《人工智能导论》机器学习实践性大作业

对手机价格分层的模型预测

姓　　名　李文涛

学　　号　320200928101

专　　业　电子信息基地班

年　　级　2020 级

对手机价格分层的模型预测

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
data_test = pd.read_csv('test.csv')
data_train = pd.read_csv('train.csv')
```

将项目处理所需的库和数据集导入。

1.2 浏览数据集

```
data_test.head()
```

用.head() 取前五个数据，以用来对数据所包含的信息进行理解。

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi
0	1	1043	1	1.8	1	14	0	5	0.1	193	...	16	226	1412	3476	12	7	2	0	1	0
1	2	841	1	0.5	1	4	1	61	0.8	191	...	12	746	857	3895	6	0	7	1	0	0
2	3	1807	1	2.8	0	1	0	27	0.9	186	...	4	1270	1366	2396	17	10	10	0	1	1
3	4	1546	0	0.5	1	18	1	25	0.5	96	...	20	295	1752	3893	10	0	7	1	1	0
4	5	1434	0	1.4	0	11	1	49	0.5	108	...	18	749	810	1773	15	8	7	1	0	1

5 rows × 21 columns

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 battery_power、blue、four_g 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。(由于每条数据的属性过多，这里列出部分重要属性)

battery_power：手机的电池容量

blue：手机是否具有蓝牙功能

four_g：手机是否具有 4G 功能

ram：手机的内存容量

three_g：手机是否具有 3G 功能

touch_screen：手机是否为触摸屏

wifi：手机是否具有 wifi 功能

目标任务为判断测试集手机的 price_range 属性。

第二部分 数据清洗

2.1 查看训练集详细信息

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

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	2
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	...	2000.000000	2000.000000	2000.000000	2000.000000	2
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	0.501750	140.249000	4.520500	...	645.108000	1251.515500	2124.213000	12.306500	
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	0.288416	35.399655	2.287837	...	443.780811	432.199447	1084.732044	4.213245	
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.000000	1.000000	...	0.000000	500.000000	256.000000	5.000000	
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	0.200000	109.000000	3.000000	...	282.750000	874.750000	1207.500000	9.000000	
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	0.500000	141.000000	4.000000	...	564.000000	1247.000000	2146.500000	12.000000	
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	0.800000	170.000000	7.000000	...	947.250000	1633.000000	3064.500000	16.000000	
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.000000	8.000000	...	1960.000000	1998.000000	3998.000000	19.000000	

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   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64
 4   fc             2000 non-null   int64
 5   four_g         2000 non-null   int64
 6   int_memory     2000 non-null   int64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64
 9   n_cores        2000 non-null   int64
 10  pc             2000 non-null   int64
 11  px_height      2000 non-null   int64
 12  px_width       2000 non-null   int64
 13  ram            2000 non-null   int64
 14  sc_h           2000 non-null   int64
 15  sc_w           2000 non-null   int64
 16  talk_time      2000 non-null   int64
 17  three_g        2000 non-null   int64
 18  touch_screen   2000 non-null   int64
 19  wifi           2000 non-null   int64
 20  price_range    2000 non-null   int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

图 2.2 训练集各数据属性分析

经观察下图我们发现，各属性数据均完整，且数据类型均为 float 和 int 类型，无需多余处理。

第三部分　数据分析

下面对处理后的数据通过相关直方图和相关系数热力图进行观察比较，选取我们所需要的对手机价格分层的判断属性。

由于数据属性过多，这里我们以 battery_power、ram 两属性联合对价格层级 price_range 的影响。

```
import seaborn as sns
import matplotlib.pyplot as plt
```

导入对相关性分析必要的包。

```
sns.lmplot( data = data_train , x = 'battery_power' ,y = 'ram' , hue = '
price_range');
```

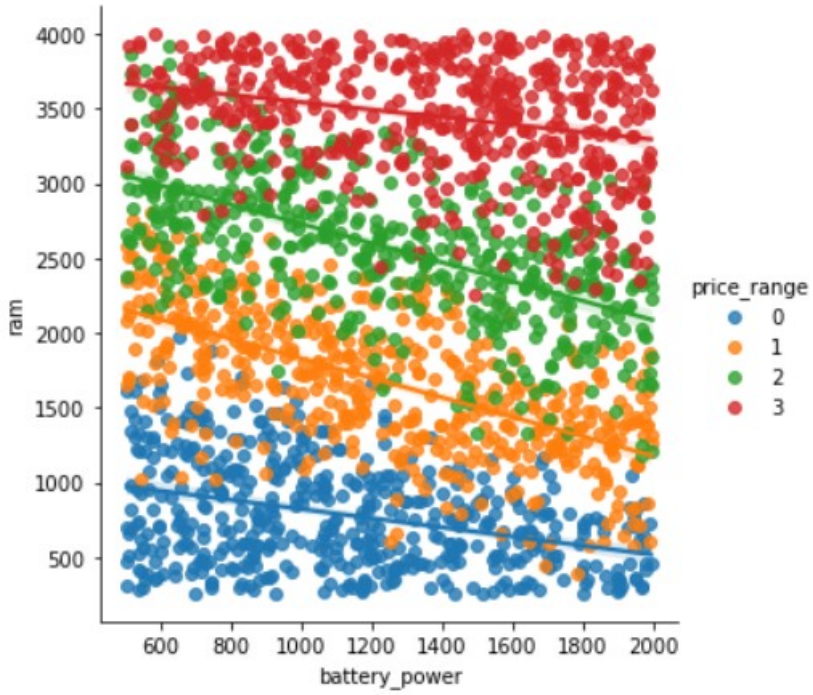

图 3.1　battery_power、ram、price_range 的相关关系程度

```
sns.pairplot(data_train)
```

利用 sns.pairplot 函数进一步对所有属性的相关性进行分析，由于导出的图过大，这里将其单独作为附件导出。

第四部分 构建模型

考虑到该数据集数据的特殊性，这里采用 KNN 来进行模型的构建。以下步骤按照参考资料中的代码进行模型的构建。

4.1 处理并划分数据集

```
# 划分数据集
data_train_y = data_train.pop('price_range')
data_train_x = data_train
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data_train_x, data_train_y
,random_state=42)
```

这里利用 random_state 对每次拆分出的训练集和测试集进行规定，以避免每次数据集的划分随机。

4.2 模型训练

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train, y_train)
```

第五部分 评估模型

下面我们分别从混淆矩阵和.score 函数来对我们所训练出的模型进行评估。这里我们将评估这一步骤封装为一个输入模型得出评估的函数，以便之后的进一步改进。

```
from sklearn.metrics import confusion_matrix,accuracy_score
def framework(model):
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test,y_pred)
    acc = accuracy_score(y_test,y_pred)
    print("ACC:",acc)
    sns.heatmap(cm,annot=True,cmap='Blues')
    plt.show()
```

```
framework(model)
```

评估模型。

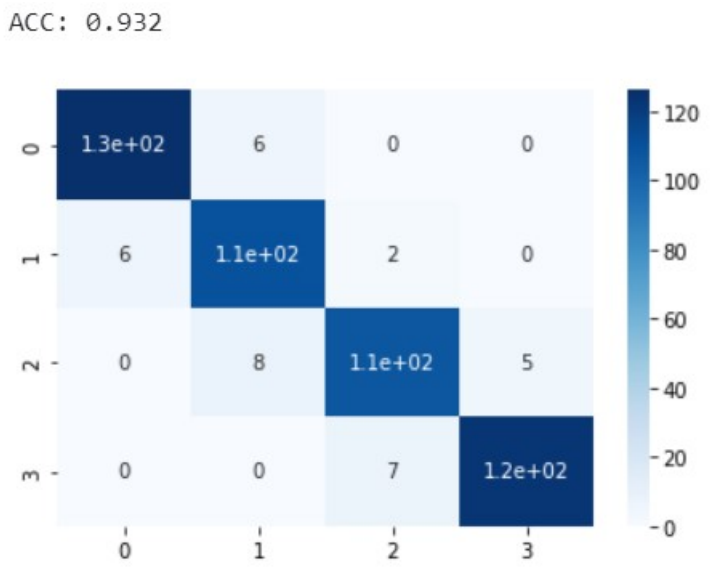

图 5.1 KNN 模型评估结果

第六部分 方案改进

经查阅资料，了解到交叉验证可以提高模型的准确度，以下对 KNN 模型进行交叉验证的改良。

```
from sklearn.model_selection import GridSearchCV
n_neighbors = [5,6,7,8,9,10,11,12,13,14,15,16]
parameters = dict(n_neighbors=n_neighbors)
```

规定划分参数。

```
model_improve = GridSearchCV(KNeighborsClassifier(),parameters,cv=10)
model_improve.fit(X_train,y_train)
model_improve.best_score_
model_improve.best_params_
```

调用交叉验证函数，得到交叉验证最佳划分参数和在最佳划分参数下的评估结果。

```
best_model = model_improve.best_estimator_
framework(best_model)
```

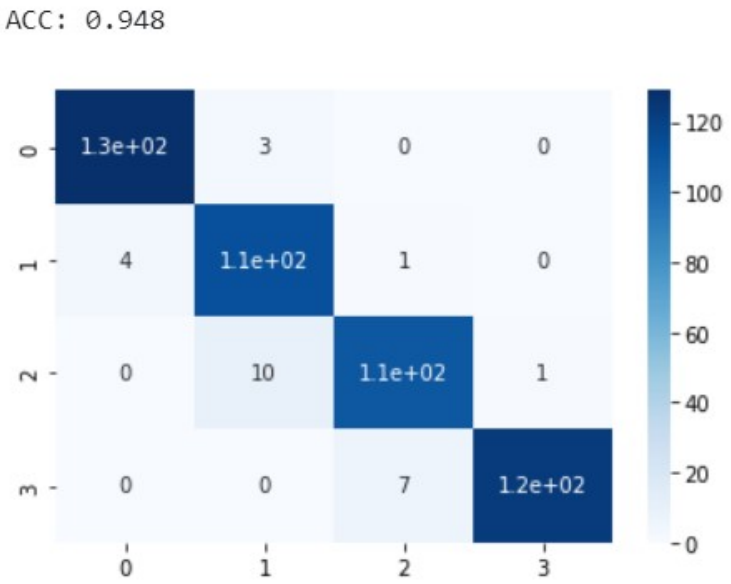

图 6.1 交叉验证后得到的最佳 KNN 模型评估结果

第七部分　方案实施及结果提交

```
# 提交数据
test =  data_test.drop('id',axis=1)
target = best_model.predict(test)
price_range = pd.DataFrame(target,columns=['price_range'])
result = pd.concat([data_test,price_range],axis=1)
#拼接得到的price_range与测试集
result.to_csv('result.csv')#输出结果到文件
```

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
0	1	1043	1	1.8	1	14	0	5	0.1	193	...	226	1412	3476	12	7	2	0	1	0	3
1	2	841	1	0.5	1	4	1	61	0.8	191	...	746	857	3895	6	0	7	1	0	0	3
2	3	1807	1	2.8	0	1	0	27	0.9	186	...	1270	1366	2396	17	10	10	0	1	1	2
3	4	1546	0	0.5	1	18	1	25	0.5	96	...	295	1752	3893	10	0	7	1	1	0	3
4	5	1434	0	1.4	0	11	1	49	0.5	108	...	749	810	1773	15	8	7	1	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	996	1700	1	1.9	0	0	1	54	0.5	170	...	644	913	2121	14	8	15	1	1	0	2
996	997	609	0	1.8	1	0	0	13	0.9	186	...	1152	1632	1933	8	1	19	0	1	1	1
997	998	1185	0	1.4	0	1	1	8	0.5	80	...	477	825	1223	5	0	14	1	0	0	0
998	999	1533	1	0.5	1	0	0	50	0.4	171	...	38	832	2509	15	11	6	0	1	0	2
999	1000	1270	1	0.5	0	4	1	35	0.1	140	...	457	608	2828	9	2	3	1	0	1	2

1000 rows × 22 columns

图 7.1　部分结果输出实例

第八部分 总结与收获

进一步了解了人工智能模型构建的步骤，学习了对模型进一步优化的方法。认识到封装模型构建函数的不足之处，了解到对于数据的性质，采取不同的模型预测是十分必要的。