Perform the following operations using ‘python’ Language on the Heart Disease dataset.

1. Data cleaning
2. Data integration
3. Data transformation
4. Data model Building

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df = pd.read_csv(r"C:\Users\yasha\Desktop\Ashish\sem 6\DSBDA\DSBDA Lab Datase
df
```

Out[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targ
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	

1025 rows × 14 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1025 non-null   int64  
 1   sex       1025 non-null   int64  
 2   cp        1025 non-null   int64  
 3   trestbps  1025 non-null   int64  
 4   chol      1025 non-null   int64  
 5   fbs       1025 non-null   int64  
 6   restecg   1025 non-null   int64  
 7   thalach   1025 non-null   int64  
 8   exang     1025 non-null   int64  
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64  
 11  ca        1025 non-null   int64  
 12  thal      1025 non-null   int64  
 13  target    1025 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

In [ ]:
```
# data cleaning
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
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
```

In [7]:
```
df.duplicated()
```

Out[7]:
```
0       False
1       False
2       False
3       False
4       False
        ...  
1020     True
1021     True
1022     True
1023     True
1024     True
Length: 1025, dtype: bool
```

In [10]:
```
duplicate_rows = df[df.duplicated()]
print("number of duplicate rows are :", duplicate_rows.shape)
```

```
number of duplicate rows are : (723, 14)
```

In [15]:
```
df=df.drop_duplicates()
duplicate_rows = df[df.duplicated()]
print("number of duplicated rows are :",duplicate_rows.shape)
```

```
number of duplicated rows are : (0, 14)
```

In [ ]:
```
#data integration
```

In [17]:
```
df1=df[['age','sex','cp','ca']].loc[0:15]
df1
```

Out[17]:

	age	sex	cp	ca
0	52	1	0	2
1	53	1	0	0
2	70	1	0	0
3	61	1	0	1
4	62	0	0	3
5	58	0	0	0
6	58	1	0	3
7	55	1	0	1
8	46	1	0	0
9	54	1	0	2
10	71	0	0	0
11	43	0	0	0
12	34	0	1	0
13	51	1	0	3
14	52	1	0	0

In [18]:
```
df2=df[['age','sex','cp','ca']].loc[16:30]
df2
```

Out[18]:

	age	sex	cp	ca
16	51	0	2	1
17	54	1	0	1
18	50	0	1	0
19	58	1	2	0
20	60	1	2	0
21	67	0	0	2
22	45	1	0	0
23	63	0	2	0
24	42	0	2	0
25	61	0	0	0
26	44	1	2	0
27	58	0	1	2
28	56	1	2	1
29	55	0	0	0
30	44	1	0	0

In [19]:
```
merge = pd.merge(df1,df2,on="age",how="inner")
merge
```

Out[19]:

	age	sex_x	cp_x	ca_x	sex_y	cp_y	ca_y
0	61	1	0	1	0	0	0
1	58	0	0	0	1	2	0
2	58	0	0	0	0	1	2
3	58	1	0	3	1	2	0
4	58	1	0	3	0	1	2
5	55	1	0	1	0	0	0
6	54	1	0	2	1	0	1
7	51	1	0	3	0	2	1

In [ ]:
```
#data transformation
```

In [22]:
```
df['target']=df['target'].apply(lambda x:1 if x>0 else 0)
df
```

Out[22]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targ
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
723	68	0	2	120	211	0	0	115	0	1.5	1	0	2	
733	44	0	2	108	141	0	1	175	0	0.6	1	0	2	
739	52	1	0	128	255	0	1	161	1	0.0	2	1	3	
843	59	1	3	160	273	0	0	125	0	0.0	2	0	2	
878	54	1	0	120	188	0	1	113	0	1.4	1	1	3	

302 rows × 14 columns

In [ ]:
```
#error correction
```

In [24]:
```
df =df.applymap(lambda x: df.mean() if x< 0 else x)
df
# if there is negative value it will replace it with mean of the data frame
```

Out[24]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targ
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
723	68	0	2	120	211	0	0	115	0	1.5	1	0	2	
733	44	0	2	108	141	0	1	175	0	0.6	1	0	2	
739	52	1	0	128	255	0	1	161	1	0.0	2	1	3	
843	59	1	3	160	273	0	0	125	0	0.0	2	0	2	
878	54	1	0	120	188	0	1	113	0	1.4	1	1	3	

302 rows × 14 columns

In [ ]:
```
#model building
```

In [27]:
```
from sklearn.model_selection import train_test_split
x = merge.drop(['age'], axis=1)
y = merge['age']
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_sta
```

In [29]:
```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(x_train,y_train)
```

Out[29]:
```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [31]:
```
from sklearn.metrics import classification_report, confusion_matrix
y_pred = logreg.predict(x_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[0 0 0 0]
 [1 0 0 0]
 [0 0 1 0]
 [1 0 0 0]]
              precision    recall  f1-score   support

          54       0.00      0.00      0.00         0
          55       0.00      0.00      0.00         1
          58       1.00      1.00      1.00         1
          61       0.00      0.00      0.00         1

    accuracy                           0.33         3
   macro avg       0.25      0.25      0.25         3
weighted avg       0.33      0.33      0.33         3


C:\Users\yasha\anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\yasha\anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1344: UndefinedMetricWarning: Recall and F-score are ill-defined and being
set to 0.0 in labels with no true samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\yasha\anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\yasha\anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1344: UndefinedMetricWarning: Recall and F-score are ill-defined and being
set to 0.0 in labels with no true samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\yasha\anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\yasha\anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1344: UndefinedMetricWarning: Recall and F-score are ill-defined and being
set to 0.0 in labels with no true samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```