Nama : Kurniawan Eko Putra

Kelas : Instrumentasi 5A

NPT : 41210015

Heart Disease Dataset

Tentang Dataset

Dataset Penyakit Jantung diimpor menggunakan pandas, sebuah pustaka Python populer untuk analisis data. Dataset ini terdiri dari 303 baris dan 14 kolom, mencakup berbagai fitur dengan deskripsi sebagai berikut.

- age
- sex
- chest pain type (4 values)
- resting blood pressure
- serum cholestoral in mg/dl
- fasting blood sugar > 120 mg/dl
- resting electrocardiographic results (values 0,1,2)
- maximum heart rate achieved
- exercise induced angina
- oldpeak = ST depression induced by exercise relative to rest
- the slope of the peak exercise ST segment
- number of major vessels (0-3) colored by flourosopy
- thal: 0 = normal; 1 = fixed defect; 2 = reversable defect

Kolom target adalah label yang menandakan kehadiran penyakit jantung, di mana 1 berarti ada penyakit jantung dan 0 berarti tidak ada.

Import Library

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import metrics
from sklearn.metrics import accuracy_score, classification_report
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import numpy as np
```

Load the Heart Disease dataset

```
data = pd.read_csv('input/heart.csv')
```

Explore the dataset and handle any missing values

```
data.head()
```

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
slope  \
0   63    1   3       145   233    1        0      150      0      2.3
0
1   37    1   2       130   250    0        1      187      0      3.5
0
2   41    0   1       130   204    0        0      172      0      1.4
2
3   56    1   1       120   236    0        1      178      0      0.8
2
4   57    0   0       120   354    0        1      163      1      0.6
2

   ca  thal  target
0   0     1       1
1   0     2       1
2   0     2       1
3   0     2       1
4   0     2       1
```

```
data.info()
data.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

```
              age         sex          cp    trestbps        chol
fbs  \
```

```
count  303.000000  303.000000  303.000000  303.000000  303.000000
303.000000
mean    54.366337    0.683168    0.966997  131.623762  246.264026
0.148515
std      9.082101    0.466011    1.032052   17.538143   51.830751
0.356198
min     29.000000    0.000000    0.000000   94.000000  126.000000
0.000000
25%     47.500000    0.000000    0.000000  120.000000  211.000000
0.000000
50%     55.000000    1.000000    1.000000  130.000000  240.000000
0.000000
75%     61.000000    1.000000    2.000000  140.000000  274.500000
0.000000
max     77.000000    1.000000    3.000000  200.000000  564.000000
1.000000

           restecg     thalach       exang     oldpeak       slope
ca  \
count  303.000000  303.000000  303.000000  303.000000  303.000000
303.000000
mean     0.528053  149.646865    0.326733    1.039604    1.399340
0.729373
std      0.525860   22.905161    0.469794    1.161075    0.616226
1.022606
min      0.000000   71.000000    0.000000    0.000000    0.000000
0.000000
25%      0.000000  133.500000    0.000000    0.000000    1.000000
0.000000
50%      1.000000  153.000000    0.000000    0.800000    1.000000
0.000000
75%      1.000000  166.000000    1.000000    1.600000    2.000000
1.000000
max      2.000000  202.000000    1.000000    6.200000    2.000000
4.000000

              thal      target
count  303.000000  303.000000
mean     2.313531    0.544554
std      0.612277    0.498835
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

Cek apakah ada missing value atau tidak

```
data.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Tidak ada nilai yang hilang dalam dataset, yang merupakan keadaan ideal karena ini mengurangi kebutuhan untuk teknik imputasi data yang bisa mempengaruhi akurasi model.

Mengganti nilai yang hilang (jika ada)

Misalnya, jika ada nilai NaN, kita bisa mengganti dengan nilai rata-rata atau median

```
data.fillna(data.mean(), inplace=True)
print(data)
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang
oldpeak  \
0     63    1   3       145   233    1        0      150      0
2.3
1     37    1   2       130   250    0        1      187      0
3.5
2     41    0   1       130   204    0        0      172      0
1.4
3     56    1   1       120   236    0        1      178      0
0.8
4     57    0   0       120   354    0        1      163      1
0.6
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      .
..
298   57    0   0       140   241    0        1      123      1
0.2
299   45    1   3       110   264    0        1      132      0
1.2
300   68    1   0       144   193    1        1      141      0
3.4
301   57    1   0       130   131    0        1      115      1
1.2
302   57    0   1       130   236    0        0      174      0
```

```
0.0

     slope  ca  thal  target
0        0   0     1       1
1        0   0     2       1
2        2   0     2       1
3        2   0     2       1
4        2   0     2       1
..     ...  ..   ...     ...
298      1   0     3       0
299      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0

[303 rows x 14 columns]
```

Preprocess the data

Normalize Features

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

# Normalisasi fitur
scaler = StandardScaler()
features = data.drop('target', axis=1) # Asumsi kolom 'target' adalah
label
features_scaled = scaler.fit_transform(features)
```

Normalisasi dilakukan pada fitur menggunakan `StandardScaler`. Ini penting karena beberapa algoritma pembelajaran mesin, termasuk SVM, sensitif terhadap skala data. Normalisasi membantu meningkatkan kinerja dan stabilitas model. Data dibagi menjadi fitur (X) dan label (y). Kolom `target` adalah label yang kita coba prediksi.

Split the dataset into training and testing sets.

```
# Load your data or ensure 'df' is defined
X = data.drop('target', axis=1)
y = data['target']
# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
# Normalization using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# Print information after preprocessing
```

```
print("Dataset shape after preprocessing:", data.shape)
print("Missing values after preprocessing:\n", data.isnull().sum())
```

```
Dataset shape after preprocessing: (303, 14)
Missing values after preprocessing:
 age         0
sex          0
cp           0
trestbps     0
chol         0
fbs          0
restecg      0
thalach      0
exang        0
oldpeak      0
slope        0
ca           0
thal         0
target       0
dtype: int64
```

Train a classification model -Support Vector Machine

SVM adalah algoritma pembelajaran mesin yang populer untuk klasifikasi. SVM bekerja dengan mencari hiperplane dalam ruang multi-dimensi yang dengan terbaik memisahkan kelas-kelas data yang berbeda. SVM dilatih dengan data pelatihan dan diharapkan bisa memetakan pola yang bisa digunakan untuk memprediksi keberadaan penyakit jantung pada data yang belum diketahui.

```
#Create a svm Classifier
ml = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
ml.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = ml.predict(X_test)
```

Evaluate the model's performance on the testing set

```
# Evaluasi performa model pada set pengujian.

# Melakukan prediksi pada set pengujian
y_pred = ml.predict(X_test)

# Menghitung akurasi dan laporan klasifikasi
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
```

```
print("Akurasi Model: ", accuracy)
print("Laporan Klasifikasi:\n", classification_rep)
```

```
Akurasi Model:  0.8688524590163934
Laporan Klasifikasi:
              precision    recall  f1-score   support

           0       0.86      0.86      0.86        29
           1       0.88      0.88      0.88        32

    accuracy                           0.87        61
   macro avg       0.87      0.87      0.87        61
weighted avg       0.87      0.87      0.87        61
```

Presisi (Precision):

Presisi mengukur seberapa banyak dari kelas yang diprediksi sebagai positif oleh model yang sebenarnya benar-benar positif. Tujuannya adalah untuk mengurangi jumlah false positives.

Recall (Sensitivitas atau Recall): Recall mengukur seberapa banyak dari kelas positif yang sebenarnya diprediksi dengan benar oleh model. Fokus utamanya adalah untuk mengurangi false negatives.

F1-Score: F1-score adalah ukuran yang menggabungkan presisi dan recall ke dalam satu nilai tunggal. Ini adalah harmonic mean dari presisi dan recall. F1-score bermanfaat ketika kita ingin mencari keseimbangan antara presisi dan recall.

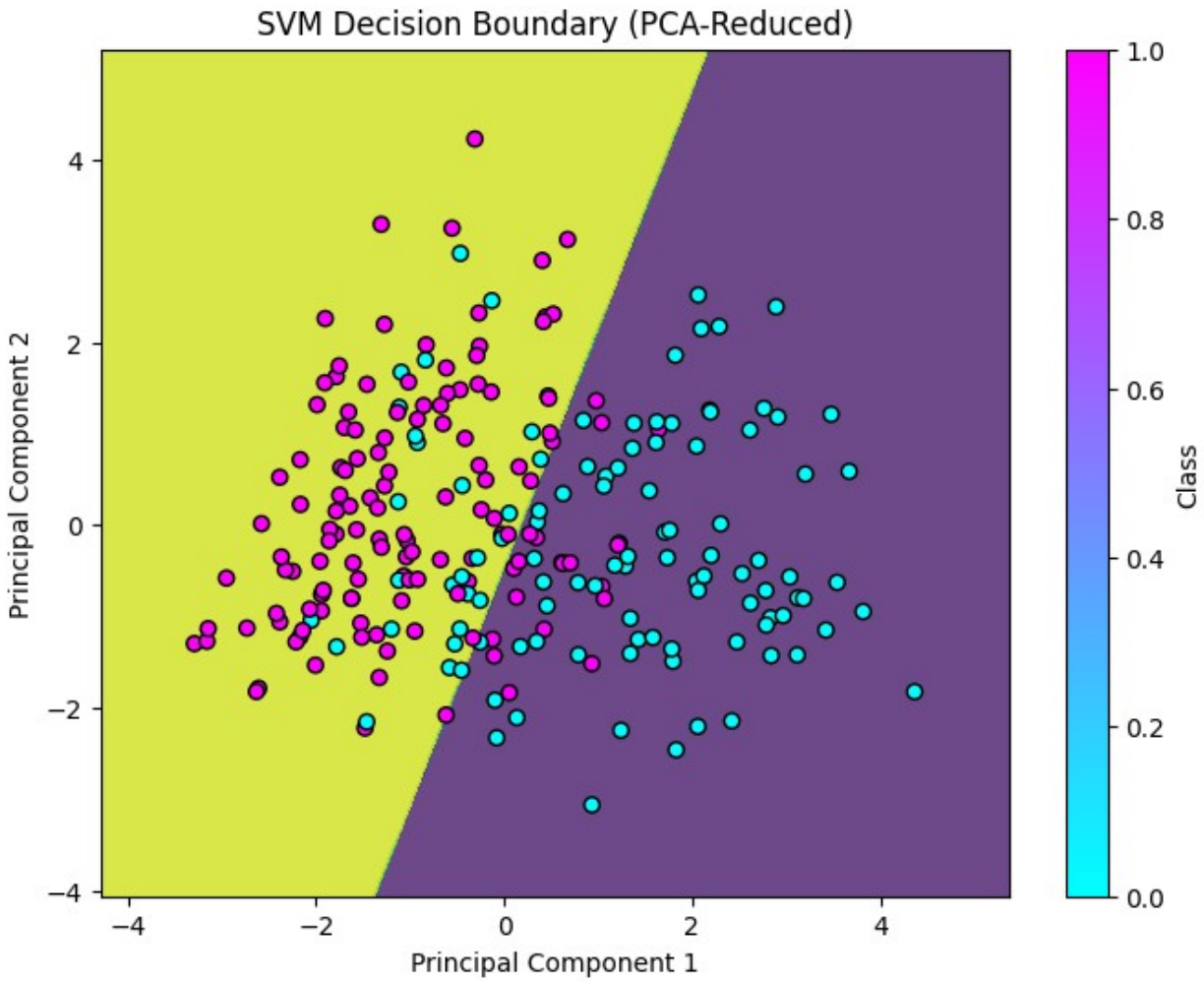
Support: Support adalah jumlah sampel aktual yang termasuk dalam setiap kelas pada set pengujian.

Akurasi (Accuracy): Akurasi adalah rasio prediksi yang benar secara keseluruhan dari jumlah total prediksi. Meskipun berguna, akurasi bisa menyesatkan ketika data tidak seimbang (imbalance) antara kelas-kelas target.

Metrik-metrik ini memberikan pemahaman yang lebih dalam tentang bagaimana model klasifikasi berkinerja terhadap setiap kelas yang diuji. Presisi, recall, dan F1-score penting untuk dipertimbangkan bersamaan untuk mengevaluasi apakah model memiliki kinerja yang baik dalam mengklasifikasikan data. Support menyediakan informasi tentang jumlah sampel yang sesungguhnya miliki kelas tertentu.

```
# Reduksi dimensi menggunakan PCA ke 2 komponen
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

# Inisialisasi model SVM dengan kernel linear
svm_model = SVC(kernel='linear', random_state=42)

# Training model SVM dengan data yang sudah direduksi dimensinya
svm_model.fit(X_train_pca, y_train)
```

```
# Buat mesh grid untuk plot decision boundary
h = 0.02  # Ukuran grid
x_min, x_max = X_train_pca[:, 0].min() - 1, X_train_pca[:, 0].max() +
1
y_min, y_max = X_train_pca[:, 1].min() - 1, X_train_pca[:, 1].max() +
1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min,
y_max, h))
Z = svm_model.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plot decision boundary dan data points
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, alpha=0.8)
plt.scatter(X_train_pca[:, 0], X_train_pca[:, 1], c=y_train,
cmap='cool', edgecolor='black')  # Change colormap here
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('SVM Decision Boundary (PCA-Reduced)')
plt.colorbar(label='Class')
plt.show()
```


SVM Decision Boundary (PCA-Reduced)
Principal Component 2
Principal Component 1
Class
4
2
0
−2
−4
−4
−2
0
2
4
1.0
0.8
0.6
0.4
0.2
0.0

Titanic Dataset

Tentang Dataset

Jenis Data: Data terkait kecelakaan kapal Titanic *Jumlah Fitur:* Termasuk informasi seperti kelas tiket, usia, jenis kelamin, jumlah saudara kandung atau pasangan, jumlah orang tua atau anak-anak, biaya tiket, dan lainnya.

Target: Variabel target umumnya adalah apakah seorang penumpang selamat atau tidak selamat (0 untuk tidak selamat, 1 untuk selamat).

Penamaan Kolom yang digunakan:

1. Passenger ID : Mengidentifikasi penumpang, fitur numerik (IDE Penumpang/Nomor Tiket)
2. Survived : Menjelaskan keselamatan penumpang, Nilai 1 menandakan penumpang selamat dan 0 tidak selamat.
3. Pclass : Kelas Tiket (1 = 1 (Atas), 2 = 2 (Tengah), 3 = 3 (bawah)).
4. Age : Usia dalam tahun
5. Sibsp : Jumlah saudara kandung/pasangan di kapal Titanic
6. Parch : Jumlah orang tua/anak di kapal Titanic
7. Ticket : Nomor tiket
8. Fare : Tarif Penumpang
9. Cabin : Nomor kabin
10. Embarked : Awal naik penumpang ke kapal Titanic (Pelabuhan Embarkasi. C =
11. Cherbourg, Q = Queenstown, S = Southampton)

Import Library

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

Load the Titanic dataset

```
dataTrain = pd.read_csv("input/titanicTrain.csv")
dataTest = pd.read_csv("input/titanicTest.csv")
```

Explore the dataset and handle any missing values

```
dataTrain['Name_Title'] = dataTrain['Name'].apply(lambda x:
x.split(',')[1]).apply(lambda x: x.split()[0])
dataTrain['Name_Title'].value_counts()
```

```
Name_Title
Mr.          517
Miss.        182
Mrs.         125
Master.       40
Dr.            7
Rev.           6
Mlle.          2
Major.         2
Col.           2
the            1
Capt.          1
Ms.            1
Sir.           1
Lady.          1
Mme.           1
Don.           1
Jonkheer.      1
Name: count, dtype: int64
```

```
dataTest['Name_Title'] = dataTest['Name'].apply(lambda x: x.split(',')
[1]).apply(lambda x: x.split()[0])
dataTest['Name_Title'].value_counts()
```

```
Name_Title
Mr.        240
Miss.       78
Mrs.        72
Master.     21
Col.         2
Rev.         2
Ms.          1
Dr.          1
Dona.        1
Name: count, dtype: int64
```

Mencetak 5 Data Pertama

```
dataTrain.head()
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
```

```
4            5         0       3

                                                  Name     Sex   Age
SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0
1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
1
2                              Heikkinen, Miss. Laina  female  26.0
0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
1
4                            Allen, Mr. William Henry    male  35.0
0

   Parch            Ticket     Fare Cabin Embarked Name_Title
0      0         A/5 21171   7.2500   NaN        S        Mr.
1      0          PC 17599  71.2833   C85        C       Mrs.
2      0  STON/O2. 3101282   7.9250   NaN        S      Miss.
3      0            113803  53.1000  C123        S       Mrs.
4      0            373450   8.0500   NaN        S        Mr.
```

```
dataTest.head()
```

```
   PassengerId  Pclass                                          Name
Sex  \
0          892       3                              Kelly, Mr. James
male
1          893       3              Wilkes, Mrs. James (Ellen Needs)
female
2          894       2                      Myles, Mr. Thomas Francis
male
3          895       3                              Wirz, Mr. Albert
male
4          896       3  Hirvonen, Mrs. Alexander (Helga E Lindqvist)
female

    Age  SibSp  Parch   Ticket     Fare Cabin Embarked Name_Title
0  34.5      0      0   330911   7.8292   NaN        Q        Mr.
1  47.0      1      0   363272   7.0000   NaN        S       Mrs.
2  62.0      0      0   240276   9.6875   NaN        Q        Mr.
3  27.0      0      0   315154   8.6625   NaN        S        Mr.
4  22.0      1      1  3101298  12.2875   NaN        S       Mrs.
```

Mencari informasi dan deskripsi dari dataset

```
dataTrain.info()
dataTrain.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
```

```
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
 12  Name_Title   891 non-null    object
dtypes: float64(2), int64(5), object(6)
memory usage: 90.6+ KB
```

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

```
dataTest.info()
dataTest.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Pclass       418 non-null    int64
 2   Name         418 non-null    object
```

```
 3   Sex          418 non-null    object
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object
 10  Embarked     418 non-null    object
 11  Name_Title   418 non-null    object
dtypes: float64(2), int64(4), object(6)
memory usage: 39.3+ KB
```

```
       PassengerId      Pclass         Age       SibSp       Parch
Fare
count   418.000000  418.000000  332.000000  418.000000  418.000000
417.000000
mean   1100.500000    2.265550   30.272590    0.447368    0.392344
35.627188
std     120.810458    0.841838   14.181209    0.896760    0.981429
55.907576
min     892.000000    1.000000    0.170000    0.000000    0.000000
0.000000
25%     996.250000    1.000000   21.000000    0.000000    0.000000
7.895800
50%    1100.500000    3.000000   27.000000    0.000000    0.000000
14.454200
75%    1204.750000    3.000000   39.000000    1.000000    0.000000
31.500000
max    1309.000000    3.000000   76.000000    8.000000    9.000000
512.329200
```

Cek apakah ada missing value atau tidak

```
dataTrain.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
Name_Title       0
dtype: int64
```

Berdasarkan data di atas, dapat diketahui bahwa ada 3 kolom yang memiliki missing value pada dataset training, yakni variabel age, cabin, dan embarked

```
dataTest.isnull().sum()
```

```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
Name_Title       0
dtype: int64
```

Berdasarkan data di atas, dapat diketahui bahwa ada 3 kolom yang memiliki missing value pada dataset testing, yakni variabel age, fare, dan cabin

Handle Missing Value

```
dataTrain.dropna(inplace=True)
dataTest.dropna(inplace=True)
```

Cek kembali apakah masih ada data yang null

```
dataTrain.isnull().sum()
dataTest.isnull().sum()
```

```
PassengerId    0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
Name_Title     0
dtype: int64
```

```
dataTrain.drop(['Name','Ticket', "Cabin"],axis = 1, inplace = True)
dataTest.drop(['Name','Ticket', "Cabin"],axis = 1, inplace = True)
```

Preprocess the data

Normalisasi data

```
label_encoder = LabelEncoder()
dataTrain['Sex'] = label_encoder.fit_transform(dataTrain['Sex'])
dataTrain['Embarked'] =
label_encoder.fit_transform(dataTrain['Embarked'])
dataTrain['Name_Title'] =
label_encoder.fit_transform(dataTrain['Name_Title'])

dataTest['Sex'] = label_encoder.fit_transform(dataTest['Sex'])
dataTest['Embarked'] =
label_encoder.fit_transform(dataTest['Embarked'])
dataTest['Name_Title'] =
label_encoder.fit_transform(dataTest['Name_Title'])

scaler = StandardScaler()
feature_numerik = ["Age","Fare"]
dataTrain[feature_numerik] =
scaler.fit_transform(dataTrain[feature_numerik])
dataTest[feature_numerik] =
scaler.fit_transform(dataTest[feature_numerik])
```

Split the dataset into training and testing sets.

```
X = dataTrain.drop(['Survived'],axis=1)
y = dataTrain['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2,shuffle=True, random_state=5)
```

Train a classification model using Logistic Regression

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
c:\Users\kekop\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning:
lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

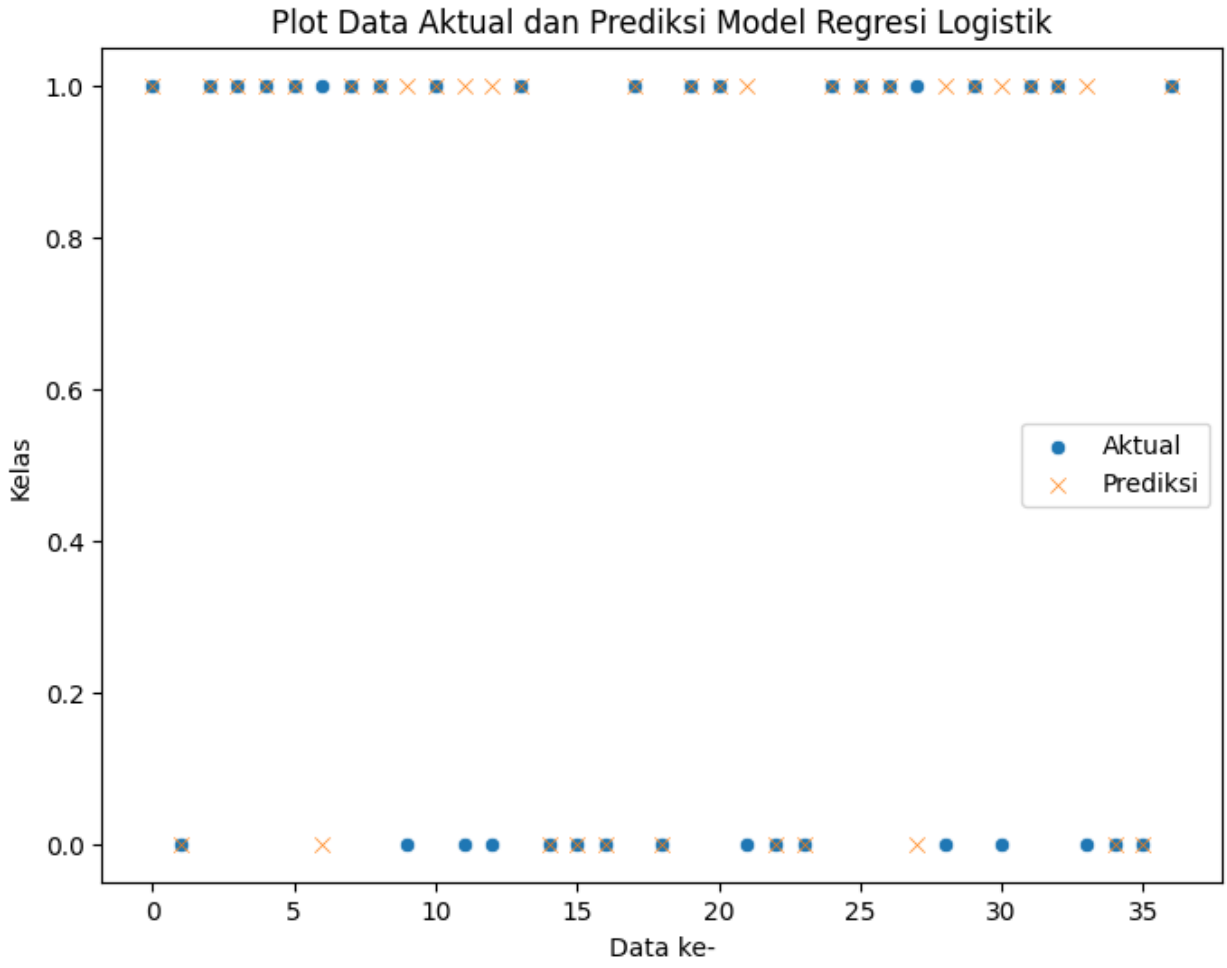
Evaluate the model's performance on the testing set

```
from sklearn import metrics
accuracy = metrics.accuracy_score(y_test, y_pred)
precision = metrics.precision_score(y_test, y_pred)
recall = metrics.recall_score(y_test, y_pred)

# Menampilkan hasil evaluasi
print("Model Regresi Logistik :")
print("Akurasi: {:.2f}".format(accuracy*100), "%")
print("Presisi: {:.2f}".format(precision*100), "%")
print("Sensitivitas: {:.2f}".format(recall*100), "%")
```

```
Model Regresi Logistik :
Akurasi: 75.68 %
Presisi: 73.08 %
Sensitivitas: 90.48 %
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

# Misalkan Anda sudah memiliki X_train, X_test, y_train, y_test yang
sudah disiapkan

# Inisialisasi dan melatih model regresi logistik
model = LogisticRegression()
model.fit(X_train, y_train)

# Melakukan prediksi menggunakan data uji
y_pred = model.predict(X_test)

# Membuat DataFrame dari data aktual dan hasil prediksi
plot_data = pd.DataFrame({'Aktual': y_test, 'Prediksi': y_pred})

# Menambahkan kolom indeks untuk plot
plot_data['Index'] = range(len(plot_data))

# Membuat scatter plot untuk membandingkan data aktual dan hasil
prediksi
plt.figure(figsize=(8, 6))
sns.scatterplot(data=plot_data, x='Index', y='Aktual', label='Aktual',
marker='o')
sns.scatterplot(data=plot_data, x='Index', y='Prediksi',
label='Prediksi', marker='x')
plt.xlabel('Data ke-')
plt.ylabel('Kelas')
plt.title('Plot Data Aktual dan Prediksi Model Regresi Logistik')
plt.legend()
plt.show()
```

```
c:\Users\kekop\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning:
lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```



Kelas 1 : berarti kemungkinan penumpang selamat Kelas 0 : berarti kemungkinan penumpang tidak selamat *Jika ada tanda aktual dan prediksi pada 1 plot, maka berarti prediksi bernilai benar

Latihan 3 - Dataset Mushrooms

Tentang Dataset

Jenis Data: Data terkait dengan karakteristik fisik dan kimia jamur *Jumlah Fitur:* Fitur-fitur dalam dataset mencakup berbagai atribut seperti warna, bentuk, ukuran, dan aroma jamur. *Target:* Variabel target menunjukkan apakah jamur tersebut dapat dimakan (edible) atau beracun (poisonous). *Penamaan Kolom yang digunakan:*

1. class : kelas: dapat dimakan = e, beracun = p
2. cap.shape : bentuk topi: bel = b, kerucut = c, cembung = x, datar = f, knobbed = k, cekung = s
3. cap.surface : tutup-permukaan : berserat = f, alur = g, bersisik = y, halus = s
4. cap.color : warna topi : coklat = n, buff = b, kayu manis = c, abu-abu = g, hijau = r, pink p, ungu = u, merah = e, putih = w, kuning = y
5. bruises : memar = t, no = f
6. odor : bau : almond = a, adas = l, creosote = c, amis = y, busuk = f, apak = m, tidak ada = n, pedas = p, pedas = s
7. gill.attachment : lampiran-insang : terlampir = a, turun = d, bebas = f, berlekuk = n
8. gill.spacing : jarak insang : dekat = c, ramai = w, jauh = d
9. gill.size : ukuran insang : luas = b, sempit = n
10. gill.color : warna insang : hitam = k, coklat = n, buff = b, coklat = h, abu-abu = g, hijau = r, oranye = o, merah muda = p, ungu = u, merah = e, putih = w, kuning = y
11. stalk.shape :tangkai-bentuk: memperbesar = e, meruncing = t
12. stalk.root :root-root: bulbous = b, club = c, cup = u, sama = e, rhizomorphs = z,
13. rooted= r, missing =?
14. stalk.surface.above.ring:tangkai-permukaan-atas-cincin: berserat = f, bersisik = y, halus = k, halus = s
15. stalk.surface.below.ring:tangkai-permukaan-di bawah-cincin: berserat = f, bersisik = y,halus = k, halus = s
16. veil.type :tipe kerudung: sebagian = p, universal = u
17. veil.color :kerudung-warna: coklat = n, oranye = o, putih = w, kuning = y
18. ring.number :dering-angka: tidak ada = n, satu = o, dua = t
19. ring.type :tipe cincin: jaring laba-laba = c, evanescent = e, flaring = f, besar = l, tidak ada = n, liontin = p, selubung = s, zona = z
20. spore.print.color :spora-cetak-warna: hitam = k, coklat = n, buff = b, coklat = h, hijau = r, oranye = o, ungu = u, putih = w, kuning = y
21. population :populasi: berlimpah = a, berkerumun = c, banyak = n, tersebar = s,
22. beberapa = v, soliter = y
23. habitat :habitat: rumput = g, daun = l, padang rumput = m, jalur = p, perkotaan = u,
24. limbah = w, kayu = d

Import Library

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
plt.style.use('ggplot')
pd.set_option('display.max_columns', 200)
from sklearn.tree import plot_tree
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
```

Load the Mushroom dataset

```
data = pd.read_csv('input/mushrooms.csv')
```

Explore the dataset and handle any missing values

Mencetak 5 data pertama

```
data.head()
```

```
  class cap-shape cap-surface cap-color bruises odor gill-
attachment  \
0     p         x           s         n       t    p                f

1     e         x           s         y       t    a                f

2     e         b           s         w       t    l                f

3     p         x           y         w       t    p                f

4     e         x           s         g       f    n                f


  gill-spacing gill-size gill-color stalk-shape stalk-root  \
0            c         n          k           e          e
1            c         b          k           e          c
2            c         b          n           e          c
3            c         n          n           e          e
4            w         b          k           t          e

  stalk-surface-above-ring stalk-surface-below-ring stalk-color-above-
ring  \
0                        s                        s
w
1                        s                        s
w
2                        s                        s
```

```
w
3                         s                         s
w
4                         s                         s
w

  stalk-color-below-ring veil-type veil-color ring-number ring-type  \
0                      w         p          w           o         p
1                      w         p          w           o         p
2                      w         p          w           o         p
3                      w         p          w           o         p
4                      w         p          w           o         e

  spore-print-color population habitat
0                 k          s       u
1                 n          n       g
2                 n          n       m
3                 k          s       u
4                 n          a       g
```

Mencari informasi dan deskripsi dari dataset

```
data.info()
data.describe()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8124 entries, 0 to 8123
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   class                     8124 non-null   object
 1   cap-shape                 8124 non-null   object
 2   cap-surface               8124 non-null   object
 3   cap-color                 8124 non-null   object
 4   bruises                   8124 non-null   object
 5   odor                      8124 non-null   object
 6   gill-attachment           8124 non-null   object
 7   gill-spacing              8124 non-null   object
 8   gill-size                 8124 non-null   object
 9   gill-color                8124 non-null   object
 10  stalk-shape               8124 non-null   object
 11  stalk-root                8124 non-null   object
 12  stalk-surface-above-ring  8124 non-null   object
 13  stalk-surface-below-ring  8124 non-null   object
 14  stalk-color-above-ring    8124 non-null   object
 15  stalk-color-below-ring    8124 non-null   object
 16  veil-type                 8124 non-null   object
 17  veil-color                8124 non-null   object
 18  ring-number               8124 non-null   object
 19  ring-type                 8124 non-null   object
```

```
 20  spore-print-color         8124 non-null   object
 21  population                8124 non-null   object
 22  habitat                   8124 non-null   object
dtypes: object(23)
memory usage: 1.4+ MB

       class cap-shape cap-surface cap-color bruises  odor gill-
attachment  \
count   8124      8124        8124      8124    8124  8124
8124
unique     2         6           4        10       2     9
2
top        e         x           y         n       f     n
f
freq    4208      3656        3244      2284    4748  3528
7914

       gill-spacing gill-size gill-color stalk-shape stalk-root  \
count          8124      8124       8124        8124       8124
unique            2         2         12           2          5
top               c         b          b           t          b
freq           6812      5612       1728        4608       3776

       stalk-surface-above-ring stalk-surface-below-ring  \
count                      8124                     8124
unique                        4                        4
top                           s                        s
freq                       5176                     4936

       stalk-color-above-ring stalk-color-below-ring veil-type veil-
color  \
count                    8124                    8124      8124
8124
unique                      9                       9         1
4
top                         w                       w         p
w
freq                     4464                    4384      8124
7924

       ring-number ring-type spore-print-color population habitat
count         8124      8124              8124       8124    8124
unique           3         5                 9          6       7
top              o         p                 w          v       d
freq          7488      3968              2388       4040    3148
```

Cek apakah ada kolom yang null atau tidak

```
data.isna().sum()
data.isnull().sum()
```

```
class                       0
cap-shape                   0
cap-surface                 0
cap-color                   0
bruises                     0
odor                        0
gill-attachment             0
gill-spacing                0
gill-size                   0
gill-color                  0
stalk-shape                 0
stalk-root                  0
stalk-surface-above-ring    0
stalk-surface-below-ring    0
stalk-color-above-ring      0
stalk-color-below-ring      0
veil-type                   0
veil-color                  0
ring-number                 0
ring-type                   0
spore-print-color           0
population                  0
habitat                     0
dtype: int64
```

Preprocess the data

Encode Categorical Variables

```
label_encoder = LabelEncoder()

for col in data.columns:
    data[str(col)] = label_encoder.fit_transform(data[str(col)])

scaler = MinMaxScaler()
for col in data.columns:
    col_data = data[str(col)].values.reshape(-1, 1)
    data[str(col)] = scaler.fit_transform(col_data)
```

Split the dataset into training and testing sets

```
X = data.drop('class',axis='columns')
y = data['class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.2, random_state = 1,shuffle=True)
```

Train a classification model using Random Forest

```
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_predict = model.predict(X_test)
```

Evaluate the model's performance on the testing sets

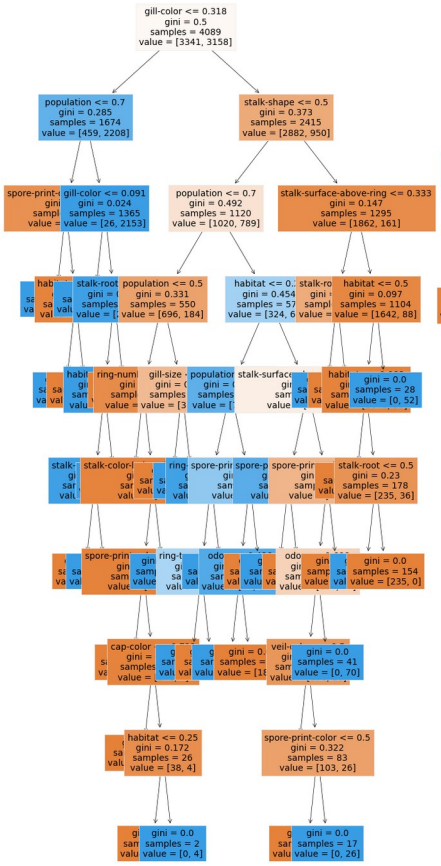
```
accuracy = accuracy_score(y_test, y_predict)
conf_matrix = confusion_matrix(y_test, y_predict)
class_report = classification_report(y_test, y_predict)

print("Akurasi: {:.2f}".format(accuracy*100),"%")
print("Confusion Matrix:")
print(conf_matrix)
print("Evaluasi Model:")
print(class_report)
```

```
Akurasi: 100.00 %
Confusion Matrix:
[[820   0]
 [  0 805]]
Evaluasi Model:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       820
         1.0       1.00      1.00      1.00       805

    accuracy                           1.00      1625
   macro avg       1.00      1.00      1.00      1625
weighted avg       1.00      1.00      1.00      1625
```

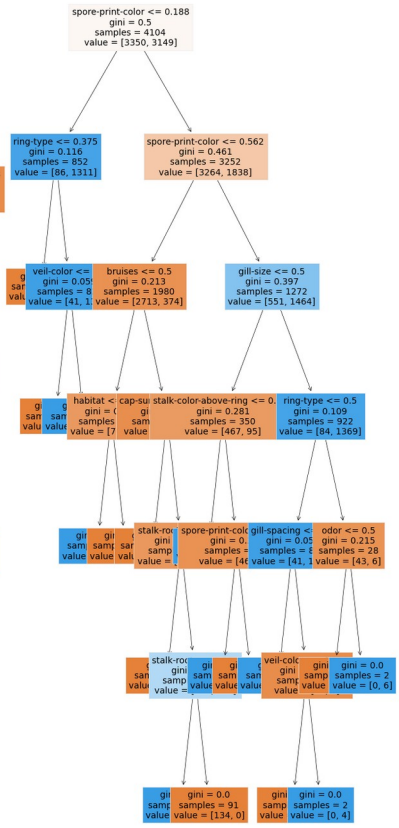
```
num_trees_to_visualize = 3
plt.figure(figsize=(20, 15))
for index in range(num_trees_to_visualize):
    plt.subplot(1, 3, index + 1)
    plot_tree(model.estimators_[index], feature_names=X_train.columns,
filled=True, fontsize=10) # Memperbaiki sintaks plot_tree
    plt.title(f'Tree {index + 1}')
    plt.tight_layout()
plt.show()
```

```
C:\Users\kekop\AppData\Local\Temp\ipykernel_24320\1168726285.py:7:
UserWarning: The figure layout has changed to tight
  plt.tight_layout()
```

Tree 1



Tree 2



Tree 3