```
import pandas as pd
import sys as os
from sklearn.metrics import accuracy_score as acc
from collections import defaultdict
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier as DT

def namely(train,test):
# reading the files train and test
    List1 = []
    List1 = train.readlines()
    train.close()
    List2 = []
    List2 = test.readlines()
    test.close()
   # adding words to the train data
    trainData = defaultdict(list)

    for i in range(0, len(List1)):
        if '. '  in List1[i]:
            Lwords =[]
            L =''
            R = 'EOP'
            # adding the left word to a period
            Lwords = List1[i].split('. ')
            L = Lwords[0].split(' ')[1]
            trainData['Left'].append(L)
```

```
# adding the right word to a period
if(i<len(List1)-1):
    R = List1[i+1].split(' ')[1]
trainData['Right'].append(R)
# left word size
if(len(L)<3):
    trainData['size'].append(1)
else:
    trainData['size'].append(0)
# Left is capital
if(L.isnumeric()):
    trainData['Lcap'].append('NA')
elif(L.istitle()):
    trainData['Lcap'].append(1)
else:
    trainData['Lcap'].append(0)
# R is capital
if(R.isnumeric()):
    trainData['Rcap'].append('NA')
elif(R.istitle()):
    trainData['Rcap'].append(1)
else:
    trainData['Rcap'].append(0)
# dots in left word
if('.' in L):
    trainData['dots'].append(1)
else:
    trainData['dots'].append(0)
```

```
        # quotation in right word
        if("" in R):
            trainData['quotation'].append(1)
        else:
            trainData['quotation'].append(0)
        # left word is a number
        if(L.isnumeric()):
            trainData['Number'].append(1)
        else:
            trainData['Number'].append(0)
       #class Label end of sentence or not
        if 'NEOS'  in Lwords[1]:
            trainData['label'].append(1)
        elif 'EOS' in Lwords[1]:
            trainData['label'].append(0)
        else:
            trainData['label'].append('NA')

testData = defaultdict(list)
for i in range(0, len(List2)):
    if '. '  in List2[i]:
        Lwords =[]
        L =''
        R = 'EOP'
        # Left word to a period
        Lwords = List2[i].split('. ')
        L = Lwords[0].split(' ')[1]
        testData['Left'].append(L)
```

```
# Right word to a period
if(i<len(List2)-1):
    R = List1[i+1].split(' ')[1]
testData['Right'].append(R)
# left word size
if(len(L)<3):
    testData['size'].append(1)
else:
    testData['size'].append(0)
# Left is capital
if(L.isnumeric()):
    testData['Lcap'].append('NA')
elif(L.istitle()):
    testData['Lcap'].append(1)
else:
    testData['Lcap'].append(0)
# Right is capital
if(R.isnumeric()):
    testData['Rcap'].append('NA')
elif(R.istitle()):
    testData['Rcap'].append(1)
else:
    testData['Rcap'].append(0)
# dots in left word
if('.' in L):
    testData['dots'].append(1)
else:
    testData['dots'].append(0)
```

```
        # quotation in right word
        if("" in R):
            testData['quotation'].append(1)
        else:
            testData['quotation'].append(0)

        # left word is a number
        if(L.isnumeric()):
            testData['Number'].append(1)
        else:
            testData['Number'].append(0)
        # class Label end of sentence or not
        if 'NEOS'  in Lwords[1]:
            testData['label'].append(1)
        elif 'EOS' in Lwords[1]:
            testData['label'].append(0)
        else:
            testData['label'].append('NA')
trainData =pd.DataFrame.from_dict(trainData)
testData =pd.DataFrame.from_dict(testData)

#Accuracy calculation for 5 features
trainData['size'] = trainData.index
testData['size'] = testData.index
trainData['Left'] = trainData.index
testData['Left'] = testData.index
trainData['Right'] = trainData.index
testData['Right'] = testData.index
```

```
    trainData['label'] = trainData['label'].map({1: 1, 0: 0, 'NA':-1})
    testData['label'] = testData['label'].map({1: 1, 0: 0, 'NA':-1})
    trainData['Lcap'] = trainData['Lcap'].map({1: 1, 0: 0, 'NA':-1})
    testData['Lcap'] = testData['Lcap'].map({1: 1, 0: 0, 'NA':-1})
    trainData['Rcap'] = trainData['Rcap'].map({1: 1, 0: 0, 'NA':-1})
    testData['Rcap'] = testData['Rcap'].map({1: 1, 0: 0, 'NA':-1})
    trainData['dots'] = trainData['dots'].map({1: 1, 0: 0})
    testData['dots'] = testData['dots'].map({1: 1, 0: 0})
    trainData['quotation'] = trainData['quotation'].map({1: 1, 0: 0})
    testData['quotation'] = testData['quotation'].map({1: 1, 0: 0})
    trainData['Number'] = trainData['Number'].map({1: 1, 0: 0})
    testData['Number'] = testData['Number'].map({1: 1, 0: 0})
    features = ['label','Left','Right','size','Lcap','Rcap']
    X = trainData[features]
    y = trainData['label']
    test_data_X = testData[features]
    test_data_y = testData['label']
    X_train,  X_test, y_train, y_test = train_test_split(X, y,test_size=0.2 )
    # decision tree
    clf = DT()
# Training the classifier
    clf = clf.fit(X_train,y_train)
#classifier prediction
    test_pred=clf.predict(test_data_X)
    accuracy=acc(test_data_y,test_pred)
    print("Accuracy for the 5 features:",accuracy*100)
    # 8 features
    all_features = ['label','Left','Right','size','Lcap','Rcap','dots','quotation','Number']
```

```
X_8 = trainData[all_features]
y_8 = trainData['label']
test_data_X_8 = testData[all_features]
test_data_y_8 = testData['label']
X_train_8,  X_test_8, y_train_8, y_test_8 = train_test_split(X_8, y_8,test_size=0.2 )
 # decision tree
clf8 = DT()
# Training the classifier
clf8 = clf8.fit(X_train_8,y_train_8)
#classifier prediction
test_pred_8=clf8.predict(test_data_X_8)
accuracy8=acc(test_data_y_8,test_pred_8)
print("Accuracy for the 8 features combined:",accuracy8*100)

# 3 features
feature3 = ['dots','quotation','Number']
X3 = trainData[feature3]

y3 = trainData['label']
test_data_X3 = testData[feature3]

test_data_y3 = testData['label']
X_train_3,  X_test_3, y_train_3, y_test_3 = train_test_split(X3, y3,test_size=0.2 )

 # decision tree
clf3 = DT()

# Training the classifier
```

```
    clf3 = clf3.fit(X_train_3,y_train_3)
#classifier prediction



    testpred3=clf3.predict(test_data_X3)
    accuracy3=acc(test_data_y3,testpred3)
    print("Accuracy from the 3 features :",accuracy3*100)



def main():

    input1 = os.argv[1]
    input2 = os.argv[2]
    train = open(input1)
    test = open(input2)
    namely(train,test)



if __name__ == "__main__":
    main()
```