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


# Load dữ liệu
def loadExcel(filename) -> pd.DataFrame:
    '''Code here'''
    if filename.endswith('.xls'):
        # Dùng engine 'xlrd' cho file .xls
        return pd.read_excel(filename, engine='xlrd')
    elif filename.endswith('.xlsx'):
        # Dùng engine 'openpyxl' cho file .xlsx
        return pd.read_excel(filename, engine='openpyxl')
    else:
        raise ValueError("Unsupported file format. Please use .xls or .xlsx file.")



# tạo tập train test (chia data_train (gộp X_train và y_train) và X_test và y_test)
def splitTrainTest(data, target, ratio = 0.25): # data --> frame
    from sklearn.model_selection import train_test_split
    data_X = data.drop([target], axis = 1)
    data_y = data[[target]]
    '''Code here'''
    X_train, X_test, y_train, y_test = train_test_split(data_X, data_y, test_size=ratio, random_state=42)

    # Gộp X_train và y_train thành data_train
    data_train = pd.concat([X_train, y_train], axis=1)

    return data_train, X_test, y_test # đều là dạng frame


# hàm tính trung bình của từng lớp trong biến target
def mean_class(data_train, target): # tên cột target, data_train là dạng pandas
    df_group = data_train.groupby(by = target).mean() # tất cả các cột đều dạng số, --> frame # sắp xếp theo bảng chữ cái tăng dần(mặc định)
    return df_group # kết quả là dataframe


# hàm dự đoán dùng khoảng cách euclid
def target_pred(data_group, data_test): # data_test ở dạng mảng, data_group là đã đem tính trung bình các lớp(là df_group)
    dict_ = dict()
    for index, value in enumerate(data_group.values):
        result = np.sqrt(np.sum(((data_test - value)**2), axis = 1)) # khoảng cách euclid
        if index in dict_:
            dict_.append(result)
        else:
            dict_[index] = result # Lưu ý chỗ này không phải là [result] vì result là dạng mảng, nếu thêm vào vậy thì nó chỉ có một phần tử.
    # dict_ kết quả dạng {0: [2.0], 1: [1.0540925533894596]}
    '''Code here'''
    df = pd.DataFrame(dict_)

    return df.idxmin(axis = 1) # hàm này tìm cột chứa giá trị nhỏ nhất

##### Có thể phát triển: cho thêm một tham số metric vào hàm, nếu là euclid thì dùng khoảng cách euclid, manhattan thì dùng khoảng cách manha


### Demo bằng ví dụ Iris


data = loadExcel('Iris.xls')
data
```

	sepallength	sepalwidth	petallength	petalwidth	iris
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

```
data_train, X_test, y_test = splitTrainTest(data, 'iris', ratio = 0.3)
print(data_train)
print(X_test)
print(y_test)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[31], line 1
----> 1 data_train, X_test, y_test = splitTrainTest(data, 'iris', ratio = 0.3)
      2 print(data_train)
      3 print(X_test)

Cell In[26], line 3, in splitTrainTest(data, target, ratio)
      2 def splitTrainTest(data, target, ratio = 0.25): # data --> frame
----> 3     from sklearn.model_selection import train_test_split
      4     data_X = data.drop([target], axis = 1)
      5     data_y = data[[target]]

ModuleNotFoundError: No module named 'sklearn'
```

```
df_group = mean_class(data_train, 'iris')
df_group
```

```
#  tính khoảng cách và trả về kết quả lớp có khoảng cách gần nhất
df1 = pd.DataFrame(target_pred(df_group, X_test.values), columns = ['Predict'])
df1
```

```
# set index y_test để nối 2 frame
y_test.index = range(0, len(y_test))
y_test.columns = ['Actual']
y_test
```

```
df2 = pd.DataFrame(y_test)
df2
```

```
pd.concat([df1, df2],axis = 1)
```