```
#Importing required libraries
import numpy as np
import pandas as pd
```

```
#Generate a dummy dataset.
#X = np.random.randint(10,50,100).reshape(20,5)
df=pd.read_csv('ctg_data.csv')
df
```

| | b | e | AC | FM | UC | DL | DS | DP | DR | LB | ... | C | D | E | AD | DE | LD | FS | SUSP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 240 | 357 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 120 | ... | -1 | -1 | -1 | -1 | -1 | -1 | 1 | -1 |
| **1** | 5 | 632 | 4 | 0 | 4 | 2 | 0 | 0 | 0 | 132 | ... | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 |
| **2** | 177 | 779 | 2 | 0 | 5 | 2 | 0 | 0 | 0 | 133 | ... | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 |
| **3** | 411 | 1192 | 2 | 0 | 6 | 2 | 0 | 0 | 0 | 134 | ... | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 |
| **4** | 533 | 1147 | 4 | 0 | 5 | 0 | 0 | 0 | 0 | 132 | ... | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **2121** | 2059 | 2867 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 140 | ... | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 |
| **2122** | 1576 | 2867 | 1 | 0 | 9 | 0 | 0 | 0 | 0 | 140 | ... | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 |
| **2123** | 1576 | 2596 | 1 | 0 | 7 | 0 | 0 | 0 | 0 | 140 | ... | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 |
| **2124** | 1576 | 3049 | 1 | 0 | 9 | 0 | 0 | 0 | 0 | 140 | ... | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 |
| **2125** | 2796 | 3415 | 1 | 1 | 5 | 0 | 0 | 0 | 0 | 142 | ... | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

2126 rows × 42 columns

```
df.dtypes
```

```
b             int64
e             int64
AC            int64
FM            int64
UC            int64
DL            int64
DS            int64
DP            int64
DR            int64
LB            int64
AC.1        float64
FM.1        float64
UC.1        float64
DL.1        float64
DS.1        float64
DP.1        float64
```

```
ASTV          int64
MSTV        float64
ALTV          int64
MLTV        float64
Width         int64
Min           int64
Max           int64
Nmax          int64
Nzeros        int64
Mode          int64
Mean          int64
Median        int64
Variance      int64
Tendency      int64
A             int64
B             int64
C             int64
D             int64
E             int64
AD            int64
DE            int64
LD            int64
FS            int64
SUSP          int64
CLASS         int64
NSP           int64
dtype: object
```

```
df.isna().sum()
```

```
b           0
e           0
AC          0
FM          0
UC          0
DL          0
DS          0
DP          0
DR          0
LB          0
AC.1        0
FM.1        0
UC.1        0
DL.1        0
DS.1        0
DP.1        0
ASTV        0
MSTV        0
ALTV        0
MLTV        0
Width       0
Min         0
Max         0
Nmax        0
Nzeros      0
```

```
Mode        0
Mean        0
Median      0
Variance    0
Tendency    0
A           0
B           0
C           0
D           0
E           0
AD          0
DE          0
LD          0
FS          0
SUSP        0
CLASS       0
NSP         0
dtype: int64
```

```
df.dropna()
```

| | b | e | AC | FM | UC | DL | DS | DP | DR | LB | ... | C | D |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 240 | 357 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 120 | ... | -1 | -1 |
| **1** | 5 | 632 | 4 | 0 | 4 | 2 | 0 | 0 | 0 | 132 | ... | -1 | -1 |
| **2** | 177 | 779 | 2 | 0 | 5 | 2 | 0 | 0 | 0 | 133 | ... | -1 | -1 |
| **3** | 411 | 1192 | 2 | 0 | 6 | 2 | 0 | 0 | 0 | 134 | ... | -1 | -1 |
| **4** | 533 | 1147 | 4 | 0 | 5 | 0 | 0 | 0 | 0 | 132 | ... | -1 | -1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **2121** | 2059 | 2867 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 140 | ... | -1 | -1 |
| **2122** | 1576 | 2867 | 1 | 0 | 9 | 0 | 0 | 0 | 0 | 140 | ... | -1 | -1 |
| **2123** | 1576 | 2596 | 1 | 0 | 7 | 0 | 0 | 0 | 0 | 140 | ... | -1 | -1 |

```
df.isna().sum()
```

```
b           0
e           0
AC          0
FM          0
UC          0
DL          0
DS          0
```

```
DP          0
DR          0
LB          0
AC.1        0
FM.1        0
UC.1        0
DL.1        0
DS.1        0
DP.1        0
ASTV        0
MSTV        0
ALTV        0
MLTV        0
Width       0
Min         0
Max         0
Nmax        0
Nzeros      0
Mode        0
Mean        0
Median      0
Variance    0
Tendency    0
A           0
B           0
C           0
D           0
E           0
AD          0
DE          0
LD          0
FS          0
SUSP        0
CLASS       0
NSP         0
dtype: int64
```

```
Features=df.drop('NSP', axis=1)
Label=df['NSP']
```

## Split Data into Training/Testing

```
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics
```

```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(Features, Label, test_size=0.3, random_st
```

# Decision Tree

```
# Import Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier

# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred_train = clf.predict(X_train)
print("Decision Tree Model Accuracy with training data (in %):",metrics.accuracy_score(y_trai
```

```
Decision Tree Model Accuracy with training data (in %): 99.93279569892472
```

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=8)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

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

print("Decision Tree model accuracy(in %):",metrics.accuracy_score(y_test, y_pred)*100)
```

```
Decision Tree model accuracy(in %): 98.58934169278997
```

# Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)

y_pred_train = gnb.predict(X_train)

print('Training-set accuracy(in %):', metrics.accuracy_score(y_train, y_pred_train)*100)
```

```
Training-set accuracy(in %): 98.65591397849462
```

```
# making predictions on the testing set
y_pred = gnb.predict(X_test)

# comparing actual response values (y_test) with predicted response values
(y_pred)
```

```
from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test, y_pred)*10
```

```
Gaussian Naive Bayes model accuracy(in %): 98.11912225705329
```

## Random Forest

```
# importing random forest classifier from assemble module
from sklearn.ensemble import RandomForestClassifier
```

```
# creating a RF classifier
rfclf = RandomForestClassifier(n_estimators = 100)

# Training the model on the training dataset
# fit function is used to train the model using the training sets as parameters
rfclf.fit(X_train, y_train)

y_pred_train = rfclf.predict(X_train)

print('Training-set accuracy(in %):', metrics.accuracy_score(y_train, y_pred_train)*100)
```

```
Training-set accuracy(in %): 99.93279569892472
```

```
# performing predictions on the test dataset
y_pred = rfclf.predict(X_test)

# using metrics module for accuracy calculation
print("Random Forest model accuracy(in %): ", metrics.accuracy_score(y_test, y_pred)*100)
```

```
Random Forest model accuracy(in %):  99.21630094043887
```

## SVM

```
#Import svm model
from sklearn import svm

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

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

y_pred_train = svmclf.predict(X_train)
```

```
print('Training-set accuracy(in %):', metrics.accuracy_score(y_train, y_pred_train)*100)
```

```
Training-set accuracy(in %): 99.1263440860215
```

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

# using metrics module for accuracy calculation
print("SVM model accuracy(in %): ", metrics.accuracy_score(y_test, y_pred)*100)
```

```
SVM model accuracy(in %):  98.58934169278997
```