```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Downloading and Displaying the Dataset

```
df = pd.read_csv("/content/drive/MyDrive/DSBDA_dataset/heart_disease.csv")
```

```
df.head(20)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2
10	54	1	0	140	239	0	1	160	0	1.2	2	0	2
11	48	0	2	130	275	0	1	139	0	0.2	2	0	2
12	49	1	1	130	266	0	1	171	0	0.6	2	0	2
13	64	1	3	110	211	0	0	144	1	1.8	1	0	2
14	58	0	3	150	283	1	0	162	0	1.0	2	0	2
15	50	0	2	120	219	0	1	158	0	1.6	1	0	2
16	58	0	2	120	340	0	1	172	0	0.0	2	0	2
17	66	0	3	150	226	0	1	114	0	2.6	0	0	2
18	43	1	0	150	247	0	1	171	0	1.5	2	0	2
19	69	0	3	140	239	0	1	151	0	1.8	2	2	2

```
df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.

```
df.info()
```

```
<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
```

```
df.shape
```

```
(303, 14)
```

Data Cleaning

```
df.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
```

```
df.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

Change the datatype

```
df.dtypes
```

```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

```
df['age']= df['age'].astype('object')
df['sex']= df['sex'].astype('object')
df['cp']= df['cp'].astype('object')
df['trestbps']= df['trestbps'].astype('object')
df['chol']= df['chol'].astype('object')
df['fbs']= df['fbs'].astype('object')
```

```
df['restecg']= df['restecg'].astype('object')
df['thalach']= df['thalach'].astype('object')
df['exang']= df['exang'].astype('object')
df['oldpeak']= df['oldpeak'].astype('object')
df['slope']= df['slope'].astype('object')
df['ca']= df['ca'].astype('object')
df['thal']= df['thal'].astype('object')
df['target']= df['target'].astype('object')
```

```
df.dtypes
```

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

```
df['target'] = df.target.replace({1: "Disease", 0: "No_disease"})
df['sex'] = df.sex.replace({1: "Male", 0: "Female"})
df['cp'] = df.cp.replace({0: "typical_angina", 1: "atypical_angina", 2:"non-anginal pain", 3: "asymtomatic"})
df['exang'] = df.exang.replace({1: "Yes", 0: "No"})
df['fbs'] = df.fbs.replace({1: "True", 0: "False"})
df['slope'] = df.slope.replace({0: "upsloping", 1: "flat",2:"downsloping"})
df['thal'] = df.thal.replace({1: "fixed_defect", 2: "reversable_defect", 3:"normal"})
```

```
df.head(20)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
0	63	Male	asymtomatic	145	233	True	0	150	No	2.3	upsloping	0
1	37	Male	non-anginal pain	130	250	False	1	187	No	3.5	upsloping	0
2	41	Female	atypical_angina	130	204	False	0	172	No	1.4	downsloping	0
3	56	Male	atypical_angina	120	236	False	1	178	No	0.8	downsloping	0
4	57	Female	typical_angina	120	354	False	1	163	Yes	0.6	downsloping	0
5	57	Male	typical_angina	140	192	False	1	148	No	0.4	flat	0
6	56	Female	atypical_angina	140	294	False	0	153	No	1.3	flat	0
7	44	Male	atypical_angina	120	263	False	1	173	No	0.0	downsloping	0
8	52	Male	non-anginal pain	172	199	True	1	162	No	0.5	downsloping	0
9	57	Male	non-anginal pain	150	168	False	1	174	No	1.6	downsloping	0
10	54	Male	typical_angina	140	239	False	1	160	No	1.2	downsloping	0
11	48	Female	non-anginal pain	130	275	False	1	139	No	0.2	downsloping	0
12	49	Male	atypical_angina	130	266	False	1	171	No	0.6	downsloping	0
13	64	Male	asymtomatic	110	211	False	0	144	Yes	1.8	flat	0
14	58	Female	asymtomatic	150	283	True	0	162	No	1.0	downsloping	0
15	50	Female	non-anginal pain	120	219	False	1	158	No	1.6	flat	0
16	58	Female	non-anginal pain	120	340	False	1	172	No	0.0	downsloping	0

```
continous_features = ['age','trestbps','chol','thalach','oldpeak']
def outliers(df_out, drop = False):
  for each_feature in df_out.columns:
      feature_data = df_out[each_feature]
      Q1 = np.percentile(feature_data, 25.) # 25th percentile of the data of the given feature
      Q3 = np.percentile(feature_data, 75.) # 75th percentile of the data of the given feature
      IQR = Q3-Q1 #Interquartile Range
      outlier_step = IQR * 1.5 #That's we were talking about above
      outliers = feature_data[~((feature_data >= Q1 - outlier_step) &
      (feature_data <= Q3 + outlier_step))].index.tolist()
      if not drop:
        print('For the feature {}, No of Outliers is {}'.format(each_feature, len(outliers)))
      if drop:
        df.drop(outliers, inplace = True, errors = 'ignore')
        print('Outliers from {} feature removed'.format(each_feature))
```

```
outliers(df[continous_features])
```

```
For the feature age, No of Outliers is 0
For the feature trestbps, No of Outliers is 9
For the feature chol, No of Outliers is 5
For the feature thalach, No of Outliers is 1
For the feature oldpeak, No of Outliers is 5
```

```
outliers(df[continous_features], drop=True)
```

```
Outliers from age feature removed
Outliers from trestbps feature removed
Outliers from chol feature removed
Outliers from thalach feature removed
Outliers from oldpeak feature removed
```

```
df.shape
```

```
(284, 14)
```

```
duplicated=df.duplicated().sum()
if duplicated:
 print("Duplicated rows :{}".format(duplicated))
else:
 print("No duplicates")
duplicates=df[df.duplicated(keep=False)]
duplicates.head()
```

```
Duplicated rows :1
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
163	38	Male	non-anginal pain	138	175	False	1	173	No	0.0	downsloping	4	reversabl

```
df1 = df.drop_duplicates()
df1.shape
```

```
(283, 14)
```

```
df1.head(20)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
0	63	Male	asymtomatic	145	233	True	0	150	No	2.3	upsloping	0
1	37	Male	non-anginal pain	130	250	False	1	187	No	3.5	upsloping	0
2	41	Female	atypical_angina	130	204	False	0	172	No	1.4	downsloping	0
3	56	Male	atypical_angina	120	236	False	1	178	No	0.8	downsloping	0
4	57	Female	typical_angina	120	354	False	1	163	Yes	0.6	downsloping	0
5	57	Male	typical_angina	140	192	False	1	148	No	0.4	flat	0
6	56	Female	atypical_angina	140	294	False	0	153	No	1.3	flat	0
7	44	Male	atypical_angina	120	263	False	1	173	No	0.0	downsloping	0
9	57	Male	non-anginal pain	150	168	False	1	174	No	1.6	downsloping	0
10	54	Male	typical_angina	140	239	False	1	160	No	1.2	downsloping	0
11	48	Female	non-anginal pain	130	275	False	1	139	No	0.2	downsloping	0
12	49	Male	atypical_angina	130	266	False	1	171	No	0.6	downsloping	0
13	64	Male	asymtomatic	110	211	False	0	144	Yes	1.8	flat	0
14	58	Female	asymtomatic	150	283	True	0	162	No	1.0	downsloping	0
15	50	Female	non-anginal pain	120	219	False	1	158	No	1.6	flat	0
16	58	Female	non-anginal pain	120	340	False	1	172	No	0.0	downsloping	0
17	66	Female	asymtomatic	150	226	False	1	114	No	2.6	upsloping	0
18	43	Male	typical_angina	150	247	False	1	171	No	1.5	downsloping	0

DATA TRANSFORMATION

```
from sklearn.preprocessing import LabelEncoder
labelencoder=LabelEncoder()
df1["sex"]=labelencoder.fit_transform(df1["sex"])
```

```
<ipython-input-21-413a3bd97e78>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  df1["sex"]=labelencoder.fit_transform(df1["sex"])
```

```
df1.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
0	63	1	asymtomatic	145	233	True	0	150	No	2.3	upsloping	0	
1	37	1	non-anginal pain	130	250	False	1	187	No	3.5	upsloping	0	reve
2	41	0	atypical_angina	130	204	False	0	172	No	1.4	downsloping	0	reve
3	56	1	atypical_angina	120	236	False	1	178	No	0.8	downsloping	0	reve
4	57	0	typical_angina	120	354	False	1	163	Yes	0.6	downsloping	0	reve

```
df1["sex"]=labelencoder.fit_transform(df1["sex"])
df1["cp"]=labelencoder.fit_transform(df1["cp"])
df1["fbs"]=labelencoder.fit_transform(df1["fbs"])
df1["exang"]=labelencoder.fit_transform(df1["exang"])
df1["slope"]=labelencoder.fit_transform(df1["slope"])
# df1["thal"]=labelencoder.fit_transform(df1["thal"])
df1["target"]=labelencoder.fit_transform(df1["target"])
```

```
<ipython-input-23-711d2950fcc3>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  df1["sex"]=labelencoder.fit_transform(df1["sex"])
<ipython-input-23-711d2950fcc3>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  df1["cp"]=labelencoder.fit_transform(df1["cp"])
<ipython-input-23-711d2950fcc3>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  df1["fbs"]=labelencoder.fit_transform(df1["fbs"])
<ipython-input-23-711d2950fcc3>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  df1["exang"]=labelencoder.fit_transform(df1["exang"])
<ipython-input-23-711d2950fcc3>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  df1["slope"]=labelencoder.fit_transform(df1["slope"])
<ipython-input-23-711d2950fcc3>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  df1["target"]=labelencoder.fit_transform(df1["target"])
```

```
df1.head()
```

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

```
df1['thal'].value_counts()
```

```
reversable_defect    159
normal               105
fixed_defect          17
0                      2
Name: thal, dtype: int64
```

```
df1.loc[df1['thal']==0,'thal']=np.NaN
```

Error Handling

```
df1['thal'].value_counts()
```

```
reversable_defect    159
normal               105
fixed_defect          17
Name: thal, dtype: int64
```

```
df1["thal"]=labelencoder.fit_transform(df1["thal"])
```

```
<ipython-input-27-8d6102278ce5>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  df1["thal"]=labelencoder.fit_transform(df1["thal"])
```

```
df1['ca'].value_counts()
```

```
0    165
1     63
2     35
3     16
4      4
Name: ca, dtype: int64
```

```
df1.loc[df1['ca']==4,'ca']=np.NaN
```

```
df1['thal'].value_counts()
```

```
2    159
1    105
0     17
3      2
Name: thal, dtype: int64
```

```
df1.loc[df1['thal']==3,'thal']=np.NaN
```

```
df1.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          4
thal        2
target      0
dtype: int64
```

```
median_ca = df1['ca'].median()
# median
df1['ca'] = df1['ca'].fillna(median_ca)
```

```
<ipython-input-33-fd025d1935c9>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  df1['ca'] = df1['ca'].fillna(median_ca)
```

```
median_thal = df1['thal'].median()
# median
df1['thal'] = df1['thal'].fillna(median_thal)
```

```
<ipython-input-34-abf395e8b7c4>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  df1['thal'] = df1['thal'].fillna(median_thal)
```

```
df1.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
```

DATA MODEL BUILDING

```
# Create X (feature matrix)
X = df1.drop("target", axis=1)

#Create Y (labels)
y = df1["target"]
```

```
np.random.seed(42)

from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier()

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

clf.fit(X_train, y_train);
```

```
clf.score(X_test, y_test)
```

```
0.8070175438596491
```

```
#make prediciton
y_preds = clf.predict(X_test)
y_preds
```

```
array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0])
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
def evaluate_preds(y_true, y_preds):
    """Perform evaluation comparison on y_true labels vs y_pred labels"""
    accuracy = accuracy_score(y_true, y_preds)
    precision = precision_score(y_true, y_preds)
    recall = recall_score(y_true, y_preds)
    f1 = f1_score(y_true, y_preds)
    metric_dict = {"accuracy": round(accuracy, 2),
                   "precision": round(precision, 2),
                   "recall": round(recall, 2),
                   "f1": round(f1, 2)}
    print(f"Acc: {accuracy * 100:.2f}%")
    print(f"Precision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1 score: {f1:.2f}")

    return metric_dict
```

```
evaluate_preds(y_test, y_preds)
```

```
Acc: 80.70%
Precision: 0.83
Recall: 0.74
```

```
F1 score: 0.78
{'accuracy': 0.81, 'precision': 0.83, 'recall': 0.74, 'f1': 0.78}
```

Data ingestion refers to the process of importing, collecting, and processing data from various sources into a data storage system or data lake for further analysis. The data sources could be varied, such as databases, files, sensors, IoT devices, social media, or streaming services.

There are several methods of data ingestion, including:

Batch Ingestion: In this method, data is collected in large batches, typically during off-peak hours, and transferred to the target system for processing. This method is ideal for handling large datasets that do not require immediate analysis.

Real-time Ingestion: This method involves the continuous streaming of data in real-time from the source system to the target system. It is ideal for processing data that requires immediate action, such as monitoring sensor data, social media feeds, or stock market feeds.

Change Data Capture (CDC): CDC captures the changes made to the source data since the last ingestion. It helps in reducing the data processing time by only processing the updated data. This method is ideal for processing data that is frequently updated.

Event-based Ingestion: In this method, data is ingested based on specific events or triggers, such as a new file being uploaded to a directory, a sensor reading a certain value, or a social media post with a specific keyword.

Log-based Ingestion: This method involves collecting and processing logs generated by different systems and applications. It helps in identifying and troubleshooting errors and issues in the system.

Apache NiFi: A web-based platform that enables users to design and automate the flow of data between systems. NiFi supports **real-time and batch data ingestion** and has built-in support for various data sources.

Apache Kafka: A distributed streaming platform that enables users to publish and subscribe to streams of records in **real-time**. Kafka is highly scalable and can handle large volumes of data.

Apache Flume: A distributed, reliable, and available system for efficiently collecting, aggregating, and moving large amounts of **log data** from different sources to a centralized data store.

Google Cloud Dataflow: A fully managed service for executing **batch and streaming data** processing pipelines. Dataflow supports multiple programming languages and integrates with other Google Cloud services.

AWS Glue: A fully managed extract, transform, and load (ETL) service that makes it easy to move data between data stores. Glue supports **real-time and batch data ingestion** and integrates with other AWS services.

Selenium: A popular web browser automation tool that can be used for web scraping and data extraction from websites.

Beautiful Soup: A Python library used for web scraping HTML and XML documents.