```
import pandas as pd

df = pd.read_csv('/content/breast_cancer_survival.csv')

print(df.head())

features = df.drop('Patient_Status', axis=1)
target = df['Patient_Status']

print("Features:")
print(features.head())

print("Target:")
print(target.head())
```

```
   Age  Gender  Protein1  Protein2  Protein3  Protein4 Tumour_Stage  \
0   42  FEMALE   0.95256   2.15000  0.007972 -0.048340           II
1   54  FEMALE   0.00000   1.38020 -0.498030 -0.507320           II
2   63  FEMALE  -0.52303   1.76400 -0.370190  0.010815           II
3   78  FEMALE  -0.87618   0.12943 -0.370380  0.132190            I
4   42  FEMALE   0.22611   1.74910 -0.543970 -0.390210           II

                      Histology ER status PR status HER2 status Surgery_type  \
0  Infiltrating Ductal Carcinoma  Positive  Positive    Negative        Other
1  Infiltrating Ductal Carcinoma  Positive  Positive    Negative        Other
2  Infiltrating Ductal Carcinoma  Positive  Positive    Negative   Lumpectomy
3  Infiltrating Ductal Carcinoma  Positive  Positive    Negative        Other
4  Infiltrating Ductal Carcinoma  Positive  Positive    Positive   Lumpectomy

  Date_of_Surgery Date_of_Last_Visit Patient_Status
0       20-May-18          26-Aug-18          Alive
1       26-Apr-18          25-Jan-19           Dead
2       24-Aug-18          08-Apr-20          Alive
3       16-Nov-18          28-Jul-20          Alive
4       12-Dec-18          05-Jan-19          Alive
Features:
   Age  Gender  Protein1  Protein2  Protein3  Protein4 Tumour_Stage  \
0   42  FEMALE   0.95256   2.15000  0.007972 -0.048340           II
1   54  FEMALE   0.00000   1.38020 -0.498030 -0.507320           II
2   63  FEMALE  -0.52303   1.76400 -0.370190  0.010815           II
3   78  FEMALE  -0.87618   0.12943 -0.370380  0.132190            I
4   42  FEMALE   0.22611   1.74910 -0.543970 -0.390210           II

                      Histology ER status PR status HER2 status Surgery_type  \
0  Infiltrating Ductal Carcinoma  Positive  Positive    Negative        Other
1  Infiltrating Ductal Carcinoma  Positive  Positive    Negative        Other
2  Infiltrating Ductal Carcinoma  Positive  Positive    Negative   Lumpectomy
3  Infiltrating Ductal Carcinoma  Positive  Positive    Negative        Other
4  Infiltrating Ductal Carcinoma  Positive  Positive    Positive   Lumpectomy

  Date_of_Surgery Date_of_Last_Visit
0       20-May-18          26-Aug-18
1       26-Apr-18          25-Jan-19
2       24-Aug-18          08-Apr-20
3       16-Nov-18          28-Jul-20
4       12-Dec-18          05-Jan-19
Target:
0    Alive
1     Dead
2    Alive
3    Alive
4    Alive
Name: Patient_Status, dtype: object
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score



label_encoders = {}
for column in df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le


X = df.drop('Patient_Status', axis=1)
y = df['Patient_Status']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


svm_model = SVC(kernel='linear')
svm_model.fit(X_train_scaled, y_train)
svm_predictions = svm_model.predict(X_test_scaled)
svm_accuracy = accuracy_score(y_test, svm_predictions)
print(f'SVM Accuracy: {svm_accuracy}')


rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)  # No need to scale data for Random Forest
rf_predictions = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)
print(f'Random Forest Accuracy: {rf_accuracy}')
```

```
SVM Accuracy: 0.7910447761194029
Random Forest Accuracy: 0.8059701492537313
```

```
label_encoders = {}
for column in df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le

X = df.drop('Patient_Status', axis=1)
y = df['Patient_Status']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
svm_model = SVC(kernel='linear')
svm_model.fit(X_train_scaled, y_train)
svm_predictions = svm_model.predict(X_test_scaled)
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)  # No need to scale data for Random Forest
rf_predictions = rf_model.predict(X_test)
def calculate_metrics(y_true, y_pred, model_name):
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='weighted')
    recall = recall_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')
    print(f"{model_name} Metrics:")
    print(f"Accuracy: {accuracy}")
    print(f"Precision: {precision}")
    print(f"Recall: {recall}")
    print(f"F1-Score: {f1}\n")
calculate_metrics(y_test, svm_predictions, "SVM")
calculate_metrics(y_test, rf_predictions, "Random Forest")
```

```
SVM Metrics:
Accuracy: 0.7910447761194029
Precision: 0.6432835820895523
Recall: 0.7910447761194029
F1-Score: 0.707794361525705

Random Forest Metrics:
Accuracy: 0.8059701492537313
Precision: 0.672002888781897
Recall: 0.8059701492537313
F1-Score: 0.7328021546403322

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined and be
  _warn_prf(average, modifier, msg_start, len(result))
```