CS353 Machine Learning Lab

KNN classification (05/03/21)

Shumbul Arifa (181CO152)

Task:

Perform KNN model for credit scoring system dataset.

Dataset

We are using the credit card defaulters dataset.

Loading dataset

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/00350/default%20of%20credit%20card%20clients.xls"
dataset = pd.read_excel(url,skiprows=[0])

df = pd.DataFrame(dataset)
print('\nThe shape of the dataset is', df.shape)
print('The first five tuples from the dataset are:')
df.head()
```

```
The shape of the dataset is (30000, 25)
The first five tuples from the dataset are:
```

Out[2]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AM
0	1	20000	2	2	1	24	2	2	-1	-1	...	0	0	0	
1	2	120000	2	2	2	26	-1	2	0	0	...	3272	3455	3261	
2	3	90000	2	2	2	34	0	0	0	0	...	14331	14948	15549	1
3	4	50000	2	2	1	37	0	0	0	0	...	28314	28959	29547	2
4	5	50000	1	2	1	57	-1	0	-1	0	...	20940	19146	19131	2

5 rows × 25 columns

Data analysis and preprocessing

In [3]:
```
df.describe()
```

Out[3]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000
mean	15000.500000	167484.322667	1.603733	1.853133	1.551867	35.485500	-0.016700	-0.133767	-0.166200
std	8660.398374	129747.661567	0.489129	0.790349	0.521970	9.217904	1.123802	1.197186	1.196868
min	1.000000	10000.000000	1.000000	0.000000	0.000000	21.000000	-2.000000	-2.000000	-2.000000
25%	7500.750000	50000.000000	1.000000	1.000000	1.000000	28.000000	-1.000000	-1.000000	-1.000000
50%	15000.500000	140000.000000	2.000000	2.000000	2.000000	34.000000	0.000000	0.000000	0.000000
75%	22500.250000	240000.000000	2.000000	2.000000	2.000000	41.000000	0.000000	0.000000	0.000000
max	30000.000000	1000000.000000	2.000000	6.000000	3.000000	79.000000	8.000000	8.000000	8.000000

8 rows × 25 columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ID                          30000 non-null  int64
 1   LIMIT_BAL                   30000 non-null  int64
 2   SEX                         30000 non-null  int64
 3   EDUCATION                   30000 non-null  int64
 4   MARRIAGE                    30000 non-null  int64
 5   AGE                         30000 non-null  int64
 6   PAY_0                       30000 non-null  int64
 7   PAY_2                       30000 non-null  int64
 8   PAY_3                       30000 non-null  int64
 9   PAY_4                       30000 non-null  int64
 10  PAY_5                       30000 non-null  int64
 11  PAY_6                       30000 non-null  int64
 12  BILL_AMT1                   30000 non-null  int64
 13  BILL_AMT2                   30000 non-null  int64
 14  BILL_AMT3                   30000 non-null  int64
 15  BILL_AMT4                   30000 non-null  int64
 16  BILL_AMT5                   30000 non-null  int64
 17  BILL_AMT6                   30000 non-null  int64
 18  PAY_AMT1                    30000 non-null  int64
 19  PAY_AMT2                    30000 non-null  int64
 20  PAY_AMT3                    30000 non-null  int64
 21  PAY_AMT4                    30000 non-null  int64
 22  PAY_AMT5                    30000 non-null  int64
 23  PAY_AMT6                    30000 non-null  int64
 24  default payment next month  30000 non-null  int64
dtypes: int64(25)
memory usage: 5.7 MB
```

In [5]:
```
X = df.drop('default payment next month',axis=1).values
y = df['default payment next month'].values
```

In [6]:
```
import sklearn.preprocessing as preprocessing
```

In [7]:
```
standard_scaler = preprocessing.StandardScaler()
X = standard_scaler.fit(X).transform(X)
X.shape
```

Out[7]:
```
(30000, 24)
```

Split data

In [8]:
```
from sklearn.model_selection import train_test_split
```

In [9]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=512, stratify=y)
X.shape
```

Out[9]:
```
(30000, 24)
```

In [16]:
```
X_train
```

Out[16]:
```
array([[ 0.29000304, -1.05964618,  0.81016074, ..., -0.21231319,
        -0.21595606, -0.10775071],
       [-1.66813813,  0.55890707,  0.81016074, ..., -0.30806256,
        -0.31413612, -0.29338206],
       [ 1.58442234,  1.09842483,  0.81016074, ..., -0.0527309 ,
        -0.09813998, -0.01212243],
       ...,
       [-0.17511034, -0.52012843,  0.81016074, ..., -0.18039673,
        -0.18990561, -0.18087821],
       [ 1.36756958,  1.48379465, -1.23432296, ..., -0.30806256,
        -0.31413612, -0.29338206],
       [ 1.47368656, -1.21379411,  0.81016074, ..., -0.11656381,
        -0.31413612, -0.07568711]])
```

In [22]:
```
X_test
```

Out[22]:
```
array([[ 1.56767919, -0.44305446,  0.81016074, ..., -0.30806256,
        -0.14068467, -0.11900109],
       [ 1.4203394 , -0.13475861, -1.23432296, ..., -0.15218258,
        -0.14853908, -0.15061467],
       [ 1.46306332, -0.44305446,  0.81016074, ...,  1.92608952,
        -0.22727949, -0.20900417],
       ...,
       [ 1.00972789, -0.90549825,  0.81016074, ...,  0.05348708,
         0.06294077, -0.07591212],
       [ 0.27707039, -0.82842429,  0.81016074, ..., -0.11656381,
        -0.22904673, -0.20900417],
       [ 0.65408012,  1.56086861, -1.23432296, ...,  0.48103995,
         0.4716971 ,  0.24410508]])
```

Implementing KNN Classifier

k-NN is a type of classification where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then normalizing the training data can improve its accuracy dramatically.

In [10]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 15)
```

In [11]:
```
#model training
knn.fit(X_train, y_train)
```

Out[11]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=15, p=2,
                     weights='uniform')
```

In [20]:
```
#model prediction
y_predict = knn.predict(X_test)
y_predict
```

Out[20]:
```
array([1, 0, 0, ..., 0, 0, 0])
```

In [13]:
```
#accuracy computation
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
print("Accuracy: ", accuracy_score(y_test, y_predict))
print("Percentage Accuracy: ", accuracy_score(y_test, y_predict) * 100)

print("Confusion matrix")
print(confusion_matrix(y_test, y_predict))
```

```
Accuracy:  0.8126666666666666
Percentage Accuracy:  81.26666666666667
Confusion matrix
[[6676  333]
 [1353  638]]
```

In [21]:
```
print("Classification matrix:\n", classification_report(y_test,y_predict))
```

```
Classification matrix:
              precision    recall  f1-score   support

           0       0.83      0.95      0.89      7009
           1       0.66      0.32      0.43      1991

    accuracy                           0.81      9000
   macro avg       0.74      0.64      0.66      9000
weighted avg       0.79      0.81      0.79      9000
```