7_titanic_machine_learning_from_disaster

吴清柳

July 24, 2023

1 泰坦尼克：从灾难中生存

使用机器学习方法预测从泰坦尼克灾难中生存下来的概率

1.1 第一步：定义问题

根据船上人员的性别, 年龄, 职业等信息, 设计一种算法来预测泰坦尼克号上的乘客的生存概率

1.2 第二步：获取数据

```
pip install --upgrade kaggle
```

如果本地没有数据, 则从 Kaggle 上进行下载. 需要设置账号和 API 到 ~/.kaggle/kaggle.json

```
kaggle competitions download -c titanic
```

1.3 第三步：数据预处理

对数据进行清理

[3]:

```
import sys
import pandas as pd
import matplotlib
import numpy as np
import scipy as sp
import IPython
import sklearn

import random
import time

# ignore warnings
import warnings
warnings.filterwarnings('ignore')
print('-'*25)

# Input data files are available in the `../dataset/` directory
from subprocess import check_output
print(check_output(['ls', '../dataset']).decode('utf8'))
```

```
-------------------------
gender_submission.csv
test.csv
titanic.zip
train.csv
```

1.4 加载数据建模库

使用 scikit-learn 库来开发机器学习算法. 在 sklearn 中, 算法被叫做 Estimators, 实现在他们各自的类里. 对于数据可视化, 使用 matplotlib 和 seaborn 库.

[8]:
```
# common model algorithms
from sklearn import svm, tree, linear_model, neighbors, naive_bayes, ensemble,␣
 ↪discriminant_analysis, gaussian_process
from xgboost import XGBClassifier

# common model helpers
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn import feature_selection, model_selection, metrics

# visualization
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.pylab as pylab
import seaborn as sns
from pandas.plotting import scatter_matrix

# configure visualization defaults
# show plots in jupyter notebook inplace
%matplotlib inline
mpl.style.use('ggplot')
sns.set_style('white')
pylab.rcParams['figure.figsize']=12,8
```

1.5 了解数据

了解数据的形状 (数据类型, 数据值) 等等

1. Survived 代表乘客是否存活
2. PassengerID 和 Ticket 被假设为随机独立标识符, 对输出没有影响, 因此会被从分析中移除
3. Pclass 代表票型, 并映射社会经济状态, 表示 1 = 上层阶级, 2 = 中层阶级, 3 = 下层阶级;
4. Name 是名字数据类型, 可能可以在特征工程中根据 title 判断性别, 从 surname 中家庭大小.
5. Sex 和 Embarked 变量是命名数据类型. 会被转为 dummy 变量来进行数学计算.
6. Age 和 Fare 变量是连续量化数据类型;
7. SibSp 表示同在船上的兄弟姐妹的数量, Parch 表示同在船上的父母孩子. 都是离散量化数据类型. 可以在特征工程中建立一个家庭大小, 是孤立的变量;
8. Cabin 变量是命名数据类型, 可以在特征工程中大致定位事故发生时在船上的位置, 以及根据等级判断接机. 然而, 犹豫有许多 Null 值, 该变量用处不大, 被排除在分析之外;

[11]:

```python
# import data
data_raw = pd.read_csv("../dataset/train.csv")

# break dataset into train, test and validation
# the test file provided is the validation file for competition submission
# split the train set into train and test data
data_val = pd.read_csv("../dataset/test.csv")

# to play with our data, create a copy
# python assignment or equal passes by reference vs vslues
data1 = data_raw.copy(deep=True)

# however passing by reference is convenient, because learn both datasets at
# once
data_cleaner = [data1, data_val]

# preview data
print(data_raw.info())
data_raw.sample(10)
```

```
<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
```

[11]:

```
     PassengerId  Survived  Pclass                             Name     Sex  \
838          839         1       3                 Chip, Mr. Chang    male
585          586         1       1             Taussig, Miss. Ruth  female
201          202         0       3             Sage, Mr. Frederick    male
824          825         0       3    Panula, Master. Urho Abraham    male
864          865         0       2           Gill, Mr. John William    male
491          492         0       3             Windelov, Mr. Einar    male
```

```
646          647        0      3              Cor, Mr. Liudevit    male
690          691        1      1           Dick, Mr. Albert Adrian    male
97            98        1      1  Greenfield, Mr. William Bertram    male
592          593        0      3         Elsbury, Mr. William James    male

      Age  SibSp  Parch            Ticket     Fare    Cabin Embarked
838  32.0      0      0              1601  56.4958      NaN        S
585  18.0      0      2            110413  79.6500      E68        S
201   NaN      8      2          CA. 2343  69.5500      NaN        S
824   2.0      4      1           3101295  39.6875      NaN        S
864  24.0      0      0            233866  13.0000      NaN        S
491  21.0      0      0  SOTON/OQ 3101317   7.2500      NaN        S
646  19.0      0      0            349231   7.8958      NaN        S
690  31.0      1      0             17474  57.0000      B20        S
97   23.0      0      1          PC 17759  63.3583  D10 D12        C
592  47.0      0      0          A/5 3902   7.2500      NaN        S
```

1.5.1 数据清理：数据纠正，数据补全，数据创建和数据转换

在该阶段中，1) 修正不正常数据和离群数据，2) 完成丢失的信息，3) 为分析创建新的特征，4) 转换数据到正确的格式以用于计算和表示

[12]:

```
print('Train columns with null values:\n', data1.isnull().sum())
print('-'*10)

print('Test/Validation columns with null values:\n', data_val.isnull().sum())
print('-'*10)

data_raw.describe(include='all')
```

```
Train columns with null values:
 PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
----------
Test/Validation columns with null values:
 PassengerId      0
Pclass           0
```

```
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
----------
```

[12]:

```
        PassengerId    Survived      Pclass                     Name   Sex  \
count    891.000000  891.000000  891.000000                      891   891
unique          NaN         NaN         NaN                      891     2
top             NaN         NaN         NaN  Braund, Mr. Owen Harris  male
freq            NaN         NaN         NaN                        1   577
mean     446.000000    0.383838    2.308642                      NaN   NaN
std      257.353842    0.486592    0.836071                      NaN   NaN
min        1.000000    0.000000    1.000000                      NaN   NaN
25%      223.500000    0.000000    2.000000                      NaN   NaN
50%      446.000000    0.000000    3.000000                      NaN   NaN
75%      668.500000    1.000000    3.000000                      NaN   NaN
max      891.000000    1.000000    3.000000                      NaN   NaN

               Age       SibSp       Parch  Ticket        Fare    Cabin  \
count   714.000000  891.000000  891.000000     891  891.000000      204
unique         NaN         NaN         NaN     681         NaN      147
top            NaN         NaN         NaN  347082         NaN  B96 B98
freq           NaN         NaN         NaN       7         NaN        4
mean     29.699118    0.523008    0.381594     NaN   32.204208      NaN
std      14.526497    1.102743    0.806057     NaN   49.693429      NaN
min       0.420000    0.000000    0.000000     NaN    0.000000      NaN
25%      20.125000    0.000000    0.000000     NaN    7.910400      NaN
50%      28.000000    0.000000    0.000000     NaN   14.454200      NaN
75%      38.000000    1.000000    0.000000     NaN   31.000000      NaN
max      80.000000    8.000000    6.000000     NaN  512.329200      NaN

       Embarked
count       889
unique        3
top           S
freq        644
mean        NaN
std         NaN
min         NaN
25%         NaN
```

```
50%          NaN
75%          NaN
max          NaN
```

[13]:

```python
# completing: complete or delete missing values in train and test/validation
# dataset
for dataset in data_cleaner:
    # complete missing age with median
    dataset['Age'].fillna(dataset['Age'].median(), inplace=True)

    # complete embarked with mode
    dataset['Embarked'].fillna(dataset['Embarked'].mode()[0], inplace=True)

    # complete missing fare with median
    dataset['Fare'].fillna(dataset['Fare'].median(), inplace=True)

# delete the cabin feature/column and others previously stated to exclude in
# train dataset
drop_column=['PassengerId','Cabin','Ticket']
data1.drop(drop_column,axis=1,inplace=True)

print(data1.isnull().sum())
print('-'*10)
print(data_val.isnull().sum())
```

```
Survived    0
Pclass      0
Name        0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
----------
PassengerId      0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          327
Embarked         0
dtype: int64
```

[15]:

```
# create: feature engineering for train and test/validation dataset
for dataset in data_cleaner:
    # discrete variables
    dataset["FamilySize"] = dataset["SibSp"] + dataset["Parch"] + 1

    dataset["IsAlone"] = 1  # initialize to yes/1 is alone
    dataset["IsAlone"].loc[
        dataset["FamilySize"] > 1
    ] = 0  # now update to no/0 if family size is greater than 1

    # quick and dirty code split title from name
    dataset["Title"] = (
        dataset["Name"]
        .str.split(", ", expand=True)[1]
        .str.split(".", expand=True)[0]
    )

    # continuous variable bins using qcut, cut into bins with approximately
    # equal amount of elements
    dataset["FareBin"] = pd.qcut(dataset["Fare"], 4)

    # age bins/buckets using cut with the same interval of each bin
    dataset["AgeBin"] = pd.cut(dataset["Age"].astype(int), 5)

# cleanup rare title names
print(data1["Title"].value_counts())
stat_min = (
    10  # while small is arbitrary, we'll use the common minimum in statistics
)
title_names = (
    data1["Title"].value_counts() < stat_min
)  # this will create a true false series with title name as index

# apply and lambda functions are quick and dirty code to find and replace with
# fewer lines of code
data1["Title"] = data1["Title"].apply(
    lambda x: "Misc" if title_names.loc[x] == True else x
)
print(data1['Title'].value_counts())
print('-'*10)

# preview data again
data1.info()
data_val.info()
data1.sample(10)
```

```
Title
Mr                517
```

```
Miss            182
Mrs             125
Master           40
Dr                7
Rev               6
Mlle              2
Major             2
Col               2
the Countess      1
Capt              1
Ms                1
Sir               1
Lady              1
Mme               1
Don               1
Jonkheer          1
Name: count, dtype: int64
Title
Mr        517
Miss      182
Mrs       125
Master     40
Misc       27
Name: count, dtype: int64
----------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Survived    891 non-null    int64
 1   Pclass      891 non-null    int64
 2   Name        891 non-null    object
 3   Sex         891 non-null    object
 4   Age         891 non-null    float64
 5   SibSp       891 non-null    int64
 6   Parch       891 non-null    int64
 7   Fare        891 non-null    float64
 8   Embarked    891 non-null    object
 9   FamilySize  891 non-null    int64
 10  IsAlone     891 non-null    int64
 11  Title       891 non-null    object
 12  FareBin     891 non-null    category
 13  AgeBin      891 non-null    category
dtypes: category(2), float64(2), int64(6), object(4)
memory usage: 85.9+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
```

```
Data columns (total 16 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          418 non-null    float64
 5   SibSp        418 non-null    int64
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         418 non-null    float64
 9   Cabin        91 non-null     object
 10  Embarked     418 non-null    object
 11  FamilySize   418 non-null    int64
 12  IsAlone      418 non-null    int64
 13  Title        418 non-null    object
 14  FareBin      418 non-null    category
 15  AgeBin       418 non-null    category
dtypes: category(2), float64(2), int64(6), object(6)
memory usage: 47.1+ KB
```

[15]:

```
     Survived  Pclass                                               Name  \
321         0       3                                   Danoff, Mr. Yoto
847         0       3                                 Markoff, Mr. Marin
543         1       2                                  Beane, Mr. Edward
731         0       3                           Hassan, Mr. Houssein G N
320         0       3                                 Dennis, Mr. Samuel
369         1       1                      Aubart, Mme. Leontine Pauline
270         0       1                              Cairns, Mr. Alexander
746         0       3                        Abbott, Mr. Rossmore Edward
599         1       1       Duff Gordon, Sir. Cosmo Edmund ("Mr Morgan")
820         1       1  Hays, Mrs. Charles Melville (Clara Jennings Gr…

        Sex   Age  SibSp  Parch     Fare Embarked  FamilySize  IsAlone Title  \
321    male  27.0      0      0   7.8958        S           1        1    Mr
847    male  35.0      0      0   7.8958        C           1        1    Mr
543    male  32.0      1      0  26.0000        S           2        0    Mr
731    male  11.0      0      0  18.7875        C           1        1    Mr
320    male  22.0      0      0   7.2500        S           1        1    Mr
369  female  24.0      0      0  69.3000        C           1        1  Misc
270    male  28.0      0      0  31.0000        S           1        1    Mr
746    male  16.0      1      1  20.2500        S           3        0    Mr
599    male  49.0      1      0  56.9292        C           2        0  Misc
820  female  52.0      1      1  93.5000        S           3        0   Mrs

             FareBin         AgeBin
```

```
321   (-0.001, 7.91]   (16.0, 32.0]
847   (-0.001, 7.91]   (32.0, 48.0]
543   (14.454, 31.0]   (16.0, 32.0]
731   (14.454, 31.0]  (-0.08, 16.0]
320   (-0.001, 7.91]   (16.0, 32.0]
369  (31.0, 512.329]   (16.0, 32.0]
270   (14.454, 31.0]   (16.0, 32.0]
746   (14.454, 31.0]  (-0.08, 16.0]
599  (31.0, 512.329]   (48.0, 64.0]
820  (31.0, 512.329]   (48.0, 64.0]
```

1.5.2 转换格式

将类别数据转换成dummy变量, 用于数学分析.

此外, 为数据建模定义x(independent/features/explanatory/predictor/etc.)和y(dependent/target/outcome/response/etc.)变量

[ ]:

```
# convert: convert objects to categoryusiing Label Encoder for train and
# test/validation dataset

# code categorical data
label = LabelEncoder()
for dataset in data_cleaner:
    dataset["Sex_Code"] = label.fit_transform(dataset["Sex"])
    dataset["Embarked_Code"] = label.fit_transform(dataset["Embarked"])
    dataset["Title_Code"] = label.fit_transform(dataset["Title"])
    dataset["FareBin_Code"] = label.fit_transform(dataset["FareBin"])
    dataset["AgeBin_Code"] = label.fit_transform(dataset["AgeBin"])

# define y variable aka target/outcome
Target = ["Survived"]

# define x variables for original features aka feature selection
data1_x = [
    "Sex",
    "Pclass",
    "Embarked",
    "Title",
    "SibSp",
    "Parch",
    "Age",
    "Fare",
    "FamilySize",
    "IsAlone",
]  # pretty name/values for charts
data1_x_calc = [
    "Sex_Code",
    "Pclass",
    "Embarked_Code",
    "Title_Code",
    "SibSp",
    "Parch",
    "Age",
    "Fare",
]  # code for algorithm calculation

# define x variables original w/bin features to remove continuous variables
data1_x_bin = [
    "Sex_Code",
    "Pclass",
    "Embarked_Code",
    "Title_Code",
    "FamilySize",
    "AgeBin_Code",
    "FareBin_Code",
]
data1_xy_bin = Target + data1_x_bin
print("Bin X Y: ", data1_xy_bin, "\n")

# define x and y variables for dummy features original
data1_dummy=pd.get_dummies(data1[data1_x])
data1_x_dummy
```