k-Nearest Neighbors

import dataset

```
import pandas as pd
df = pd.read_csv('teleCust1000t.csv')
df.head()
```

	region	tenure	age	marital	address	income	ed	employ	retire	gender	reside	custcat
0	2	13	44	1	9	64.0	4	5	0.0	0	2	1
1	3	11	33	1	7	136.0	5	5	0.0	0	6	4
2	3	68	52	1	24	116.0	1	29	0.0	1	2	3
3	2	33	33	0	12	33.0	2	0	0.0	1	1	1
4	2	23	30	1	9	30.0	1	2	0.0	0	4	3

```
# df.info()
```

```
# df.describe()
```

```
df['custcat'].value_counts()
```

```
custcat
3    281
1    266
4    236
2    217
Name: count, dtype: int64
```

cleaning

```
# clean the data
```

encoding

```
# encode the data
```

define x , y

```
import numpy as np
x = df.loc[: , 'region':'reside'] .values
y = df['custcat'].values
```

```
x[:5]
# y[:5]
```

```
array([[  2.,  13.,  44.,   1.,   9.,  64.,   4.,   5.,   0.,   0.,   2.],
       [  3.,  11.,  33.,   1.,   7., 136.,   5.,   5.,   0.,   0.,   6.],
       [  3.,  68.,  52.,   1.,  24., 116.,   1.,  29.,   0.,   1.,   2.],
       [  2.,  33.,  33.,   0.,  12.,  33.,   2.,   0.,   0.,   1.,   1.],
       [  2.,  23.,  30.,   1.,   9.,  30.,   1.,   2.,   0.,   0.,   4.]])
```

spliting

```
### finding best random state

# from sklearn.model_selection import train_test_split
# from sklearn.preprocessing import StandardScaler
# from sklearn.neighbors import KNeighborsClassifier
```

```
# from sklearn.metrics import accuracy_score

# import time
# t1 = time.time()
# lst = []
# for i in range(1,10):
#     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=i)
#     sc = StandardScaler().fit(x_train)
#     x_train = sc.transform(x_train)
#     x_test = sc.transform(x_test)
#     knc = KNeighborsClassifier(n_neighbors = 4)
#     knc.fit(x_train,y_train)
#     yhat_test = knc.predict(x_test)
#     acc = accuracy_score(y_test, yhat_test)
#     lst.append(acc)
# t2 = time.time()
# print(f"run time: {round((t2 - t1) / 60 , 0)} min")
# print(f"accuracy_score = {round(max(lst),2)}")
# print(f"random_state = {np.argmax(lst) + 1}")

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42)
```

scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler().fit(x_train)
x_train = sc.transform(x_train)
x_test = sc.transform(x_test)
```

fit train data

```
### finding best n

# acc_list = []
# for n in range(1,10):
#     knc = KNeighborsClassifier(n_neighbors = n)
#     knc.fit(x_train,y_train)
#     yhat_test = knc.predict(x_test)
#     acc = accuracy_score(y_test, yhat_test)
#     acc_list.append(acc)
# acc_list

from sklearn.neighbors import KNeighborsClassifier
knc = KNeighborsClassifier(n_neighbors = 4) # metric='minkowski', p=1 (manhattan) # metric='minkowski', p=
knc.fit(x_train,y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=4)
```

predict test data

```
yhat_test = knc.predict(x_test)
print(y_test[:5])
print(yhat_test[:5])
```

```
[2 1 2 3 1]
[4 2 3 2 3]
```

evaluate model

```
from sklearn.metrics import accuracy_score
print("Accuracy_score (train data): ", accuracy_score(y_train, knc.predict(x_train)))
print("Accuracy_score (test data): ", accuracy_score(y_test, yhat_test))
```

```
Accuracy_score (train data):  0.548
Accuracy_score (test data):  0.304
```

predict new data

```
knc.predict(sc.transform([[1.,  68.,  41.,   1.,  21.,  72.,   1.,  22.,   0.,   0.,   3.]]))
```

```
array([3])
```

save the model

```
# import joblib
# joblib.dump(knc, 'knc_model.pkl')
```

load the model

```
# import joblib
# knc = joblib.load('knc_model.pkl')
```