```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import KFold, cross_validate
from sklearn.metrics import accuracy_score, precision_score, recall_score
```

```
data = pd.read_csv('Laterality_edited.csv', index_col=0)
data.index = range(data.shape[0])
```

There are 20 columns (19 inputs, 1 output)

35 samples

14 cols, contains categorical data

```
data.describe()
```

	Hipp_Vol_LI	Hipp_FLAIR_LI	Cg_LI	Fx_LI	Hipp_MD_LI	Overall_Laterality_
count	34.000000	34.000000	34.000000	34.000000	34.000000	34.0000
mean	-0.015906	-0.089070	-0.030609	0.015556	-0.000051	0.3823
std	0.062391	0.266247	0.045279	0.032205	0.000191	0.4932
min	-0.129227	-0.558875	-0.140300	-0.046900	-0.000765	0.0000
25%	-0.065504	-0.307987	-0.057650	-0.008400	-0.000136	0.0000
50%	-0.015769	-0.071029	-0.030700	0.020700	-0.000023	0.0000
75%	0.010313	0.079627	-0.000875	0.042300	0.000047	1.0000
max	0.106508	0.499853	0.080900	0.071600	0.000284	1.0000

Input-Output Split

The last column of dataset is the output and the other ones are the inputs

```
X = data.iloc[:, 0:-1]
y = data.iloc[:, -1]
```

The output is binary, so we are facing binary classification task

```
y.unique()
```

```
array([0, 1], dtype=int64)
```

We do not do normalization, since we want to use ada boost with deTree base estimator, I expalined why this is the case in second question.

Dealing with categorical data

since we are going to use deTree as base classifier for ada boost, we don't need to convert categorical data to numerical, but because this task was specified in TA class (I heard so, I'm not sure) we're going to do so

In the next cell, we have two different set of numbers, one reperesnting numerical cols the other for categorical cols

```
num_cols = [5,7,9,11,13]
cat_cols = list(set(range(19)).difference(num_cols))
```

Here we make an ecoder object wich we will use for one hot encoding

```
enc = OneHotEncoder(sparse= False)
```

using the encoder object, here we transformed cat features to one hot features

```
X_cat_oh = pd.DataFrame(enc.fit_transform(X.iloc[:, cat_cols]))
X_num = X.iloc[:, num_cols]
```

now in this cell, we merged the numerical features with one hot features

```
X_oh = pd.concat([X_num, X_cat_oh], axis=1)
```

defining a list of k for k-fold cross validations

```
list_k = [5, 7, 10]
```

Now here we used sklearn cross_validate function to cross validate ada boost with 10,000 base estimators on our data sets, base estimator would be a deTree and during the evaluation we'll evaluate accuracy, precision and recall and F1. we use multi threaded method to reduce training time. -1 means use all available cpu cores

```
list_evals = []
for k in list_k:
    clf = AdaBoostClassifier(n_estimators=10000)
    eval = cross_validate(clf, X_oh, y, cv=k, n_jobs=-1, scoring=['accuracy', 'precision', 'r
    list_evals.append(eval)



for eval in list_evals:
    k = len(eval['fit_time'])
    acc_list = eval['test_accuracy']
    pre_list = eval['test_precision']
    rec_list = eval['test_recall']
    f1_list = eval['test_f1']
    print(f'---- k:{k} ----')
    print(f'Folds acc: {acc_list}')
    print(f'Folds precision: {pre_list}')
    print(f'Folds recall: {rec_list}')
    print(f'Folds F1: {f1_list}')
    print(f'Overal ACC: {np.mean(acc_list)}')
    print(f'Overal Precision: {np.mean(pre_list)}')
    print(f'Overal Recall: {np.mean(rec_list)}')
    print(f'Overal F1: {np.mean(f1_list)}')
```

```
---- k:5 ----
Folds acc: [0.85714286 0.85714286 0.85714286 0.71428571 0.83333333]
Folds precision: [1.         1.         0.75       1.         0.66666667]
Folds recall: [0.5        0.66666667 1.         0.33333333 1.        ]
Folds F1: [0.66666667 0.8        0.85714286 0.5        0.8       ]
Overal ACC: 0.8238095238095238
Overal Precision: 0.8833333333333334
Overal Recall: 0.7
Overal F1: 0.7247619047619048
---- k:7 ----
Folds acc: [1.   1.   1.   1.   0.6  0.8  0.75]
Folds precision: [1.  1.  1.  1.  0.5 1.  0.5]
Folds recall: [1.  1.  1.  1.  0.5 0.5 1. ]
Folds F1: [1.         1.         1.         1.         0.5        0.66666667
 0.66666667]
Overal ACC: 0.8785714285714284
Overal Precision: 0.8571428571428571
Overal Recall: 0.8571428571428571
Overal F1: 0.8333333333333334
---- k:10 ----
Folds acc: [1.         0.75       1.         1.         1.         0.33333333
 0.66666667 0.66666667 0.66666667 1.        ]
Folds precision: [1.  1.  1.  1.  1.  0.  0.5 0.  0.5 1. ]
Folds recall: [1.  0.5 1.  1.  1.  0.  1.  0.  1.  1. ]
Folds F1: [1.         0.66666667 1.         1.         1.         0.
 0.66666667 0.         0.66666667 1.        ]
Overal ACC: 0.8083333333333332
Overal Precision: 0.7
Overal Recall: 0.75
Overal F1: 0.7
```

based on above results, k=7 had the best accuracy, k=5 had best precision and k=7 had the best recall. for this task we are more interested in recall, because bigger recall means smaller false negative, which means more patients with disease classified correctly. In medical cases, classifiying a healthy person as sick is not as dangerous as classifiy a sick patient as healthy, so recall is more important.