《人工智能导论》机器学习实践性大作业

基于泰坦尼克号的乘客信息预测其生存率

姓　　名　李文涛

学　　号　320200928101

专　　业　电子信息基地班

年　　级　2020 级

基于泰坦尼克号的乘客信息预测其生存率

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

data_train=pd.read_csv("./train.csv")
data_test = pd.read_csv("./test.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)
    data_test.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。

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

图 1.1　训练集数据实例

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

图 1.2　测试集数据实例

1.3 对数据属性的理解

通过上一步我们发现该数据集有 PassengerId、Survived、Pclass 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。

PassengerId：乘客的编号

Survived：该名乘客是否生还（1 为生还，0 为遇难）

Pclass：该乘客所乘坐的座舱的等级

Name：该乘客的姓名

Sex：该乘客的性别

Age：该乘客的年龄

SibSp：该乘客在船上有多少堂兄弟妹

Parch：该乘客在船上有多少父母或者孩子

Ticket：该乘客的船票编号

Fare：该乘客的船票票价

Cabin：该乘客的船舱编号

Embarked：该乘客的登船港口

第二部分　数据清洗

2.1　查看训练集详细信息

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

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
       PassengerId      Pclass         Age       SibSp       Parch        Fare
count   418.000000  418.000000  332.000000  418.000000  418.000000  417.000000
mean   1100.500000    2.265550   30.272590    0.447368    0.392344   35.627188
std     120.810458    0.841838   14.181209    0.896760    0.981429   55.907576
min     892.000000    1.000000    0.170000    0.000000    0.000000    0.000000
25%     996.250000    1.000000   21.000000    0.000000    0.000000    7.895800
50%    1100.500000    3.000000   27.000000    0.000000    0.000000   14.454200
75%    1204.750000    3.000000   39.000000    1.000000    0.000000   31.500000
max    1309.000000    3.000000   76.000000    8.000000    9.000000  512.329200
```

图 2.1　训练集及测试集各数据属性分析

2.2　查看数据缺失情况

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None
```

图 2.2　训练集各数据属性分析

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Pclass       418 non-null    int64
 2   Name         418 non-null    object
 3   Sex          418 non-null    object
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object
 10  Embarked     418 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
None
```

图 2.3　测试集各数据属性分析

经观察上图我们发现，属性 Age、Cabin、Embarked、Fare 在两个数据集中有缺失的部分，下面我们对这些部分进行数据的填充。

2.3 数据填充

经查阅资料，我们采用以下的代码对数据集中 Age、Cabin、Embarked、Fare 四个属性有缺失的数据进行填充。

```
# 填充数据值
def fillna_data(df_train, df_test):
    # 对训练集和测试集中的"Age"数据进行平均值填充
    df_train['Age'] = df_train['Age'].fillna(df_train['Age'].mean())
    df_test['Age'] = df_test['Age'].fillna(df_test['Age'].mean())
    # 添加一个新的类别"Missing"来填充"Cabin"
    df_train['Cabin'] = df_train['Cabin'].fillna('Missing')
    df_test['Cabin'] = df_test['Cabin'].fillna('Missing')
    # 用出现频率最多的类别填充训练集中的"Embarked"属性
    df_train['Embarked'] = df_train['Embarked'].fillna(
    df_train['Embarked'].mode()[0])
    # 用出现频率最多的类别填充测试集中的"Fare"属性
    df_test['Fare'] = df_test['Fare'].fillna(
    df_test['Fare'].mode()[0])
    return df_train, df_test

# 得到填充后的数据集 df_train, df_test
df_train, df_test = fillna_data(data_train, data_test)

```

2.4 数据处理

经查阅资料，为便于处理，我们将数据各属性的数据进行 one-hot 编码、归一化处理等，以备下一步数据的分析。

```
# 对数据集中的字符串数据进行编码处理
def preprocess_data(train, test):
    # 使用one-hot编码将登船港口"Embarked"进行转换
    # 训练集
    Embarked = pd.get_dummies(train['Embarked'], prefix='Embarked')
    tmp_train = pd.concat([train, Embarked], axis=1)
    tmp_train.drop(columns=['Embarked'], inplace=True)
    # 测试集
    Embarked = pd.get_dummies(test['Embarked'], prefix='Embarked')
```

```
            tmp_test = pd.concat([test, Embarked], axis=1)
            tmp_test.drop(columns=['Embarked'], inplace=True)
            # 将年龄归一化
            tmp_train['Age'] = (tmp_train['Age'] - tmp_train['Age'].min()) / (tmp_train['Age'].max() - tmp_train['Age'].min())
            tmp_test['Age'] = (tmp_test['Age'] - tmp_test['Age'].min()) / (tmp_test['Age'].max() - tmp_test['Age'].min())
            # 将船票价格归一化
            if 'Fare' in tmp_train.columns:
                tmp_train['Fare'] = (tmp_train['Fare'] - tmp_train['Fare'].min()) / (tmp_train['Fare'].max() - tmp_train['Fare'].min())
            if 'Fare' in tmp_test.columns:
                tmp_test['Fare'] = (tmp_test['Fare'] - tmp_test['Fare'].min()) / (tmp_test['Fare'].max() - tmp_test['Fare'].min())
            # 将性别"Sex"这一特征从字符串映射至数值
            # 0代表female, 1代表male
            gender_class = {'female': 0, 'male': 1}
            tmp_train['Sex'] = tmp_train['Sex'].map(gender_class)
            tmp_test['Sex'] = tmp_test['Sex'].map(gender_class)

            return tmp_train, tmp_test

```

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乘客是否生还的判断属性。

3.1　乘客存活概率与座舱等级的关系

```
# 乘客存活概率与客舱的关系
import seaborn as sns
sns.barplot(x='Pclass', y='Survived', data=df_train)
plt.show()
```

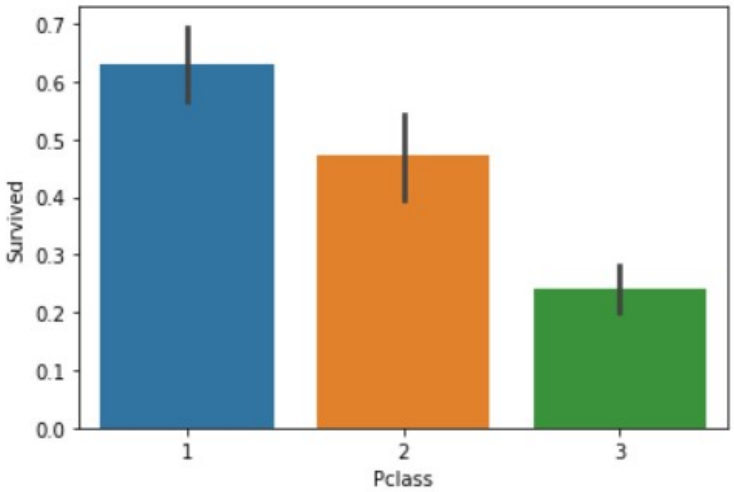

图 3.1　乘客存活概率与客舱之间的关系

3.2　乘客存活概率与性别的关系

```
sns.barplot(x='Sex', y='Survived', data=df_train)
plt.show()
```

3.3　各属性之间的相关关系程度

```
df = data_train[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', '
Survived']]
```

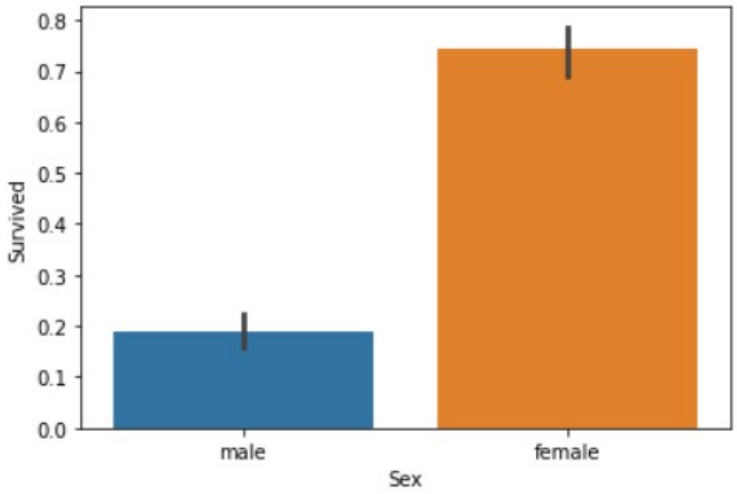

图 3.2 乘客存活概率与性别之间的关系

```
# 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()
```

```
              Pclass       Age     SibSp     Parch      Fare  Survived
Pclass      1.000000 -0.331339  0.083081  0.018443 -0.549500 -0.338481
Age        -0.331339  1.000000 -0.232625 -0.179191  0.091566 -0.069809
SibSp       0.083081 -0.232625  1.000000  0.414838  0.159651 -0.035322
Parch       0.018443 -0.179191  0.414838  1.000000  0.216225  0.081629
Fare       -0.549500  0.091566  0.159651  0.216225  1.000000  0.257307
Survived   -0.338481 -0.069809 -0.035322  0.081629  0.257307  1.000000
```

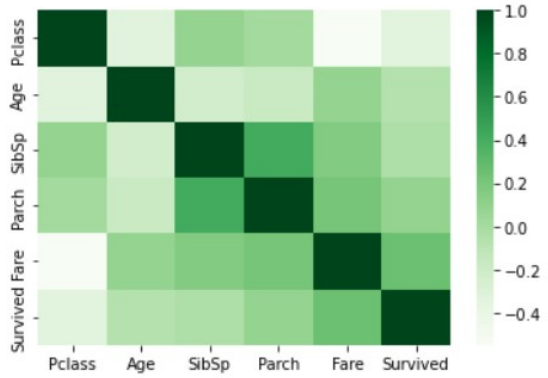

图 3.3 各属性之间的相关关系程度

通过常识和观察热力图我们发现 PassengerId, Name, Ticket, Cabin, SibSp, Parch 这几个属性与乘客是否获救的关系不大，我们认为其与乘客是否获救无关，在下一步的分析之前我们将其舍弃。

第四部分 构建模型

这里采用 SVM 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 去除无关数据

```
# 去除无关数据
df_train = df_train.drop(columns=['PassengerId', 'Name', 'Ticket', 'Cabin', 'SibSp', 'Parch'])
df_test = df_test.drop(columns=['PassengerId', 'Name', 'Ticket', 'Cabin', 'SibSp', 'Parch'])
```

利用.drop 函数把“无关”的数据属性去除。

4.2 处理并划分数据集

```
data_train, data_test = preprocess_data(df_train, df_test)
# 划分出标签类别，记录生存还是死亡，生存1，死亡0
label_train = data_train.loc[:, 'Survived']
data_train = data_train.drop(columns=['Survived'])
from sklearn.model_selection import train_test_split
train_X, test_X, train_y, test_y = train_test_split(data_train,
                                                    label_train,
                                                    train_size=.8)
```

这里取训练集的 80% 以备之后的模型测试及评估。

4.3 模型训练

```
from sklearn import svm
# 实例化对象，参数调整
clf_SVM = svm.SVC()
# 训练SVM模型
clf_SVM.fit(train_X, train_y)
```

第五部分　评估模型

下面我们分别从混淆矩阵、分类报告和.score 函数来对我们所训练出的模型进行评估。

```
from sklearn.metrics import confusion_matrix, classification_report

pred_SVM = clf_SVM.predict(test_X)
score = clf_SVM.score(test_X,test_y)
# 输出混淆矩阵
print(confusion_matrix(test_y, pred_SVM))
```

```
[[112   3]
 [ 21  43]]
```

图 5.1　混淆矩阵

```
# 输出分类报告
print(classification_report(test_y, pred_SVM))
```

	precision	recall	f1-score	support
0	0.84	0.97	0.90	115
1	0.93	0.67	0.78	64
accuracy			0.87	179
macro avg	0.89	0.82	0.84	179
weighted avg	0.88	0.87	0.86	179

图 5.2　分类报告

```
# 输出测试集评估结果
print(score)
```

```
0.8659217877094972
```

图 5.3　测试集评估结果

第六部分 方案实施及结果提交

```
data_predict = clf_SVM.predict(data_test)
data_predict_result = pd.DataFrame(data_predict,columns=['survived'])
#将需要预测的数据集放入我们构建的模型中并将结果存入data_predict_result
data2 = pd.read_csv('test.csv')
id = data2['PassengerId']
result = pd.concat([id,data_predict_result],axis=1)
result.to_csv('predict_result.csv')
#规范结果，并将结果以csv格式保存
```

第七部分　总结与收获

通过完整地对一个数据集进行分析，直到构筑模型并能够利用模型进行预测，个人了解了利用人工智能算法对一般问题解决的大致流程，能够通过查阅资料并对数据进行理解分析来完成信息的预测。

同时也发现了自己对人工智能算法方面的了解不足，在今后的学习过程中会更加主动地了解学习人工智能智能算法的实现。