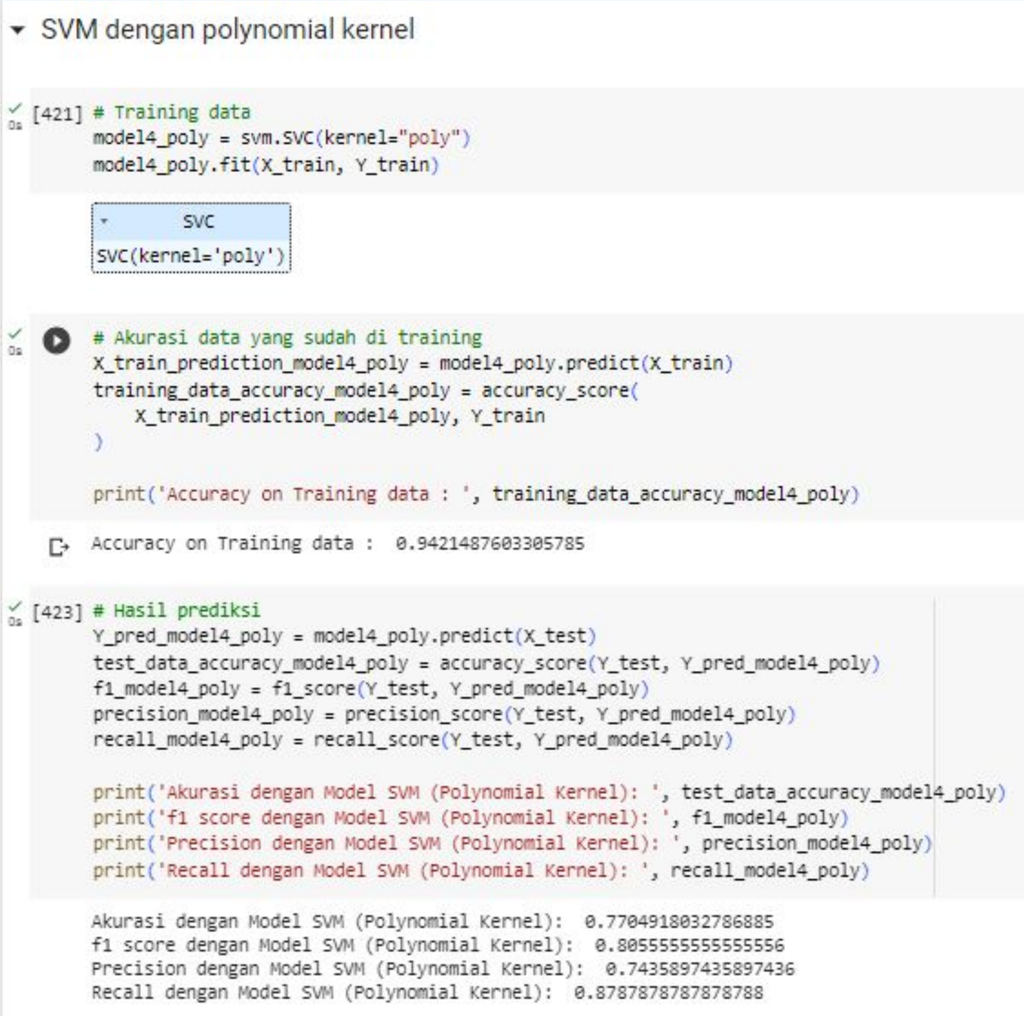
```
Akurasi dengan Model SVM (Linear Kernel):  0.8032786885245902
f1 score dengan Model SVM (Linear Kernel):  0.8333333333333333
Precision dengan Model SVM (Linear Kernel):  0.7692307692307693
Recall dengan Model SVM (Linear Kernel):  0.9090909090909091
```



```
[419] from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report

      def evaluation_parametrics(name,y_val, y_pred):

          print("\n----------------------{}----------------------\n".format(name))

          cm_test = confusion_matrix(y_val, y_pred)
          t1 = ConfusionMatrixDisplay(cm_test)
          print("\nClassification Report for Data Test\n")
          print(classification_report(y_val, y_pred))
          print("--------------------------------------------------------------------------")

          t1.plot()
```

```
evaluation_parametrics("Machine Learning - Classification", Y_test, Y_pred_model4_linear)
```

```
----------------------Machine Learning - Classification----------------------


Classification Report for Data Test

              precision    recall  f1-score   support

           0       0.86      0.68      0.76        28
           1       0.77      0.91      0.83        33

    accuracy                           0.80        61
   macro avg       0.82      0.79      0.80        61
weighted avg       0.81      0.80      0.80        61

--------------------------------------------------------------------------
```



SVM dengan polynomial kernel

```
[421] # Training data
      model4_poly = svm.SVC(kernel="poly")
      model4_poly.fit(X_train, Y_train)
```

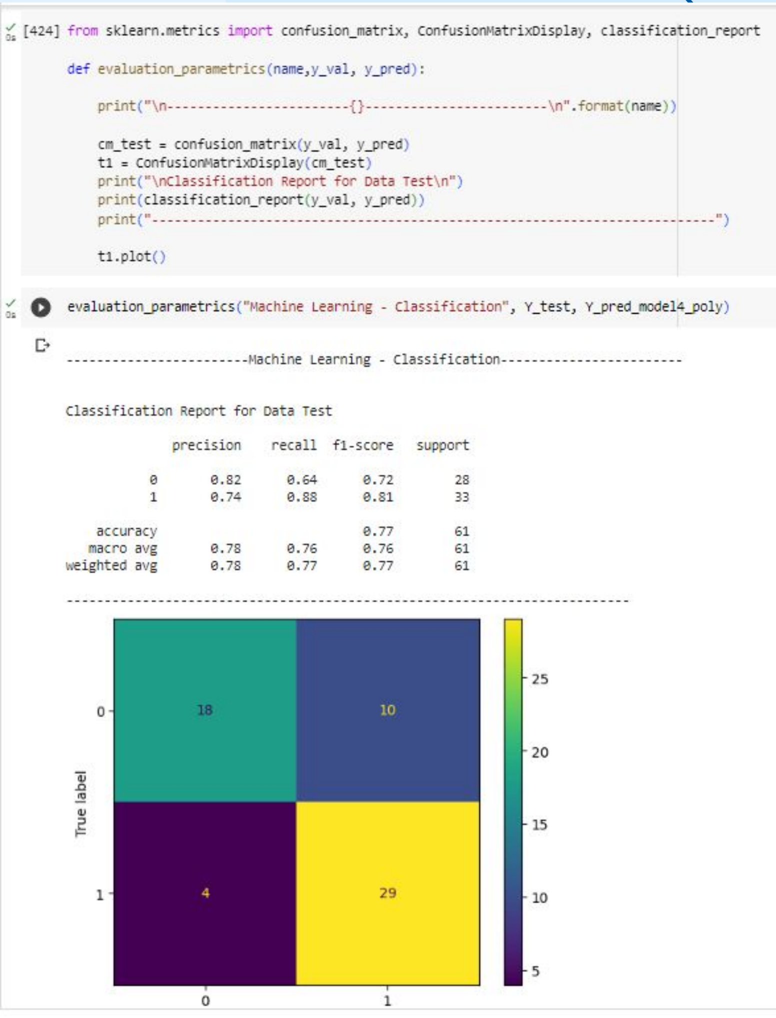
```
SVC
SVC(kernel='poly')
```

```
# Akurasi data yang sudah di training
X_train_prediction_model4_poly = model4_poly.predict(X_train)
training_data_accuracy_model4_poly = accuracy_score(
    X_train_prediction_model4_poly, Y_train
)

print('Accuracy on Training data : ', training_data_accuracy_model4_poly)
```

```
Accuracy on Training data :  0.9421487603305785
```

```
[423] # Hasil prediksi
      Y_pred_model4_poly = model4_poly.predict(X_test)
      test_data_accuracy_model4_poly = accuracy_score(Y_test, Y_pred_model4_poly)
      f1_model4_poly = f1_score(Y_test, Y_pred_model4_poly)
      precision_model4_poly = precision_score(Y_test, Y_pred_model4_poly)
      recall_model4_poly = recall_score(Y_test, Y_pred_model4_poly)

      print('Akurasi dengan Model SVM (Polynomial Kernel): ', test_data_accuracy_model4_poly)
      print('f1 score dengan Model SVM (Polynomial Kernel): ', f1_model4_poly)
      print('Precision dengan Model SVM (Polynomial Kernel): ', precision_model4_poly)
      print('Recall dengan Model SVM (Polynomial Kernel): ', recall_model4_poly)
```

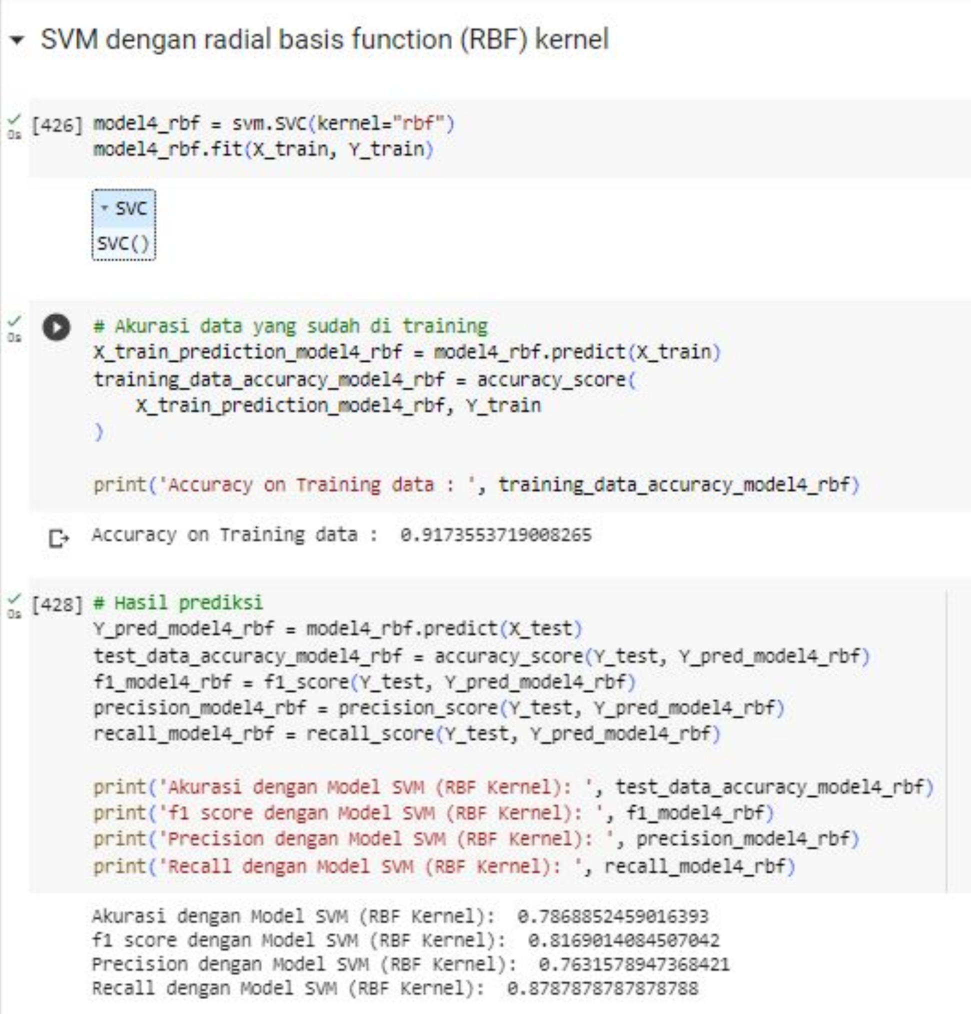
```
Akurasi dengan Model SVM (Polynomial Kernel):  0.7704918032786885
f1 score dengan Model SVM (Polynomial Kernel):  0.8055555555555556
Precision dengan Model SVM (Polynomial Kernel):  0.7435897435897436
Recall dengan Model SVM (Polynomial Kernel):  0.8787878787878788
```



```
[424] from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report

      def evaluation_parametrics(name,y_val, y_pred):

          print("\n----------------------{}----------------------\n".format(name))

          cm_test = confusion_matrix(y_val, y_pred)
          t1 = ConfusionMatrixDisplay(cm_test)
          print("\nClassification Report for Data Test\n")
          print(classification_report(y_val, y_pred))
          print("--------------------------------------------------------------------------------")

          t1.plot()
```

```
evaluation_parametrics("Machine Learning - Classification", Y_test, Y_pred_model4_poly)
```

```
----------------------Machine Learning - Classification----------------------


Classification Report for Data Test

              precision    recall  f1-score   support

           0       0.82      0.64      0.72        28
           1       0.74      0.88      0.81        33

    accuracy                           0.77        61
   macro avg       0.78      0.76      0.76        61
weighted avg       0.78      0.77      0.77        61

--------------------------------------------------------------------------------
```



SVM dengan radial basis function (RBF) kernel

```
[426] model4_rbf = svm.SVC(kernel="rbf")
      model4_rbf.fit(X_train, Y_train)
```

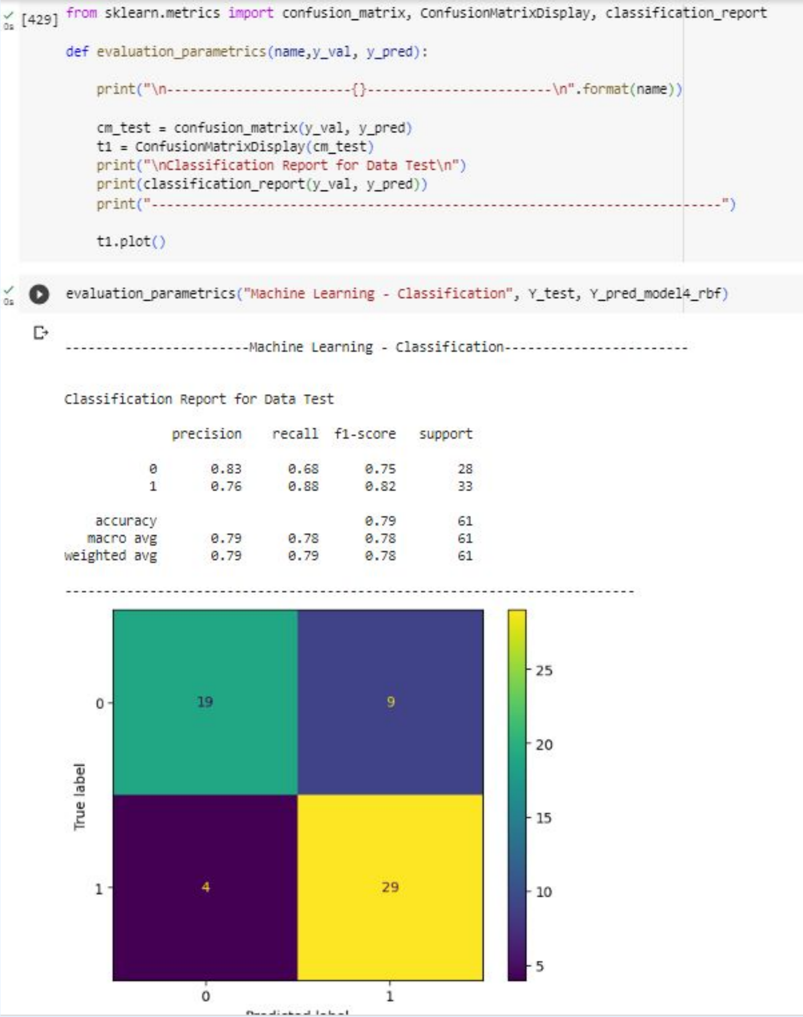
```
▾ SVC
SVC()
```

```
# Akurasi data yang sudah di training
X_train_prediction_model4_rbf = model4_rbf.predict(X_train)
training_data_accuracy_model4_rbf = accuracy_score(
    X_train_prediction_model4_rbf, Y_train
)

print('Accuracy on Training data : ', training_data_accuracy_model4_rbf)
```

```
Accuracy on Training data :  0.9173553719008265
```

```
[428] # Hasil prediksi
      Y_pred_model4_rbf = model4_rbf.predict(X_test)
      test_data_accuracy_model4_rbf = accuracy_score(Y_test, Y_pred_model4_rbf)
      f1_model4_rbf = f1_score(Y_test, Y_pred_model4_rbf)
      precision_model4_rbf = precision_score(Y_test, Y_pred_model4_rbf)
      recall_model4_rbf = recall_score(Y_test, Y_pred_model4_rbf)

      print('Akurasi dengan Model SVM (RBF Kernel): ', test_data_accuracy_model4_rbf)
      print('f1 score dengan Model SVM (RBF Kernel): ', f1_model4_rbf)
      print('Precision dengan Model SVM (RBF Kernel): ', precision_model4_rbf)
      print('Recall dengan Model SVM (RBF Kernel): ', recall_model4_rbf)
```

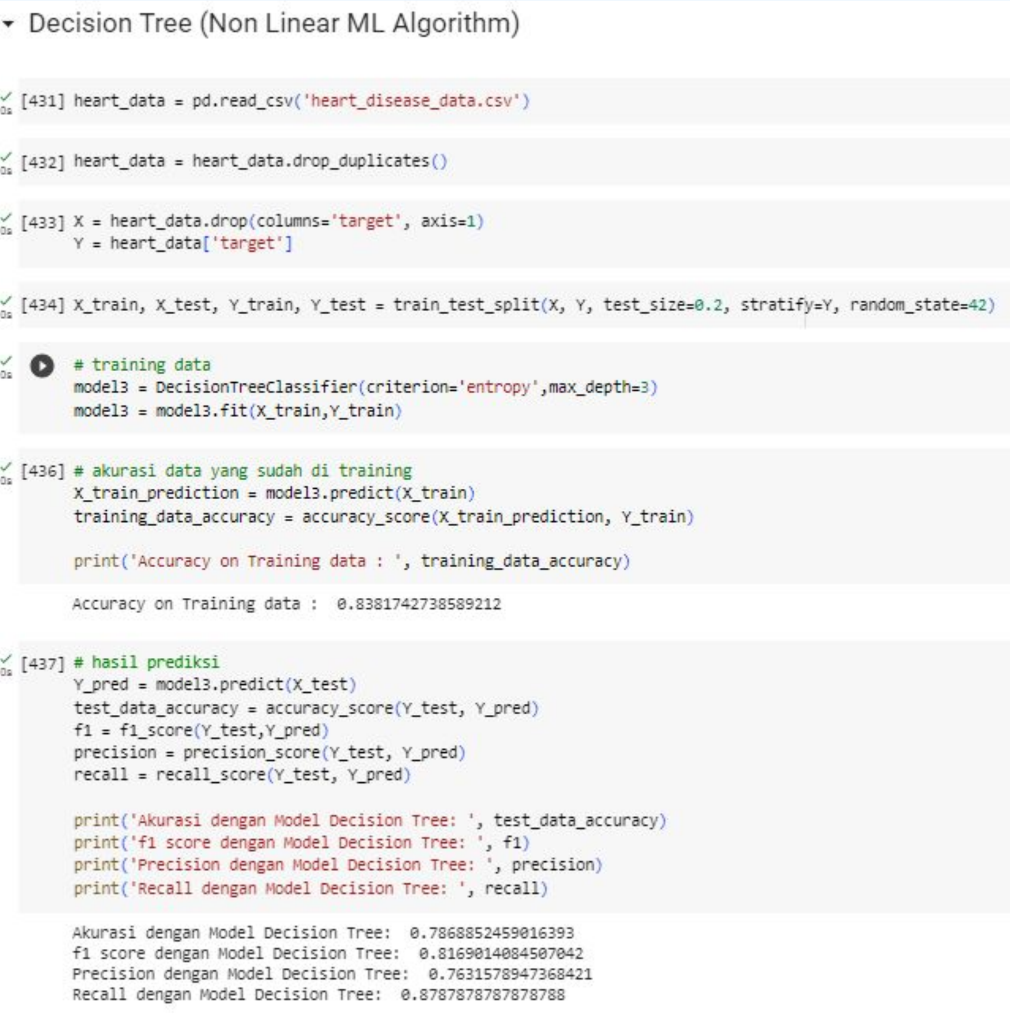
```
Akurasi dengan Model SVM (RBF Kernel):  0.7868852459016393
f1 score dengan Model SVM (RBF Kernel):  0.8169014084507042
Precision dengan Model SVM (RBF Kernel):  0.7631578947368421
Recall dengan Model SVM (RBF Kernel):  0.8787878787878788
```



```
[429] from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report

      def evaluation_parametrics(name,y_val, y_pred):

          print("\n-----------------------{}----------------------\n".format(name))

          cm_test = confusion_matrix(y_val, y_pred)
          t1 = ConfusionMatrixDisplay(cm_test)
          print("\nClassification Report for Data Test\n")
          print(classification_report(y_val, y_pred))
          print("-------------------------------------------------------------------------")

          t1.plot()
```

```
evaluation_parametrics("Machine Learning - Classification", Y_test, Y_pred_model4_rbf)
```

```
-----------------------Machine Learning - Classification-----------------------


Classification Report for Data Test

              precision    recall  f1-score   support

           0       0.83      0.68      0.75        28
           1       0.76      0.88      0.82        33

    accuracy                           0.79        61
   macro avg       0.79      0.78      0.78        61
weighted avg       0.79      0.79      0.78        61

---------------------------------------------------------------------------
```



Decision Tree (Non Linear ML Algorithm)

```
[431] heart_data = pd.read_csv('heart_disease_data.csv')
```

```
[432] heart_data = heart_data.drop_duplicates()
```

```
[433] X = heart_data.drop(columns='target', axis=1)
      Y = heart_data['target']
```

```
[434] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=42)
```

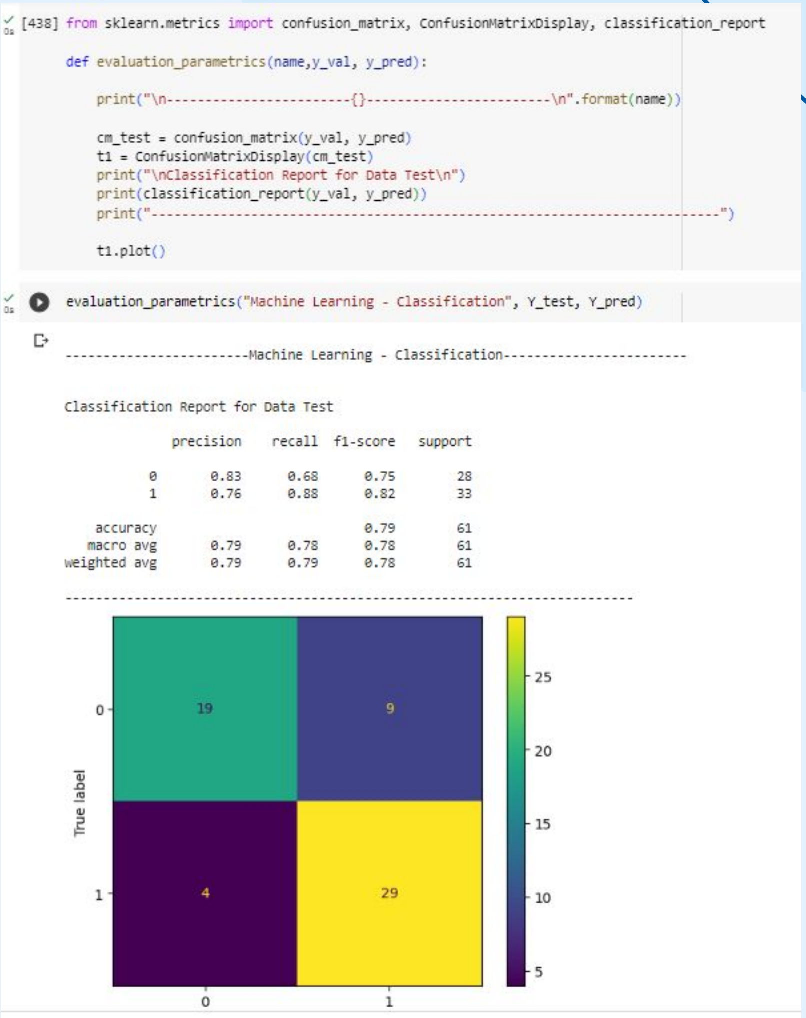
```
# training data
model3 = DecisionTreeClassifier(criterion='entropy',max_depth=3)
model3 = model3.fit(X_train,Y_train)
```

```
[436] # akurasi data yang sudah di training
      X_train_prediction = model3.predict(X_train)
      training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

      print('Accuracy on Training data : ', training_data_accuracy)
```

```
Accuracy on Training data :  0.8381742738589212
```

```
[437] # hasil prediksi
      Y_pred = model3.predict(X_test)
      test_data_accuracy = accuracy_score(Y_test, Y_pred)
      f1 = f1_score(Y_test,Y_pred)
      precision = precision_score(Y_test, Y_pred)
      recall = recall_score(Y_test, Y_pred)

      print('Akurasi dengan Model Decision Tree: ', test_data_accuracy)
      print('f1 score dengan Model Decision Tree: ', f1)
      print('Precision dengan Model Decision Tree: ', precision)
      print('Recall dengan Model Decision Tree: ', recall)
```

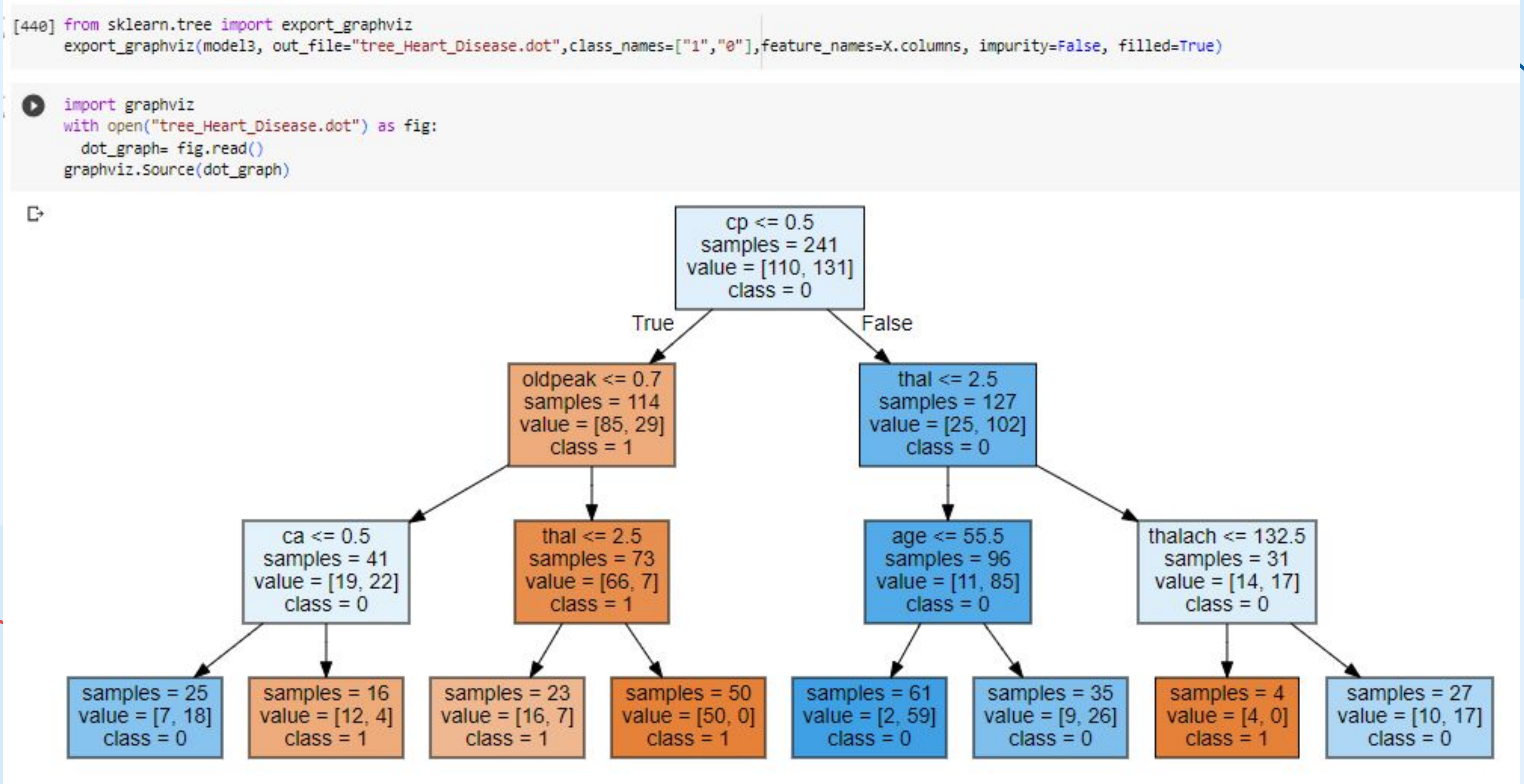
```
Akurasi dengan Model Decision Tree:  0.7868852459016393
f1 score dengan Model Decision Tree:  0.8169014084507042
Precision dengan Model Decision Tree:  0.7631578947368421
Recall dengan Model Decision Tree:  0.8787878787878788
```



```
[438] from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report

      def evaluation_parametrics(name,y_val, y_pred):

          print("\n-----------------------{}-----------------------\n".format(name))

          cm_test = confusion_matrix(y_val, y_pred)
          t1 = ConfusionMatrixDisplay(cm_test)
          print("\nClassification Report for Data Test\n")
          print(classification_report(y_val, y_pred))
          print("--------------------------------------------------------------------------------")

          t1.plot()
```

```
evaluation_parametrics("Machine Learning - Classification", Y_test, Y_pred)
```

```
-----------------------Machine Learning - Classification-----------------------


Classification Report for Data Test

              precision    recall  f1-score   support

           0       0.83      0.68      0.75        28
           1       0.76      0.88      0.82        33

    accuracy                           0.79        61
   macro avg       0.79      0.78      0.78        61
weighted avg       0.79      0.79      0.78        61

--------------------------------------------------------------------------------
```

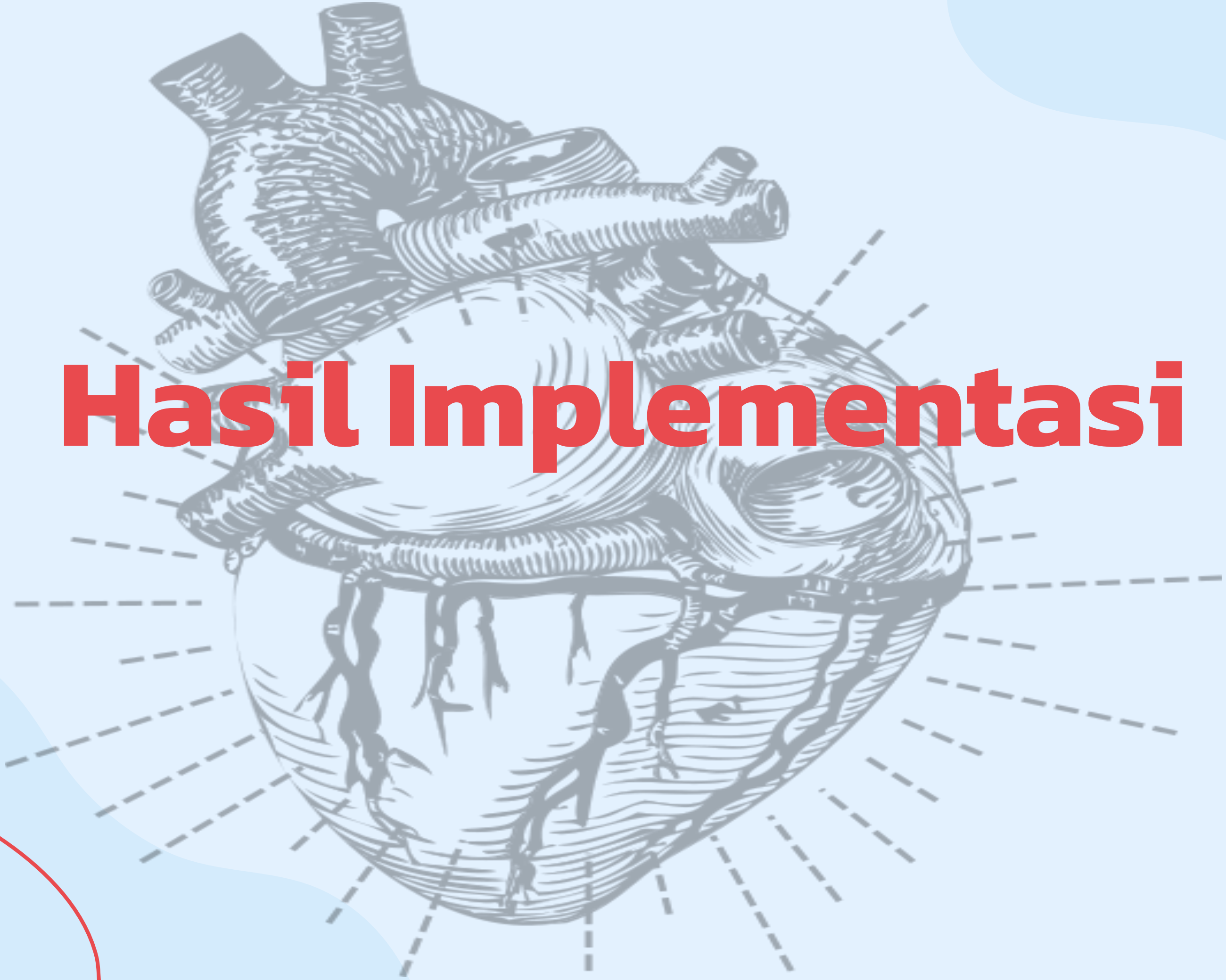


```
[440] from sklearn.tree import export_graphviz
      export_graphviz(model3, out_file="tree_Heart_Disease.dot",class_names=["1","0"],feature_names=X.columns, impurity=False, filled=True)
```

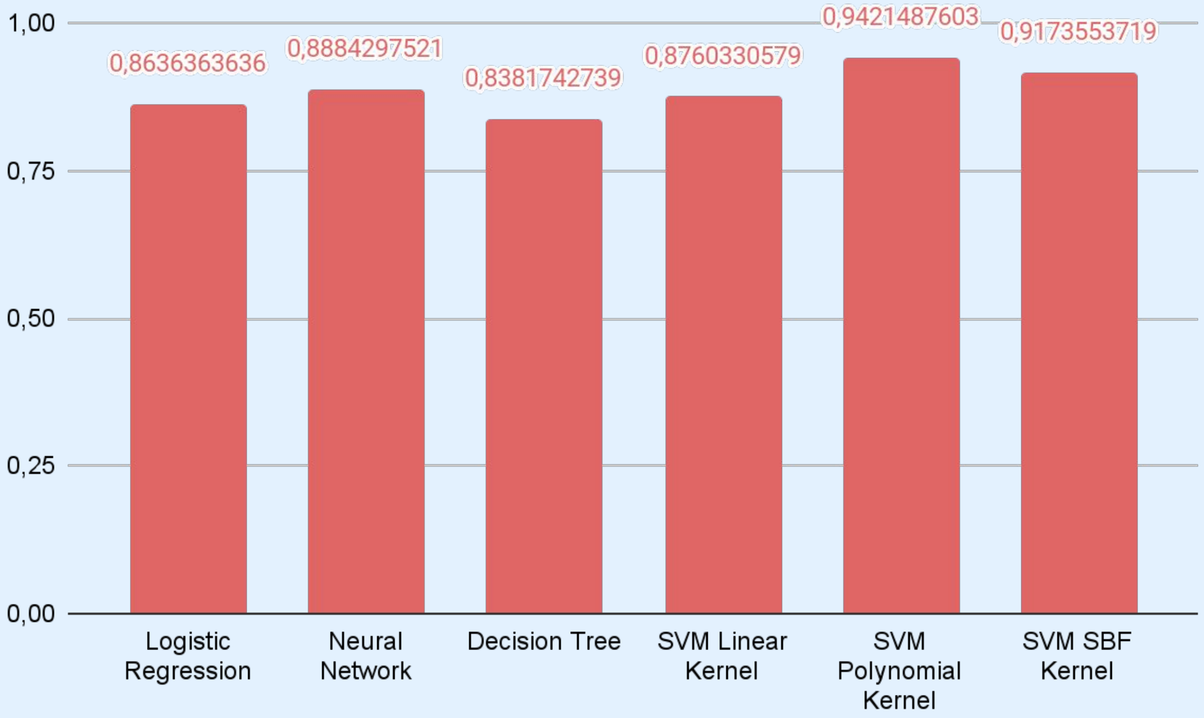
```
import graphviz
with open("tree_Heart_Disease.dot") as fig:
  dot_graph= fig.read()
graphviz.Source(dot_graph)
```

```
cp <= 0.5
samples = 241
value = [110, 131]
class = 0

True

oldpeak <= 0.7
samples = 114
value = [85, 29]
class = 1

False

thal <= 2.5
samples = 127
value = [25, 102]
class = 0

ca <= 0.5
samples = 41
value = [19, 22]
class = 0

thal <= 2.5
samples = 73
value = [66, 7]
class = 1

age <= 55.5
samples = 96
value = [11, 85]
class = 0

thalach <= 132.5
samples = 31
value = [14, 17]
class = 0

samples = 25
value = [7, 18]
class = 0

samples = 16
value = [12, 4]
class = 1

samples = 23
value = [16, 7]
class = 1

samples = 50
value = [50, 0]
class = 1

samples = 61
value = [2, 59]
class = 0

samples = 35
value = [9, 26]
class = 0

samples = 4
value = [4, 0]
class = 1

samples = 27
value = [10, 17]
class = 0
```

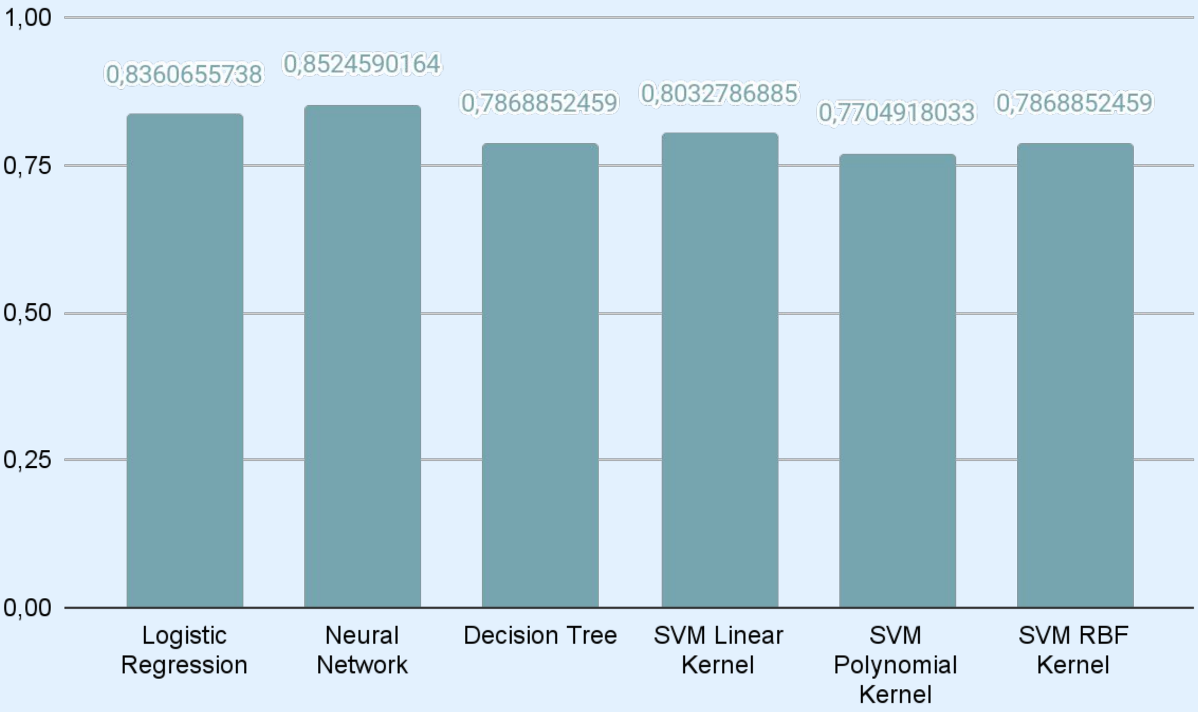
Hasil Implementasi



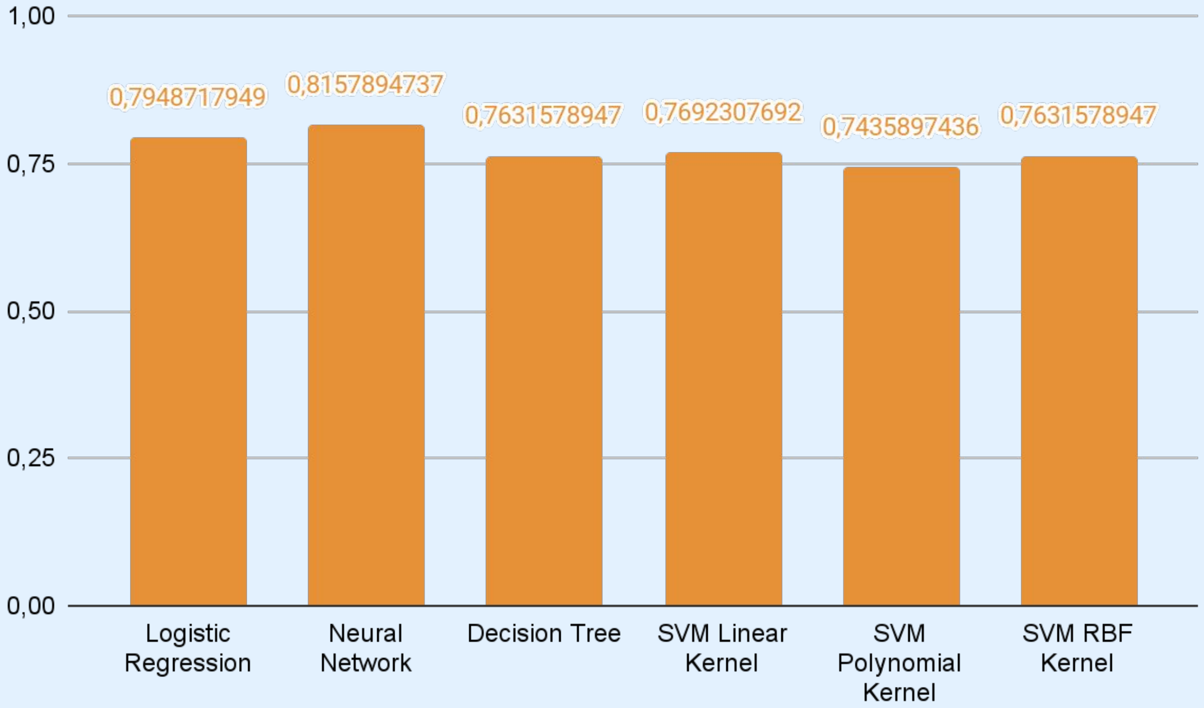
Akurasi Data Training



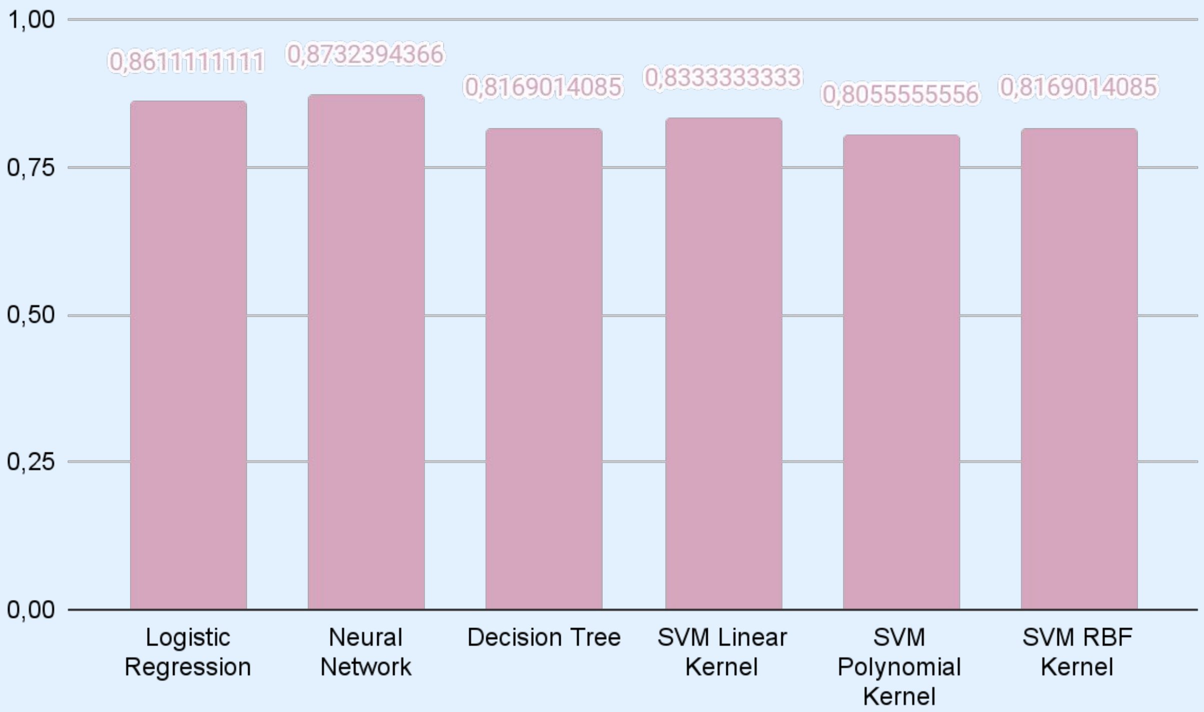
Hasil Prediksi : Akurasi



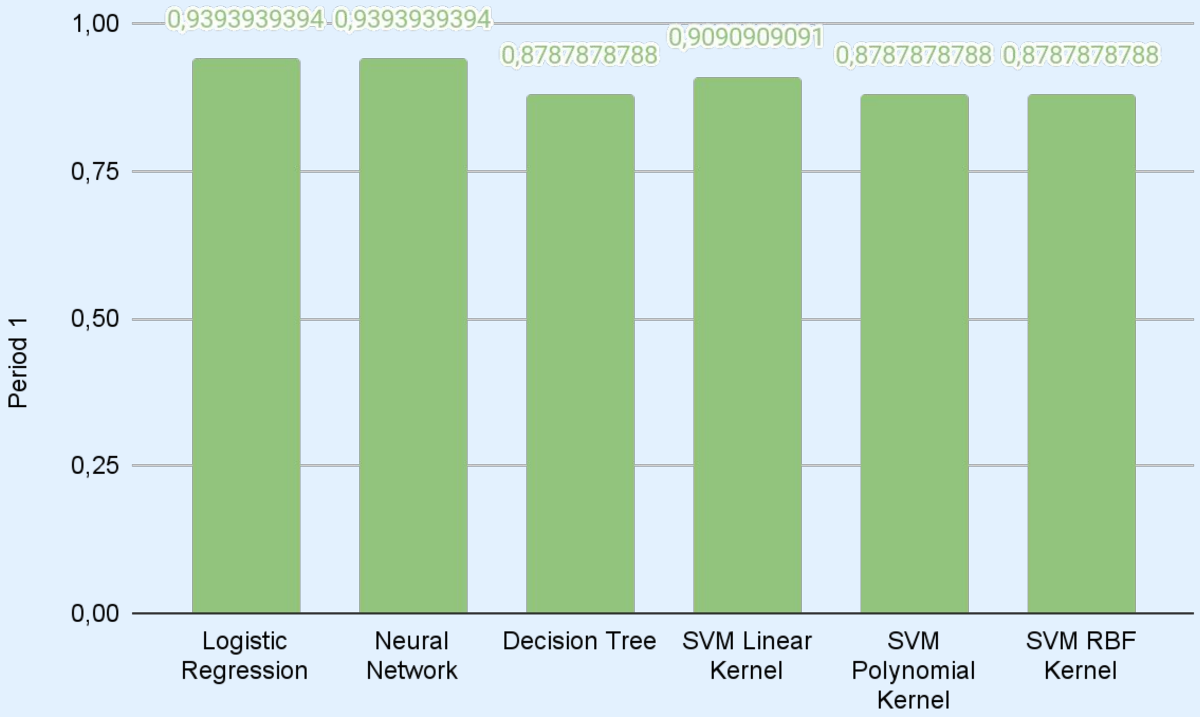
Hasil Prediksi : Presisi



Hasil Prediksi : F1 Score

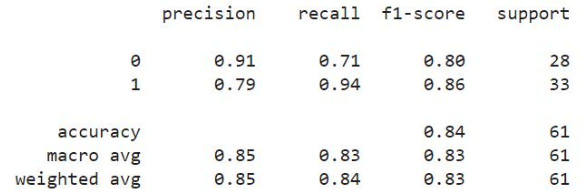


Hasil Prediksi : Recall

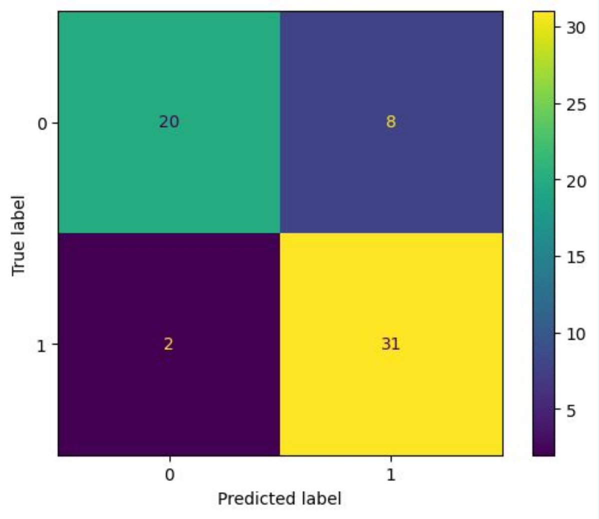


Confusion Matrix

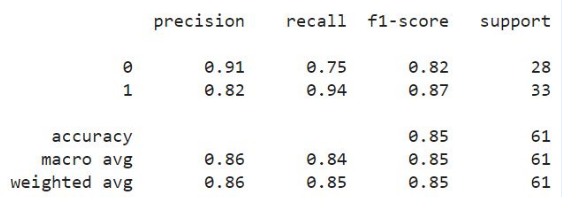
Logistic Regression



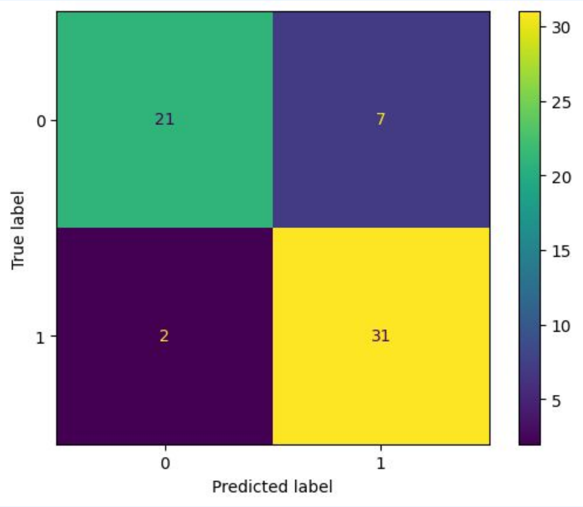
```
              precision    recall  f1-score   support

           0       0.91      0.71      0.80        28
           1       0.79      0.94      0.86        33

    accuracy                           0.84        61
   macro avg       0.85      0.83      0.83        61
weighted avg       0.85      0.84      0.83        61
```



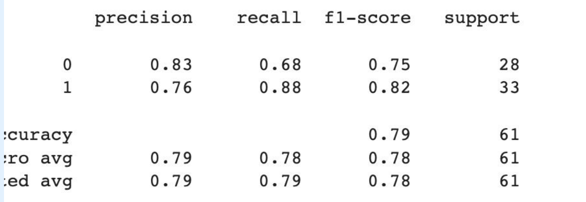
Neural Network



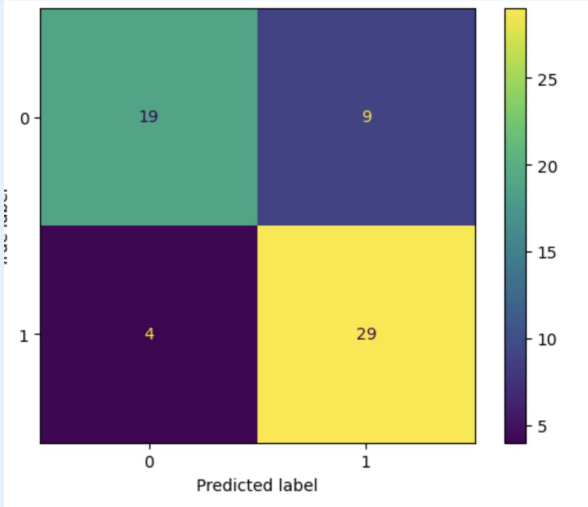
```
              precision    recall  f1-score   support

           0       0.91      0.75      0.82        28
           1       0.82      0.94      0.87        33

    accuracy                           0.85        61
   macro avg       0.86      0.84      0.85        61
weighted avg       0.86      0.85      0.85        61
```



Decision Tree

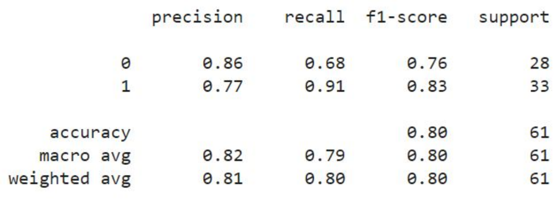


```
        precision    recall  f1-score   support

     0       0.83      0.68      0.75        28
     1       0.76      0.88      0.82        33

ccuracy                          0.79        61
ro avg       0.79      0.78      0.78        61
ted avg      0.79      0.79      0.78        61
```

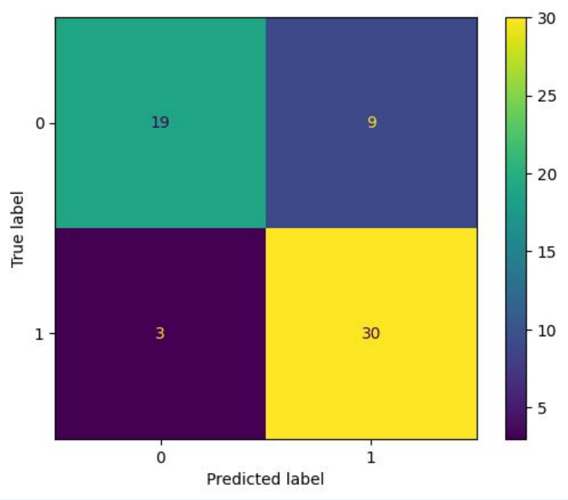


Confusion Matrix

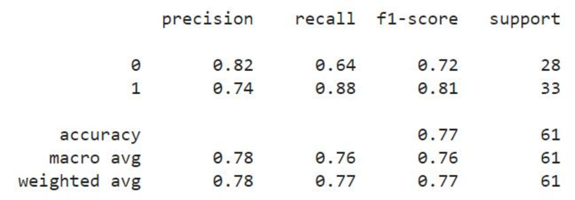
SVM Linear Kernel



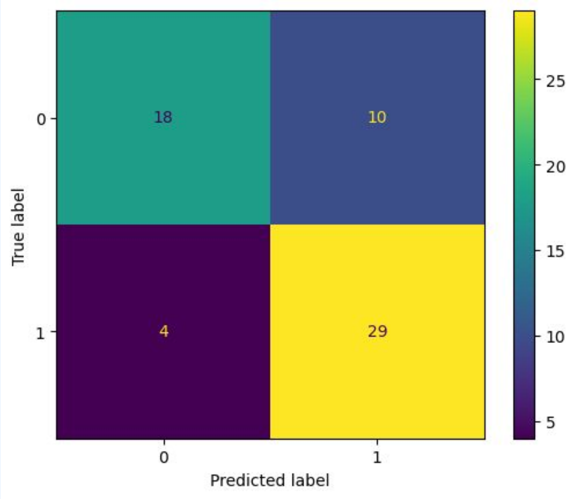
```
              precision    recall  f1-score   support

           0       0.86      0.68      0.76        28
           1       0.77      0.91      0.83        33

    accuracy                           0.80        61
   macro avg       0.82      0.79      0.80        61
weighted avg       0.81      0.80      0.80        61
```



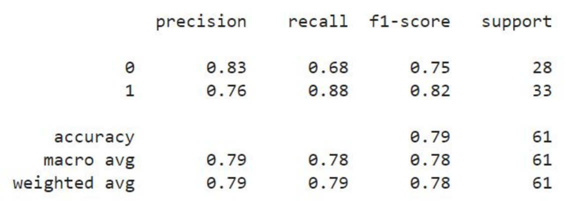
SVM Polynomial Kernel



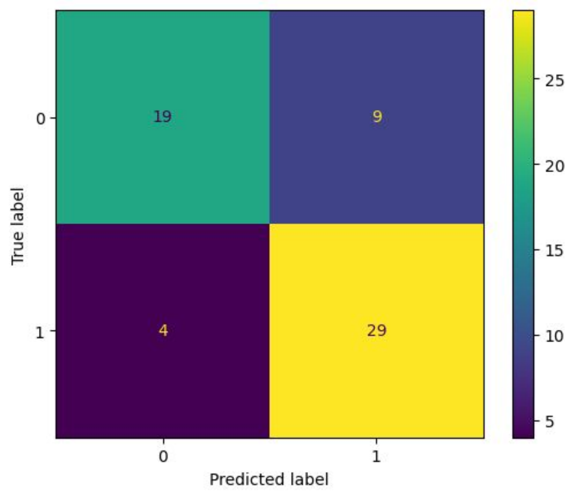
```
              precision    recall  f1-score   support

           0       0.82      0.64      0.72        28
           1       0.74      0.88      0.81        33

    accuracy                           0.77        61
   macro avg       0.78      0.76      0.76        61
weighted avg       0.78      0.77      0.77        61
```

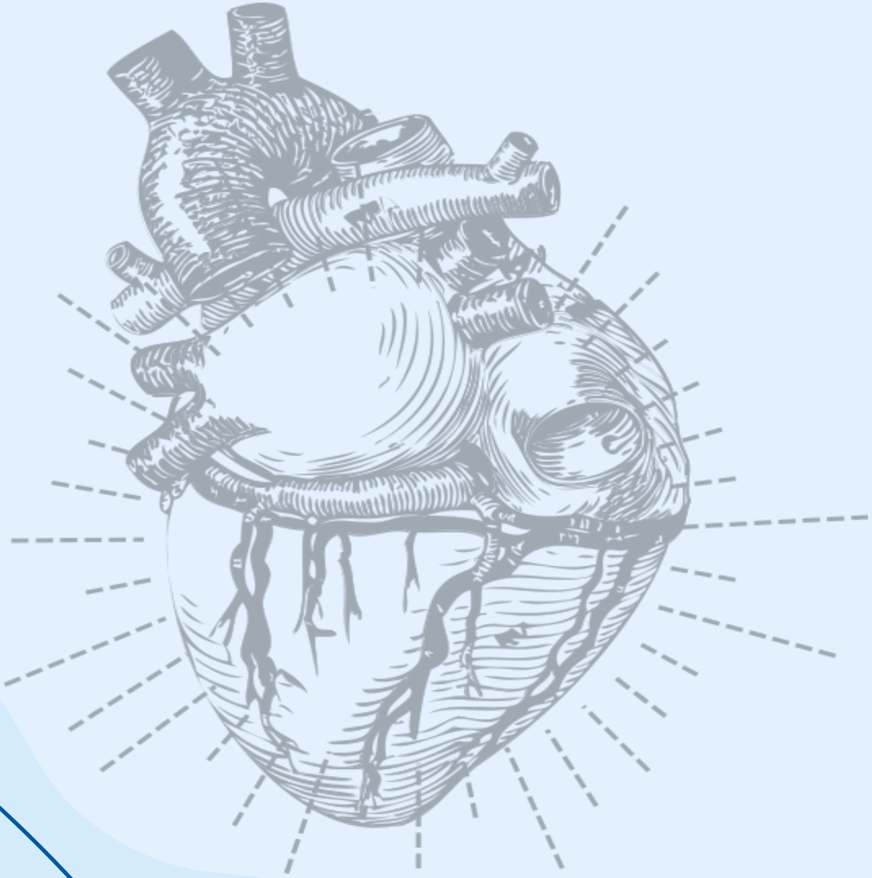


SVM RBF Kernel



```
              precision    recall  f1-score   support

           0       0.83      0.68      0.75        28
           1       0.76      0.88      0.82        33

    accuracy                           0.79        61
   macro avg       0.79      0.78      0.78        61
weighted avg       0.79      0.79      0.78        61
```





04

Kesimpulan dan Saran

Kesimpulan

1. Jenis Nyeri Dada (cp) dan Tingkat Detak Jantung Maksimum (thalach) merupakan variabel yang memiliki keterkaitan paling besar dengan target.
2. Permorfma model
   a. SVM Polynomial Kernel menunjukkan performa yang baik dengan akurasi, f1 score, presisi, dan recall yang tinggi.
   b. Logistic Regression juga menunjukkan performa yang solid dengan akurasi, f1 score, presisi, dan recall yang baik.
3. Pertimbangan model:
   a. Jika kami ingin memprioritaskan akurasi dan keandalan keseluruhan, SVM Polynomial Kernel bisa menjadi pilihan yang baik.
   b. Jika kami mencari model yang lebih sederhana dan mudah diinterpretasikan, Logistic Regression adalah pilihan yang cocok.

Saran

1. Menggunakan data eksternal untuk memperoleh fitur lain yang dapat meningkatkan performa model.
2. Melakukan validasi silang untuk mengkonfirmasi hasil dan mengurangi efek varian acak.

Terima Kasih

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**