```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
# Import preprocessing for OneHotEncoder
from sklearn.preprocessing import OneHotEncoder

df=pd.read_csv('/content/dataset.csv thyr.csv')

selected_features=['Age','Gender','Smoking','Pathology','Stage']
X=df[selected_features]
y=df['Recurred']

# Create a OneHotEncoder object
encoder = OneHotEncoder(sparse_output=False, handle_unknown='ignore') # sparse=False for array output

# Fit the encoder on the categorical features and transform them
encoded_features = encoder.fit_transform(X[['Gender', 'Smoking', 'Pathology', 'Stage']]) # Assuming these are your categorical features

# Create a DataFrame from the encoded features
encoded_df = pd.DataFrame(encoded_features, columns=encoder.get_feature_names_out(['Gender', 'Smoking', 'Pathology', 'Stage']))

# Concatenate the encoded features with the numerical features
X = pd.concat([X[['Age']], encoded_df], axis=1)

# Now, proceed with the model training
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1)

model=RandomForestClassifier()
model.fit(X_train,y_train)

y_pred=model.predict(X_test) # Corrected to X_test
accuracy=accuracy_score(y_test,y_pred)
print("Accuracy: ",accuracy)
```

```
Accuracy:  0.7402597402597403
```

```
print(X.head())
X.info()
```

```
   Age Gender Smoking       Pathology Stage
0   27      F      No  Micropapillary     I
1   34      F      No  Micropapillary     I
2   30      F      No  Micropapillary     I
3   62      F      No  Micropapillary     I
4   62      F      No  Micropapillary     I
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 383 entries, 0 to 382
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Age        383 non-null    int64
 1   Gender     383 non-null    object
 2   Smoking    383 non-null    object
 3   Pathology  383 non-null    object
 4   Stage      383 non-null    object
dtypes: int64(1), object(4)
memory usage: 15.1+ KB
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
# Import preprocessing for OneHotEncoder
from sklearn.preprocessing import OneHotEncoder

df = pd.read_csv('/content/dataset.csv thyr.csv')

selected_features = ['Age', 'Gender', 'Smoking', 'Pathology', 'Stage']
X = df[selected_features]
y = df['Recurred']

# Create a OneHotEncoder object to handle categorical features
encoder = OneHotEncoder(sparse_output=False, handle_unknown='ignore')  # sparse=False for array output

# Fit the encoder on the categorical features and transform them
```

```
categorical_features = ['Gender', 'Smoking', 'Pathology', 'Stage']
encoded_features = encoder.fit_transform(X[categorical_features])

# Create a DataFrame from the encoded features with proper column names
encoded_df = pd.DataFrame(encoded_features, columns=encoder.get_feature_names_out(categorical_features))

# Concatenate the encoded features with the numerical features ('Age')
X = pd.concat([X[['Age']], encoded_df], axis=1)

# Now, proceed with the model training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

model = RandomForestClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)
```

```
Accuracy:  0.7402597402597403
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('/content/dataset.csv thyr.csv')
print(df.info())
# Encode categorical variables
label_encoder = LabelEncoder()
df['Gender'] = label_encoder.fit_transform(df['Gender'])
print(df['Gender'])
df['Smoking'] = label_encoder.fit_transform(df['Smoking'])
print(df['Smoking'])
df['Pathology'] = label_encoder.fit_transform(df['Pathology'])
print(df['Pathology'])
df['Stage'] = label_encoder.fit_transform(df['Stage'])
print(df['Stage'])

selected_features = ['Age', 'Gender', 'Smoking', 'Pathology', 'Stage']
X = df[selected_features]
y = df['Recurred']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

model = RandomForestClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)
```

```
 11  T                    383 non-null    object
 12  N                    383 non-null    object
 13  M                    383 non-null    object
 14  Stage                383 non-null    object
 15  Response             383 non-null    object
 16  Recurred             383 non-null    object
dtypes: int64(1), object(16)
memory usage: 51.0+ KB
None
0      0
```

```
          ..
      378    1
      379    1
      380    1
      381    1
      382    1
      Name: Smoking, Length: 383, dtype: int64
      0      2
      1      2
      2      2
      3      2
      4      2
            ..
      378    3
      379    3
      380    3
      381    1
      382    3
      Name: Pathology, Length: 383, dtype: int64
      0      0
      1      0
      2      0
      3      0
      4      0
            ..
      378    4
      379    4
      380    4
      381    3
      382    3
      Name: Stage, Length: 383, dtype: int64
      Accuracy:  0.7532467532467533
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('/content/dataset.csv thyr.csv')
print(df.info())
# Encode categorical variables
label_encoder = LabelEncoder()
df['Gender'] = label_encoder.fit_transform(df['Gender'])
print(df['Gender'])
df['Smoking'] = label_encoder.fit_transform(df['Smoking'])
print(df['Smoking'])
df['Pathology'] = label_encoder.fit_transform(df['Pathology'])
print(df['Pathology'])
df['Stage'] = label_encoder.fit_transform(df['Stage'])
print(df['Stage'])

selected_features = ['Age', 'Gender', 'Smoking', 'Pathology', 'Stage']
X = df[selected_features]
y = df['Recurred']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

model = RandomForestClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)

# Get input values from the user
age = float(input("Enter Age: "))  # Convert age to float
gen = int(input("Enter Gender (0/1): "))  # Convert gender to int
smoke = int(input("Smoking (0/1): "))  # Convert smoking to int (assuming 0 for No and 1 for Yes)
pathology = int(input("Enter Pathology (numerical value): ")) # Convert pathology to corresponding numerical value
stage = int(input("Enter Stage (numerical value): ")) # Convert stage to corresponding numerical value

# Reshape the input to a 2D array with one row and multiple columns
input_data = [[age, gen, smoke, pathology, stage]]

pred = model.predict(input_data)
print(pred)
```

```
2      0
3      0
4      0
      ..
378    1
379    1
380    1
381    1
382    1
Name: Gender, Length: 383, dtype: int64
0      0
1      0
2      0
3      0
4      0
      ..
378    1
379    1
380    1
381    1
382    1
Name: Smoking, Length: 383, dtype: int64
0      2
1      2
2      2
3      2
4      2
      ..
378    3
379    3
380    3
381    1
382    3
Name: Pathology, Length: 383, dtype: int64
0      0
1      0
2      0
3      0
4      0
      ..
378    4
379    4
380    4
381    3
382    3
Name: Stage, Length: 383, dtype: int64
Accuracy:  0.7532467532467533
Enter Age: 10
Enter Gender (0/1): 1
Smoking (0/1): 0
Enter Pathology (numerical value): 1
Enter Stage (numerical value): 6
['Yes']
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Rando
  warnings.warn(
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('/content/dataset.csv thyr.csv')

# Encode categorical variables
label_encoder = LabelEncoder()
df['Gender'] = label_encoder.fit_transform(df['Gender'])
df['Smoking'] = label_encoder.fit_transform(df['Smoking'])
df['Pathology'] = label_encoder.fit_transform(df['Pathology'])
df['Stage'] = label_encoder.fit_transform(df['Stage'])

selected_features = ['Age', 'Gender', 'Smoking', 'Pathology', 'Stage']
X = df[selected_features]
y = df['Recurred']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=44)

model = RandomForestClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)

# Get user input for prediction
age = int(input("Enter Age: "))
gen = int(input("Enter Gender (0-F/1-M): "))
smoke = int(input("Smoking or not (0-NO/1-YES): "))
pathology = int(input("Pathology (encoded value)\n 0-microcapillary \n 1-papillary \n 2-follicular \n 3-hurthelcell: "))
stage = int(input("Stage (encoded value): \n I,II,IVB,IVA"))

# Create a list of user input values
user_input = [[age, gen, smoke, pathology, stage]]

# Make prediction using the trained model
pred = model.predict(user_input)
print("Prediction:", pred[0])
```

```
Accuracy:  0.7532467532467533
Enter Age: 10
Enter Gender (0-F/1-M): 0
Smoking or not (0-NO/1-YES): 1
Pathology (encoded value)
 0-microcapillary
 1-papillary
 2-follicular
 3-hurthelcell: 3
Stage (encoded value):
 I,II,IVB,IVAIVB
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-15-9b21d56fcab1> in <cell line: 0>()
     32 smoke = int(input("Smoking or not (0-NO/1-YES): "))
     33 pathology = int(input("Pathology (encoded value)\n 0-microcapillary \n 1-papillary \n 2-follicular \n 3-hurthelcell: "))
---> 34 stage = int(input("Stage (encoded value): \n I,II,IVB,IVA"))
     35
     36 # Create a list of user input values

ValueError: invalid literal for int() with base 10: 'IVB'
```

Next steps: Explain error

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('/content/dataset.csv thyr.csv')

# Encode categorical variables
label_encoder = LabelEncoder()
df['Gender'] = label_encoder.fit_transform(df['Gender'])
df['Smoking'] = label_encoder.fit_transform(df['Smoking'])
df['Pathology'] = label_encoder.fit_transform(df['Pathology'])
# Fit LabelEncoder on 'Stage' before transforming it
df['Stage'] = label_encoder.fit_transform(df['Stage'])

# Get a list of unique Stage values for user input
stage_values = df['Stage'].unique()
stage_mapping = {i: stage for i, stage in enumerate(label_encoder.classes_)}

selected_features = ['Age', 'Gender', 'Smoking', 'Pathology', 'Stage']
X = df[selected_features]
y = df['Recurred']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=44)

model = RandomForestClassifier()
model.fit(X_train, y_train)

y_pred
```

```
array(['No', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'Yes', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'Yes', 'No',
       'No', 'No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'No', 'No',
       'Yes', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'No',
       'No', 'Yes', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No',
       'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
```

```
       'No', 'No', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No',
       'No', 'Yes', 'Yes', 'No'], dtype=object)
```