```
# nhập thư viện
import numpy as np
import pandas as pd


# tạo hàm lấy dữ liệu
def loadCsv(filename) -> pd.DataFrame:
    '''Code here'''
    return pd.read_csv(filename)



# tạo hàm biến đổi cột định tính, dùng phương pháp one hot
def transform(data, columns_trans): # data dạng dataframe, data_trans là cột cần biến đổi --> dạng Series, nhiều cột cần biến đổi thì bỏ vào 
    for i in columns_trans:
        unique = data[i].unique() + '-' + i # trả lại mảng
        # tạo ma trận 0
        matrix_0 = np.zeros((len(data), len(unique)), dtype = int)
        frame_0 = pd.DataFrame(matrix_0, columns = unique)
        for index, value in enumerate(data[i]):
            frame_0.at[index, value + '-' + i] = 1
        data[unique] = frame_0
    return data # trả lại data truyền vào nhưng đã bị biến đổi


# tạo hàm scale dữ liệu về [0,1] (min max scaler)
def scale_data(data, columns_scale): # columns_scale là cột cần scale, nếu nhiều bỏ vào list ['a', 'b']
    for i in columns_scale:
        _max = data[i].max()
        _min = data[i].min()
        '''Code here'''
        min_max_scaler = lambda x: round((x - _min) / (_max - _min), 3)  # Tính min-max
        data[i] = data[i].apply(min_max_scaler)
    return data # --> trả về frame


# hàm tính khoảng cách Cosine
def cosine_distance(train_X, test_X): # cả 2 đều dạng mảng
    dict_distance = dict()
    for index, value in enumerate(test_X, start = 1):
        for j in train_X:
            result = np.sqrt(np.sum((j - value)**2))
            if index not in dict_distance:
                dict_distance[index] = [result]
            else:
                dict_distance[index].append(result)
    return dict_distance # {1: [6.0, 5.0], 2: [4.6, 3.1]}


# hàm gán kết quả theo k
def pred_test(k, train_X, test_X, train_y): # train_X, test_X là mảng, train_y là Series
    lst_predict = list()
    dict_distance = cosine_distance(train_X, test_X)
    train_y = train_y.to_frame(name = 'target').reset_index(drop = True) # train_y là frame
    frame_concat = pd.concat([pd.DataFrame(dict_distance), train_y], axis = 1)
    for i in range(1, len(dict_distance) + 1):
        sort_distance = frame_concat[[i, 'target']].sort_values(by = i, ascending = True)[:k] # sắp xếp và lấy k
        target_predict = sort_distance['target'].value_counts(ascending = False).index[0]
        lst_predict.append([i, target_predict])
    return lst_predict
```

Start coding or generate with AI.

Demo qua drug200

```
data = loadCsv('drug200.csv')
data.head()
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	DrugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	DrugY

```
df = transform(data, ['Sex', 'BP', 'Cholesterol']).drop(['Sex', 'BP', 'Cholesterol'], axis = 1)
df
```

	Age	Na_to_K	Drug	F-Sex	M-Sex	HIGH-BP	LOW-BP	NORMAL-BP	HIGH-Cholesterol	NORMAL-Cholesterol
0	23	25.355	DrugY	1	0	1	0	0	1	0
1	47	13.093	drugC	0	1	0	1	0	1	0
2	47	10.114	drugC	0	1	0	1	0	1	0
3	28	7.798	drugX	1	0	0	0	1	1	0
4	61	18.043	DrugY	1	0	0	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...
195	56	11.567	drugC	1	0	0	1	0	1	0
196	16	12.006	drugC	0	1	0	1	0	1	0
197	52	9.894	drugX	0	1	0	0	1	1	0
198	23	14.020	drugX	0	1	0	0	1	0	1
199	40	11.349	drugX	1	0	0	1	0	0	1

200 rows × 10 columns

```
scale_data(df, ['Age', 'Na_to_K']) # không cần gán biến vì nó trả về data đã truyền vào nhưng đã bị scale
df
```

	Age	Na_to_K	Drug	F-Sex	M-Sex	HIGH-BP	LOW-BP	NORMAL-BP	HIGH-Cholesterol	NORMAL-Cholesterol
0	0.136	0.597	DrugY	1	0	1	0	0	1	0
1	0.542	0.213	drugC	0	1	0	1	0	1	0
2	0.542	0.120	drugC	0	1	0	1	0	1	0
3	0.220	0.048	drugX	1	0	0	0	1	1	0
4	0.780	0.368	DrugY	1	0	0	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...
195	0.695	0.166	drugC	1	0	0	1	0	1	0
196	0.017	0.179	drugC	0	1	0	1	0	1	0
197	0.627	0.113	drugX	0	1	0	0	1	1	0
198	0.136	0.242	drugX	0	1	0	0	1	0	1
199	0.424	0.159	drugX	1	0	0	1	0	0	1

200 rows × 10 columns

```
# tạo data_X và target
data_X = df.drop(['Drug'], axis = 1).values
data_y = df['Drug']
```

```
print(data_X)
print(data_y)
```

```
[[0.136 0.597 1.    ... 0.    1.    0.   ]
 [0.542 0.213 0.    ... 0.    1.    0.   ]
 [0.542 0.12  0.    ... 0.    1.    0.   ]
 ...
 [0.627 0.113 0.    ... 1.    1.    0.   ]
 [0.136 0.242 0.    ... 1.    0.    1.   ]
 [0.424 0.159 1.    ... 0.    0.    1.   ]]
0      DrugY
1      drugC
2      drugC
3      drugX
4      DrugY
       ...
195    drugC
196    drugC
197    drugX
198    drugX
199    drugX
Name: Drug, Length: 200, dtype: object
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data_X, data_y, test_size = 0.2, random_state = 0)
```

```
print(X_train)
print(X_test)
print(y_train)
print(y_test)
print(len(X_train), len(X_test), len(y_train), len(y_test))
print(type(y_train))
```

```
[[0.458 0.462 1.    ... 0.    1.    0.   ]
 [0.237 0.206 0.    ... 0.    1.    0.   ]
 [0.271 0.754 0.    ... 0.    1.    0.   ]
 ...
 [0.424 0.12  1.    ... 1.    1.    0.   ]
 [0.898 0.126 0.    ... 0.    1.    0.   ]
 [0.407 0.343 1.    ... 1.    0.    1.   ]]
[[0.136 0.032 0.    1.    0.    1.    0.    1.    0.   ]
 [0.22  0.207 1.    0.    0.    0.    1.    1.    0.   ]
 [0.458 0.43  0.    1.    0.    1.    0.    1.    0.   ]
 [0.085 0.918 0.    1.    1.    0.    0.    0.    1.   ]
 [0.169 0.398 0.    1.    0.    0.    1.    1.    0.   ]
 [0.085 0.169 1.    0.    0.    1.    0.    0.    1.   ]
 [0.119 0.073 1.    0.    0.    0.    1.    1.    0.   ]
 [0.373 0.179 1.    0.    0.    1.    0.    0.    1.   ]
 [0.475 0.285 0.    1.    0.    1.    0.    1.    0.   ]
 [0.678 0.031 0.    1.    0.    0.    1.    0.    1.   ]
 [0.153 0.1   0.    1.    1.    0.    0.    0.    1.   ]
 [0.068 0.616 1.    0.    1.    0.    0.    0.    1.   ]
 [0.119 0.518 1.    0.    1.    0.    0.    0.    1.   ]
 [0.373 0.327 0.    1.    0.    1.    0.    0.    1.   ]
 [0.763 0.24  0.    1.    1.    0.    0.    1.    0.   ]
 [0.441 0.149 0.    1.    0.    1.    0.    1.    0.   ]
 [0.847 0.801 1.    0.    1.    0.    0.    0.    1.   ]
 [0.932 0.445 1.    0.    0.    0.    1.    1.    0.   ]
 [0.407 0.24  0.    1.    0.    1.    0.    0.    1.   ]
 [0.271 0.338 0.    1.    1.    0.    0.    0.    1.   ]
 [0.356 0.288 1.    0.    1.    0.    0.    0.    1.   ]
 [0.78  0.099 0.    1.    0.    0.    1.    1.    0.   ]
 [0.864 0.057 1.    0.    0.    0.    1.    0.    1.   ]
 [0.322 0.208 1.    0.    0.    1.    0.    0.    1.   ]
 [0.39  0.738 1.    0.    0.    1.    0.    0.    1.   ]
 [0.356 0.328 1.    0.    0.    0.    1.    1.    0.   ]
 [0.881 0.301 1.    0.    0.    0.    1.    1.    0.   ]
 [0.712 0.614 1.    0.    0.    0.    1.    0.    1.   ]
 [0.322 0.506 0.    1.    0.    0.    1.    1.    0.   ]
 [0.593 0.188 1.    0.    0.    0.    1.    0.    1.   ]
 [0.915 0.162 0.    1.    0.    1.    0.    0.    1.   ]
 [0.186 0.247 1.    0.    0.    1.    0.    1.    0.   ]
 [0.576 0.076 0.    1.    1.    0.    0.    0.    1.   ]
 [0.542 0.013 1.    0.    0.    0.    1.    0.    1.   ]
 [0.542 0.76  0.    1.    0.    1.    0.    0.    1.   ]
 [0.831 0.459 0.    1.    1.    0.    0.    0.    1.   ]
 [0.288 0.038 1.    0.    0.    0.    1.    1.    0.   ]
 [0.78  0.368 1.    0.    0.    1.    0.    1.    0.   ]
```

```
 [0.39  0.158 1.    0.    1.    0.    0.    0.    1.   ]
 [0.119 0.178 0.    1.    0.    0.    1.    1.    0.   ]]
134    DrugY
66     drugA
26     DrugY
113    drugX
168    DrugY
       ...
67     drugX
192    DrugY
117    drugX
47     drugC
```

```
test_pred = pred_test(6, X_train, X_test, y_train)
df_test_pred = pd.DataFrame(test_pred).drop([0], axis = 1)
df_test_pred.index = range(1, len(test_pred) + 1)
df_test_pred.columns = ['Predict']


df_actual = pd.DataFrame(y_test)
df_actual.index = range(1, len(y_test) + 1)
df_actual.columns = ['Actual']


pd.concat([df_test_pred, df_actual], axis = 1)
```

	Predict	Actual
1	drugC	drugC
2	drugX	drugX
3	DrugY	DrugY
4	DrugY	DrugY
5	DrugY	DrugY
6	DrugY	drugX
7	drugX	drugX
8	drugX	drugX
9	DrugY	DrugY
10	drugX	drugX

13	DrugY	DrugY
14	drugX	DrugY
15	drugB	drugB
16	drugC	drugC
17	DrugY	DrugY
18	drugX	DrugY
19	drugX	drugX