```
import pandas as pd


data=pd.read_csv('/content/breast_cancer_survival.csv')
print(data)
```

```
          Age  Gender  Protein1  Protein2  Protein3  Protein4 Tumour_Stage  \
0          42  FEMALE  0.952560   2.15000  0.007972 -0.048340           II
1          54  FEMALE  0.000000   1.38020 -0.498030 -0.507320           II
2          63  FEMALE -0.523030   1.76400 -0.370190  0.010815           II
3          78  FEMALE -0.876180   0.12943 -0.370380  0.132190            I
4          42  FEMALE  0.226110   1.74910 -0.543970 -0.390210           II
..        ...     ...       ...       ...       ...       ...          ...
329        59  FEMALE  0.024598   1.40050  0.024751  0.280320           II
330        41  FEMALE  0.100120  -0.46547  0.472370 -0.523870            I
331        54  FEMALE  0.753820   1.64250 -0.332850  0.857860           II
332        74  FEMALE  0.972510   1.42680 -0.366570 -0.107820           II
333        66  FEMALE  0.286380   1.39980  0.318830  0.836050           II

                           Histology ER status PR status HER2 status  \
0     Infiltrating Ductal Carcinoma  Positive  Positive    Negative
1     Infiltrating Ductal Carcinoma  Positive  Positive    Negative
2     Infiltrating Ductal Carcinoma  Positive  Positive    Negative
3     Infiltrating Ductal Carcinoma  Positive  Positive    Negative
4     Infiltrating Ductal Carcinoma  Positive  Positive    Positive
..                              ...       ...       ...         ...
329   Infiltrating Ductal Carcinoma  Positive  Positive    Positive
330   Infiltrating Ductal Carcinoma  Positive  Positive    Positive
331   Infiltrating Ductal Carcinoma  Positive  Positive    Negative
332  Infiltrating Lobular Carcinoma  Positive  Positive    Negative
333   Infiltrating Ductal Carcinoma  Positive  Positive    Negative

                     Surgery_type Date_of_Surgery Date_of_Last_Visit  \
0                           Other       20-May-18          26-Aug-18
1                           Other       26-Apr-18          25-Jan-19
2                      Lumpectomy       24-Aug-18          08-Apr-20
3                           Other       16-Nov-18          28-Jul-20
4                      Lumpectomy       12-Dec-18          05-Jan-19
..                            ...             ...                ...
329                    Lumpectomy       15-Jan-19          27-Mar-20
330  Modified Radical Mastectomy       25-Jul-18          23-Apr-19
331            Simple Mastectomy       26-Mar-19          11-Oct-19
332                    Lumpectomy       26-Nov-18          05-Dec-18
333  Modified Radical Mastectomy       04-Feb-19          10-Aug-19

    Patient_Status
0            Alive
1             Dead
2            Alive
3            Alive
4            Alive
..             ...
329          Alive
330          Alive
331           Dead
332          Alive
333           Dead

[334 rows x 15 columns]
```

```
print(data.dtypes)
print("shape",data.shape)
```

```
Age                    int64
Gender                object
Protein1             float64
Protein2             float64
Protein3             float64
Protein4             float64
Tumour_Stage          object
Histology             object
ER status             object
PR status             object
HER2 status           object
Surgery_type          object
Date_of_Surgery       object
Date_of_Last_Visit    object
Patient_Status        object
dtype: object
shape (334, 15)
```

```
h=data.drop({'Histology','ER status','PR status','Date_of_Surgery','Date_of_Last_Visit','Surgery_type','HER2 status'},axis=1)


print(h.isnull().sum())
```

```
Age               0
Gender            0
Protein1          0
Protein2          0
Protein3          0
Protein4          0
Tumour_Stage      0
Patient_Status    13
dtype: int64
```

```
data['Patient_Status'].isnull().sum()/h.shape[0]
```

```
0.038922155688622756
```

```
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
```

```
encoder = LabelEncoder()
h['Gender'] = encoder.fit_transform(h['Gender'])
```

```
# Define mapping for 'Gender' column
gender_mapping = {'Female': 1, 'Male': 0}

# Map 'Gender' column using the defined mapping, and fill any NaN values with 0
h['Gender'] = h['Gender'].map(gender_mapping).fillna(0).astype(int)

# Define mapping for 'Tumour_Stage' column
tumour_stage_mapping = {'Stage I': 1, 'Stage II': 2, 'Stage III': 3}

# Map 'Tumour_Stage' column using the defined mapping, and fill any NaN values with 0
h['Tumour_Stage'] = h['Tumour_Stage'].map(tumour_stage_mapping).fillna(0).astype(int)
Patient_Status_mapping = {'Alive': 1, 'Dead': 0}
```

```
h['Patient_Status'] = h['Patient_Status'].map(Patient_Status_mapping).fillna(0).astype(int)

# Now, 'Gender' column will have 1 for 'Female' and 0 for 'Male', and 'Tumour_Stage' column will have corresponding integers for each s
```

```
h.describe()
```

	Age	Gender	Protein1	Protein2	Protein3	Protein4	Tumour_Stage
count	334.000000	334.0	334.000000	334.000000	334.000000	334.000000	334.0
mean	58.886228	0.0	-0.029991	0.946896	-0.090204	0.009819	0.0
std	12.961212	0.0	0.563588	0.911637	0.585175	0.629055	0.0
min	29.000000	0.0	-2.340900	-0.978730	-1.627400	-2.025500	0.0
25%	49.000000	0.0	-0.358888	0.362173	-0.513748	-0.377090	0.0
50%	58.000000	0.0	0.006129	0.992805	-0.173180	0.041768	0.0
75%	68.000000	0.0	0.343598	1.627900	0.278353	0.425630	0.0
max	90.000000	0.0	1.593600	3.402200	2.193400	1.629900	0.0

```
y=h['Patient_Status']
x=h.drop('Patient_Status',axis=1)
```

```
x=x.fillna(0)
```

```
print(x)
print(y)
```

```
     Age  Gender  Protein1  Protein2  Protein3  Protein4  Tumour_Stage
0     42       0  0.952560   2.15000  0.007972 -0.048340             0
1     54       0  0.000000   1.38020 -0.498030 -0.507320             0
2     63       0 -0.523030   1.76400 -0.370190  0.010815             0
3     78       0 -0.876180   0.12943 -0.370380  0.132190             0
4     42       0  0.226110   1.74910 -0.543970 -0.390210             0
..   ...     ...       ...       ...       ...       ...           ...
329   59       0  0.024598   1.40050  0.024751  0.280320             0
330   41       0  0.100120  -0.46547  0.472370 -0.523870             0
331   54       0  0.753820   1.64250 -0.332850  0.857860             0
332   74       0  0.972510   1.42680 -0.366570 -0.107820             0
333   66       0  0.286380   1.39980  0.318830  0.836050             0

[334 rows x 7 columns]
```

```
0      1
1      0
2      1
3      1
4      1
      ..
329    1
330    1
331    0
332    1
333    0
Name: Patient_Status, Length: 334, dtype: int64
```

```
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, random_state=42)
```

```
print(x_train.isnull())
```

```
print(x_train)
```

```
       Age  Gender  Protein1  Protein2  Protein3  Protein4  Tumour_Stage
224  False   False     False     False     False     False         False
78   False   False     False     False     False     False         False
295  False   False     False     False     False     False         False
17   False   False     False     False     False     False         False
24   False   False     False     False     False     False         False
..     ...     ...       ...       ...       ...       ...           ...
188  False   False     False     False     False     False         False
71   False   False     False     False     False     False         False
106  False   False     False     False     False     False         False
270  False   False     False     False     False     False         False
102  False   False     False     False     False     False         False

[267 rows x 7 columns]
     Age  Gender  Protein1  Protein2  Protein3  Protein4  Tumour_Stage
224   38       0 -0.268450   0.19515 -1.024700  0.101720             0
78    77       0 -0.298700  -0.16129  0.460720 -0.396660             0
295   47       0 -0.190060   1.97790 -0.007615  0.035325             0
17    63       0  0.052728   0.72210 -0.308650 -0.531290             0
24    70       0  0.700290   0.97347 -0.296450  0.105510             0
..   ...     ...       ...       ...       ...       ...           ...
188   44       0 -0.278840   2.16880 -0.462330  0.272200             0
71    45       0 -0.711630   1.69240 -0.800760  0.794400             0
106   49       0  0.061643   1.31490 -0.099357  0.754410             0
270   79       0 -0.482690  -0.31677  0.471580  0.347440             0
102   59       0  0.359140   1.18520 -0.423490  0.415940             0

[267 rows x 7 columns]
```

```
print(x_test)
```

```
     Age  Gender  Protein1  Protein2  Protein3  Protein4  Tumour_Stage
25    76       0  0.043546  -0.40171  0.466850   1.04780             0
309   52       0 -0.179320   1.52870 -0.163130   0.83222             0
73    66       0 -1.344100   1.12800 -0.229350  -0.22993             0
195   76       0 -0.245140   0.61407  0.126510   0.39114             0
57    55       0 -0.663420   1.93820 -0.775370  -0.26366             0
..   ...     ...       ...       ...       ...       ...           ...
280   57       0 -0.213780  -0.62814  1.022600   0.20244             0
3     78       0 -0.876180   0.12943 -0.370380   0.13219             0
77    59       0  0.276410  -0.87612  1.298000  -0.68664             0
311   42       0  0.172950   1.29660 -0.856770  -0.23695             0
60    56       0 -0.675420   0.26937 -0.086603   1.07140             0

[67 rows x 7 columns]
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(x_train)
X_test_scaled = scaler.transform(x_test)
```

```
# b) Train Random Forest and AdaBoost
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier

# Train Random Forest
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train_scaled, y_train)

# Train AdaBoost
adaboost_classifier = AdaBoostClassifier(random_state=42)
adaboost_classifier.fit(X_train_scaled, y_train)
```

```
# c) Evaluate models
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Predictions
rf_pred = rf_classifier.predict(X_test_scaled)
adaboost_pred = adaboost_classifier.predict(X_test_scaled)

# Evaluation metrics
rf_accuracy = accuracy_score(y_test, rf_pred)
rf_precision = precision_score(y_test, rf_pred)
rf_recall = recall_score(y_test, rf_pred)
rf_f1 = f1_score(y_test, rf_pred)

print("Random Forest Metrics:")
print("Accuracy:", rf_accuracy)
print("Precision:", rf_precision)
print("Recall:", rf_recall)
print("F1-score:", rf_f1)

adaboost_accuracy = accuracy_score(y_test, adaboost_pred)
adaboost_precision = precision_score(y_test, adaboost_pred)
adaboost_recall = recall_score(y_test, adaboost_pred)
adaboost_f1 = f1_score(y_test, adaboost_pred)

print("AdaBoost Metrics:")
print("Accuracy:", adaboost_accuracy)
print("Precision:", adaboost_precision)
print("Recall:", adaboost_recall)
print("F1-score:", adaboost_f1)
```

```
Random Forest Metrics:
Accuracy: 0.7611940298507462
Precision: 0.7727272727272727
Recall: 0.9807692307692307
F1-score: 0.864406779661017
AdaBoost Metrics:
Accuracy: 0.746268656716418
Precision: 0.7777777777777778
Recall: 0.9423076923076923
F1-score: 0.8521739130434781
```