Logistic Regression

import dataset

```
import pandas as pd
df = pd.read_csv("ChurnData.csv")
df.head()
```

	tenure	age	address	income	ed	employ	equip	callcard	wireless	longmon	...	pager	internet	callwait	confer	ebill	logl
0	11.0	33.0	7.0	136.0	5.0	5.0	0.0	1.0	1.0	4.40	...	1.0	0.0	1.0	1.0	0.0	1.4
1	33.0	33.0	12.0	33.0	2.0	0.0	0.0	0.0	0.0	9.45	...	0.0	0.0	0.0	0.0	0.0	2.2
2	23.0	30.0	9.0	30.0	1.0	2.0	0.0	0.0	0.0	6.30	...	0.0	0.0	0.0	1.0	0.0	1.8
3	38.0	35.0	5.0	76.0	2.0	10.0	1.0	1.0	1.0	6.05	...	1.0	1.0	1.0	1.0	1.0	1.8
4	7.0	35.0	14.0	80.0	2.0	15.0	0.0	1.0	0.0	7.10	...	0.0	0.0	1.0	1.0	0.0	1.9

5 rows × 28 columns

```
df = df[['tenure', 'age', 'address', 'income', 'ed', 'employ', 'equip','churn']]
df.head()
```

	tenure	age	address	income	ed	employ	equip	churn
0	11.0	33.0	7.0	136.0	5.0	5.0	0.0	1.0
1	33.0	33.0	12.0	33.0	2.0	0.0	0.0	1.0
2	23.0	30.0	9.0	30.0	1.0	2.0	0.0	0.0
3	38.0	35.0	5.0	76.0	2.0	10.0	1.0	0.0
4	7.0	35.0	14.0	80.0	2.0	15.0	0.0	0.0

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 8 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   tenure   200 non-null    float64
 1   age      200 non-null    float64
 2   address  200 non-null    float64
 3   income   200 non-null    float64
 4   ed       200 non-null    float64
 5   employ   200 non-null    float64
 6   equip    200 non-null    float64
 7   churn    200 non-null    float64
dtypes: float64(8)
memory usage: 12.6 KB
```

```
df['churn'] = df['churn'].astype('int')
```

cleaning

```
# clean the data
```

encoding

```
# encode the data
```

define x , y

```
import numpy as np
x = np.array(df[['tenure', 'age', 'address', 'income', 'ed', 'employ', 'equip']])
```

```
x[0:5]
```

```
array([[ 11.,  33.,   7., 136.,   5.,   5.,   0.],
       [ 33.,  33.,  12.,  33.,   2.,   0.,   0.],
       [ 23.,  30.,   9.,  30.,   1.,   2.,   0.],
       [ 38.,  35.,   5.,  76.,   2.,  10.,   1.],
       [  7.,  35.,  14.,  80.,   2.,  15.,   0.]])
```

```
y = np.array(df['churn'])
y[0:5]
```

```
array([1, 1, 0, 0, 0])
```

spliting

```
### finding best random state

# from sklearn.model_selection import train_test_split
# from sklearn.linear_model import LogisticRegression
# from sklearn.preprocessing import StandardScaler
# 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)
#     LR = LogisticRegression()
#     LR.fit(x_train,y_train)
#     yhat_test = LR.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=4)
```

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

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression() # C=0.01, solver='liblinear'
LR.fit(x_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

predict test data

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

```
print(y_test[:5])
```

```
[0 0 0 0 0]
[0 0 1 0 1]
```

```
# pd.concat([pd.DataFrame(y_test), pd.DataFrame(yhat_test)], axis=1)
```

```
yhat_prob = LR.predict_proba(x_test)
yhat_prob[:5]
```

```
array([[0.74394608, 0.25605392],
       [0.9311592 , 0.0688408 ],
       [0.83203409, 0.16796591],
       [0.94658334, 0.05341666],
       [0.8349031 , 0.1650969 ]])
```

evaluation

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

```
Accuracy_score (train data):  0.7733333333333333
Accuracy_score (test data):  0.76
```

```
from sklearn.metrics import confusion_matrix, classification_report, jaccard_score, log_loss
```

```
confusion_matrix(y_test, yhat_test)
```

```
array([[34,  0],
       [12,  4]])
```

```
print(classification_report(y_test, yhat_test))
```

```
              precision    recall  f1-score   support

           0       0.74      1.00      0.85        34
           1       1.00      0.25      0.40        16

    accuracy                           0.76        50
   macro avg       0.87      0.62      0.62        50
weighted avg       0.82      0.76      0.71        50
```

```
jaccard_score(y_test, yhat_test, pos_label=0)  #def 1
```

```
np.float64(0.7391304347826086)
```

```
log_loss(y_test, yhat_prob)
```

```
0.5450336318242056
```

predicting new data

```
LR.predict(sc.transform([[11.0, 33.0, 7.0, 136.0, 5.0, 5.0, 0.0]]))
```

```
array([0])
```

save the model

```
# import joblib
# joblib.dump(LR, 'LR_model.pkl')
```

load the model

```
# import joblib
# LR = joblib.load('LR_model.pkl')
```