《人工智能导论》机器学习实践性大作业

利用 ScikitLearn 库进行电信客户流失预测

姓　　名　李文涛

学　　号　320200928101

专　　业　电子信息基地班

年　　级　2020 级

利用 ScikitLearn 库进行电信客户流失预测

中文摘要

本文首先以 AMI 码为对比对象介绍 HDB_3 的由来，简略介绍该编码方案的优缺点，接着利用 MATLAB 进行编程，完成对原数字信号序列进行 HDB_3 编码以及后续对信号的频谱分析，对编码时频域分析进行谱图可视化，并同时分析其在其传输速率和频谱利用率上的特点。最后根据其频谱特性分析其应用价值。

关键词： 优缺点,MATLAB, 时频域分析, 应用分析

Abstract

This paper introduces the advantages and advantages of HDB_3, and then uses MATLAB to program, complete the HDB_3 encoding of the original digital signal sequence and the subsequent spectrum analysis of the signal, and the spectral diagram visualization of the encoding time frequency domain analysis, and analyze its characteristics in its transmission rate and frequency utilization. Finally, its application value is analyzed according to its spectral characteristics.

Key Words: Advantages and disadvantages, time frequency domain analysis, application analysis

第一部分　理解数据

1.1　导入数据

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data_train=pd.read_csv("./churn-bigml-80.csv")
data_test=pd.read_csv("./churn-bigml-20.csv")
```

将项目处理所需的库和数据集导入。

1.2　浏览数据集

```
data_train.head(5)
```

用.head() 取前五个数据，以用来对数据所包含的信息进行理解。

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	Total night calls	Total night charge	Total intl minutes	Total intl calls	Total intl charge	Customer service calls	Churn
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10.0	3	2.70	1	False
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3	3.70	1	False
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7.32	12.2	5	3.29	0	False
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9	88	5.26	196.9	89	8.86	6.6	7	1.78	2	False
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3	2.73	3	False

图 1.1　训练集数据实例

1.3　对数据属性的理解

通过上一步我们发现该数据集有 State、Account length、International plan 等一系列属性，通过查阅相关资料，各属性所包含的意思在下方说明。

State：用户的地区

Account length：用户账户长度

Area code：地区代码

International plan：用户是否订阅国际套餐

Voice mail plan：用户是否开通语音短信

Number vmail messages：用户是否开通数字邮件

Total day minutes：用户白天通话时长
Total day calls：用户白天通话次数
Total day charge：用户白天通话话费
Total eve minutes：用户傍晚通话时长
Total eve calls：用户傍晚通话次数
Total eve charge：用户傍晚通话话费
Total night minutes：用户夜间通话时长
Total night calls：用户夜间通话次数
Total night charge：用户夜间通话话费
Total intl minutes：用户国际通话时长
Total intl calls：用户国际通话次数
Total intl charge：用户国际通话话费
Customer service calls：用户白天通话时长
Churn：该客户是否流失

第二部分　数据清洗

2.1　查看训练集详细信息

```
print(data_train.describe())
```

```
       Account length    Area code  Number vmail messages  Total day minutes  \
count     2666.000000  2666.000000            2666.000000         2666.00000
mean       100.620405   437.438860               8.021755          179.48162
std         39.563974    42.521018              13.612277           54.21035
min          1.000000   408.000000               0.000000            0.00000
25%         73.000000   408.000000               0.000000          143.40000
50%        100.000000   415.000000               0.000000          179.95000
75%        127.000000   510.000000              19.000000          215.90000
max        243.000000   510.000000              50.000000          350.80000

       Total day calls  Total day charge  Total eve minutes  Total eve calls  \
count      2666.000000       2666.000000        2666.000000      2666.000000
mean        100.310203         30.512404         200.386159       100.023631
std          19.988162          9.215733          50.951515        20.161445
min           0.000000          0.000000           0.000000         0.000000
25%          87.000000         24.380000         165.300000        87.000000
50%         101.000000         30.590000         200.900000       100.000000
75%         114.000000         36.700000         235.100000       114.000000
max         160.000000         59.640000         363.700000       170.000000

       Total eve charge  Total night minutes  Total night calls  \
count       2666.000000          2666.000000        2666.000000
mean          17.033072           201.168942         100.106152
std            4.330864            50.780323          19.418459
min            0.000000            43.700000          33.000000
...
25%            2.300000                1.000000
50%            2.750000                1.000000
75%            3.270000                2.000000
max            5.400000                9.000000
```

图 2.1　训练集各数据属性分析

2.2　查看数据缺失情况

```
print(data_train.info())
```

```
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   State                   2666 non-null   object
 1   Account length          2666 non-null   int64
 2   Area code               2666 non-null   int64
 3   International plan      2666 non-null   object
 4   Voice mail plan         2666 non-null   object
 5   Number vmail messages   2666 non-null   int64
 6   Total day minutes       2666 non-null   float64
 7   Total day calls         2666 non-null   int64
 8   Total day charge        2666 non-null   float64
 9   Total eve minutes       2666 non-null   float64
 10  Total eve calls         2666 non-null   int64
 11  Total eve charge        2666 non-null   float64
 12  Total night minutes     2666 non-null   float64
 13  Total night calls       2666 non-null   int64
 14  Total night charge      2666 non-null   float64
 15  Total intl minutes      2666 non-null   float64
 16  Total intl calls        2666 non-null   int64
 17  Total intl charge       2666 non-null   float64
 18  Customer service calls  2666 non-null   int64
 19  Churn                   2666 non-null   bool
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 398.5+ KB
None
```

图 2.2　训练集各数据属性分析

经观察上图我们发现，各属性数据均完整，下面我们对这些数据进行部分处理。

2.3 数据处理

经查阅资料，我们采用以下的代码对数据字符串的部分进行 01 编码处理。

```
    # 对数据集中的字符串数据进行编码处理
    gender_class1 = {'No': 0, 'Yes': 1}
    data_train['Voice mail plan'] = data_train['Voice mail plan'].map(
gender_class1)
    data_test['Voice mail plan'] = data_test['Voice mail plan'].map(
gender_class1)
    gender_class2 = {'No': 0, 'Yes': 1}
    data_train['International plan'] = data_train['International plan'].
map(gender_class2)
    data_test['International plan'] = data_test['International plan'].map(
gender_class2)

```

将 Voice mail plan 属性值为 No 的改为 0，属性值为 Yes 的改为 1。

将 International plan 属性值为 No 的改为 0，属性值为 Yes 的改为 1。

第三部分　数据分析

下面对处理后的数据相关系数热力图进行观察比较，选取我们所需要的对用户是否流失的判断属性。

PS：(由于数据属性过多，这里我们挑选具有代表性的几个属性进行关系分析)

```
df = data_train[['Voice mail plan','International plan','Account length',
'Area code', 'Number vmail messages', 'Total day minutes', 'Total day
calls', 'Total day charge', 'Total eve minutes','Churn']]
# 属性间相关系数
cor = df.corr()
print(cor)
# 属性间相关系数热力图
sns.heatmap(cor,cmap="Greens")
plt.show()
```

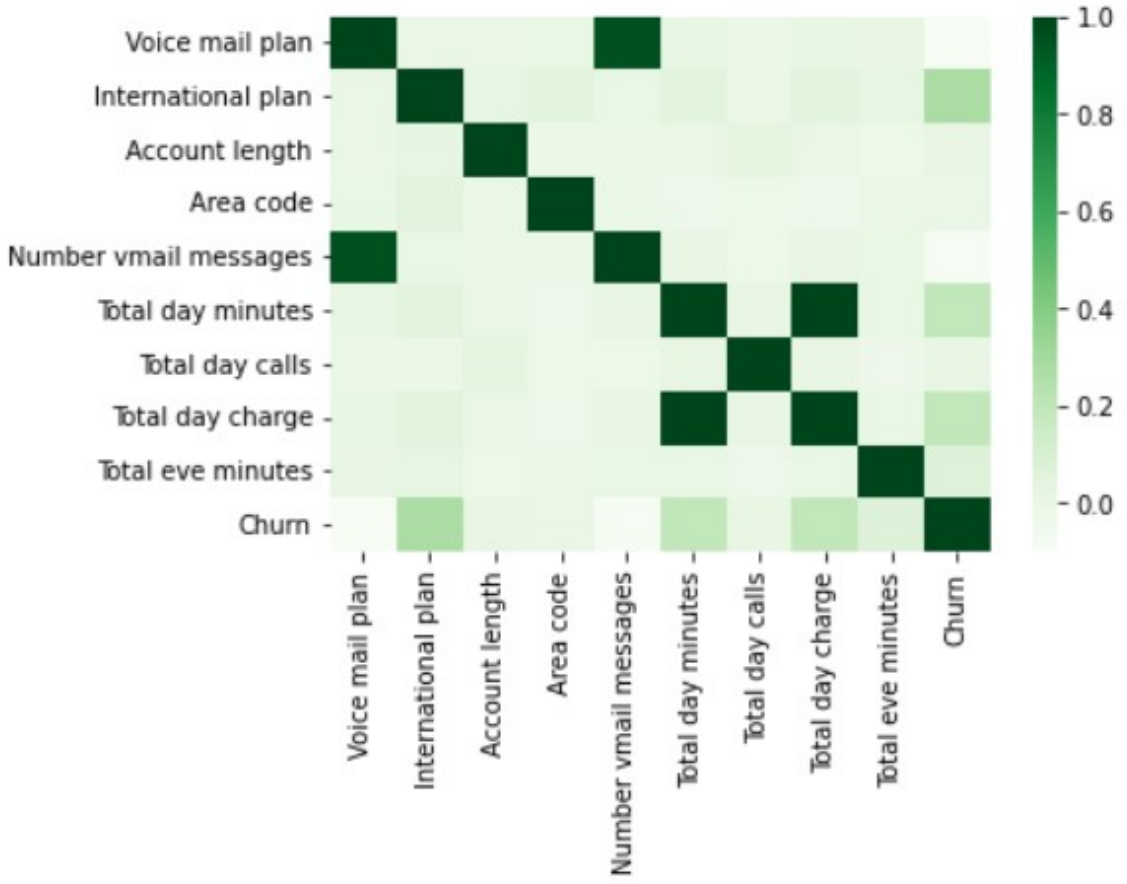


图 3.1　训练集各数据属性关系分析热力图

	Voice mail plan	International plan	Account length	\
Voice mail plan	1.000000	0.002131	0.002448	
International plan	0.002131	1.000000	0.024500	
Account length	0.002448	0.024500	1.000000	
Area code	0.007180	0.047099	-0.008620	
Number vmail messages	0.957159	0.005858	-0.002996	
Total day minutes	0.013438	0.049550	0.002847	
Total day calls	-0.007541	-0.004277	0.038862	
Total day charge	0.013439	0.049555	0.002843	
Total eve minutes	0.019132	0.026616	-0.015923	
Churn	-0.099291	0.277489	0.017728	

	Area code	Number vmail messages	Total day minutes	\
Voice mail plan	0.007180	0.957159	0.013438	
International plan	0.047099	0.005858	0.049550	
Account length	-0.008620	-0.002996	0.002847	
Area code	1.000000	-0.000584	-0.023134	
Number vmail messages	-0.000584	1.000000	0.019027	
Total day minutes	-0.023134	0.019027	1.000000	
Total day calls	-0.009629	-0.009622	0.016780	
Total day charge	-0.023130	0.019027	1.000000	
Total eve minutes	0.000679	0.011401	0.003999	
Churn	0.001019	-0.086474	0.195688	

	Total day calls	Total day charge	Total eve minutes	\
...				
Total day calls	0.018290			
Total day charge	0.195689			
Total eve minutes	0.072906			
Churn	1.000000			

图 3.2 训练集各数据属性关系分析表

通过观察发现，我们发现各属性的相关系数都大致相仿，故我们索性直接将所有属性 (不包括 State) 全部作为训练数据，送入模型构建中。

第四部分 构建模型及评估

4.1 构建模型前的准备

我们首先对训练集进行划分，将模型构建的标签和输出 (即 x 和 y) 准备好。

```
data_train = data_train.drop(columns=['State'])
data_test = data_test.drop(columns=['State'])
```

首先将无用的 State 从数据集中去除。

```
label_train = data_train.loc[:, 'Churn']
data_train = data_train.drop(columns=['Churn'])
```

将 Churn 作为输出目标，除 Churn 的以外属性作为判断依据。

```
train_X=data_train
train_y=label_train
```

将构建模型所需的包导入。

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
```

4.2 模型构建及评估

我们分别采用朴素贝叶斯、决策树、多层感知机来进行模型构建，并分别对模型进行评估。

4.2.1 朴素贝叶斯

```
# Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB

gaussian = GaussianNB()
gaussian.fit(train_X, train_y)
```

将划分好的数据导入调出模型构建包的函数中，得到模型。

```
pred_gauss = gaussian.predict(test_X)
score = gaussian.score(test_X,test_y)
```

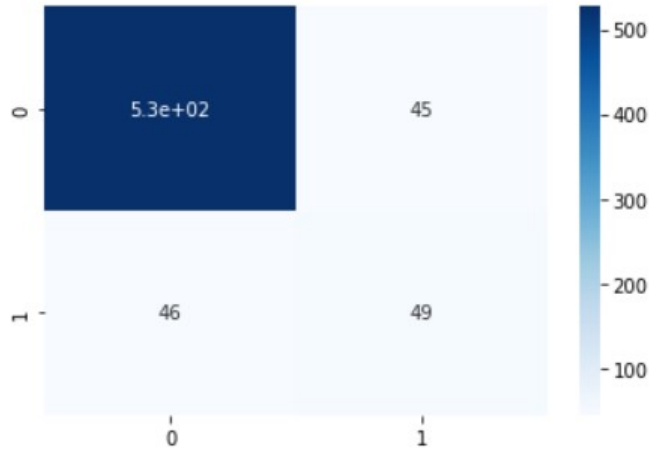
```
print(score)
```

```
0.863568215892054
```

图 4.1 朴素贝叶斯模型得分

对模型的可行性进行评估。

```
cm=confusion_matrix(test_y, pred_gauss)
TP=cm[0,0]
FP=cm[0,1]
FN=cm[1,0]
TN=cm[1,1]
sns.heatmap(cm,annot=True,cmap='Blues')
plt.show()
Accuracy = (TP+TN)/(TP+FP+TN+FN)
Precision = TP/(TP+FP)
Recall = TP/(TP+FN)
print("准确率: ",Accuracy)
print("精准率: ",Precision)
print("召回率: ",Recall)
```




```
准确率: 0.863568215892054
精准率: 0.9213286713286714
召回率: 0.9197207678883071
```

依次输出该模型的混淆矩阵、准确率、精准率、召回率。

4.2.2 决策树

```
#DecisionTree
from sklearn.tree import DecisionTreeClassifier

decisiontree = DecisionTreeClassifier(max_depth=4)
decisiontree.fit(train_X, train_y)
```

将划分好的数据导入调出模型构建包的函数中，得到模型。

```
pred_Tree = decisiontree.predict(test_X)
acc_decisiontree = decisiontree.score(test_X, test_y)
print(acc_decisiontree)
```

```
0.9310344827586207
```

图 4.2 决策树模型得分

对模型的可行性进行评估。

```
cm=confusion_matrix(test_y, pred_Tree)
TP=cm[0,0]
FP=cm[0,1]
FN=cm[1,0]
TN=cm[1,1]
sns.heatmap(cm,annot=True,cmap='Blues')
plt.show()
Accuracy = (TP+TN)/(TP+FP+TN+FN)
Precision = TP/(TP+FP)
Recall = TP/(TP+FN)
print("准确率: ",Accuracy)
print("精准率: ",Precision)
print("召回率: ",Recall)
```

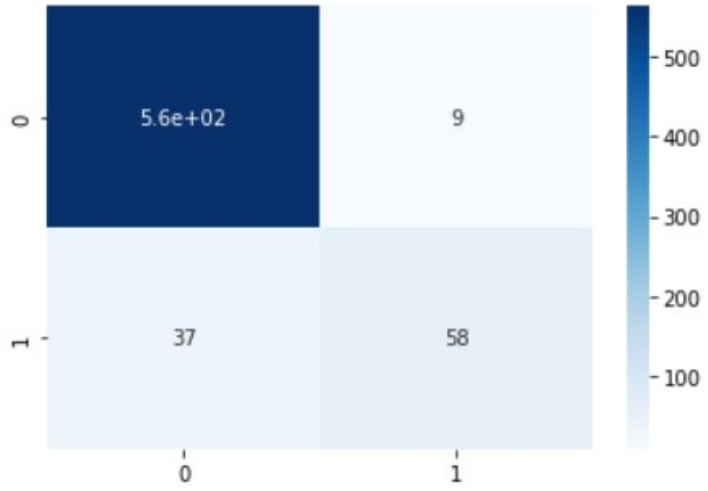
依次输出该模型的混淆矩阵、准确率、精准率、召回率。

4.2.3 多层感知机

```
# MLP
from sklearn.neural_network import MLPClassifier

mlpc = MLPClassifier(activation='tanh',solver='adam',max_iter=400)
mlpc.fit(train_X, train_y)
```

将划分好的数据导入调出模型构建包的函数中，得到模型。




```
准确率:  0.9310344827586207
精准率:  0.9842657342657343
召回率:  0.9383333333333334
```

```python
pred_MLP = mlpc.predict(test_X)
acc_gbk = mlpc.score(test_X, test_y)
print(acc_gbk)
```

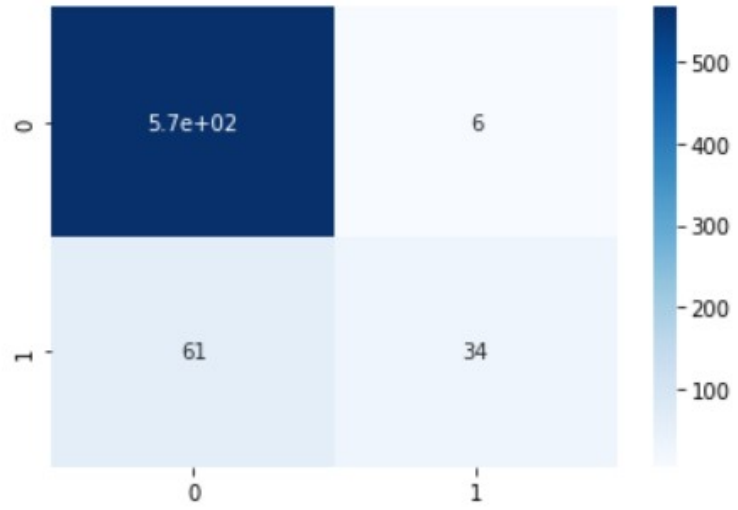
```
0.8995502248875562
```

图 4.3　多层感知机模型得分

对模型的可行性进行评估。

```python
cm=confusion_matrix(test_y, pred_MLP)
TP=cm[0,0]
FP=cm[0,1]
FN=cm[1,0]
TN=cm[1,1]
sns.heatmap(cm,annot=True,cmap='Blues')
plt.show()
Accuracy = (TP+TN)/(TP+FP+TN+FN)
Precision = TP/(TP+FP)
Recall = TP/(TP+FN)
print("准确率: ",Accuracy)
print("精准率: ",Precision)
print("召回率: ",Recall)
```

依次输出该模型的混淆矩阵、准确率、精准率、召回率。



准确率： 0.8995502248875562
精准率： 0.9895104895104895
召回率： 0.9027113237639554

第五部分 总结与收获

经评估可知各个模型的准确率均高于 80%, 达到实验预期。