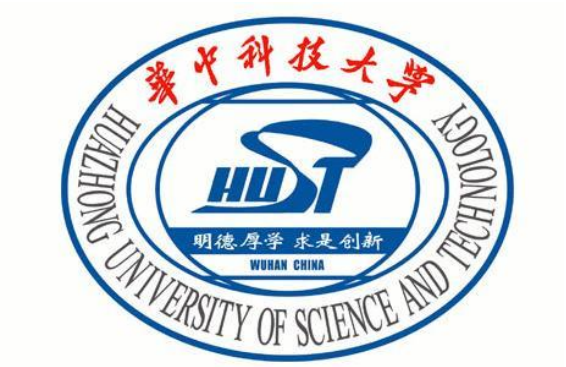
华中科技大学

计算机科学与技术学院

《机器学习》结课报告



专　　业：计算机科学与技术专业

班　　级：计卓 2101 班

学　　号：U202112071

姓　　名：王彬

成　　绩：

指导教师：何琨

完成日期：　2023 年 5 月 21 日

目录

1. 实验要求

我们选择选题一：逻辑回归应用之 Kaggle 泰坦尼克之灾（https://www.kaggle.com/competitions/titanic）。

1.1. 实验任务

问题以泰坦尼克号的沉没作为命题背景。1912 年 4 月 15 日，泰坦尼克号在处女航期间与冰山碰撞后沉没，它被广泛认为是“不可沉没”的。不幸的是，没有足够的救生艇供所有 2224 名乘客和机组人员使用，结果导致 1502 人死亡。

虽然存活涉及到一定的运气元素，但似乎有些人群比其他人更容易幸存。在这个挑战中，我们需要使用乘客数据（如姓名、年龄、性别、社会经济阶层等）构建一个可预测的模型，并回答哪些人更有可能生还。

我们需要根据公开数据集，准确并快速地提取乘客数据的特征，并以此预测测试集中的乘客人员的生存状况。我们将会把预测结果上传至 Kaggle 中，并评估模型的准确率。

1.2. 数据说明

本次竞赛的数据集（train.csv）提供了 12 种数据，分别对应乘客的个人信息和家庭状况。不同字段具备不同的数据格式，它们共同影响着该乘客的预期存活状况，其具体的字段说明见表 1。

表 1 字段数据说明

序号	变量名	数据格式	备注
1	PassengerId	Int64	乘客 ID 编号
2	Survived	Int64	存活情况
3	Pclass	Int64	乘客舱位
4	Name	object	乘客姓名
5	Sex	object	性别
6	Age	Float64	年龄
7	SibSp	Int64	堂兄弟、妹数量
8	Parch	Int64	家人数量
9	Ticket	object	船票
10	Fare	Float64	票价
11	Cabin	object	客舱数
12	Embarked	object	登船港口

而在测试集（test.csv）中，题目中也给出了除了存活状况之外的 11 种数据。我们可以通过给出的数据进行乘客的存活情况的预测。训练集的数据共 891 条，而测试集的数据共有 418 条。

2. 算法设计与实现

我们主要希望通过逻辑回归模型对原题进行求解。在本节中，我们将介绍对数据的认识、清洗与特征选择的过程，并介绍我们使用模型对泰坦尼克号其余乘客的存活情况的预测方式，再对我们的模型进行反馈与调整。

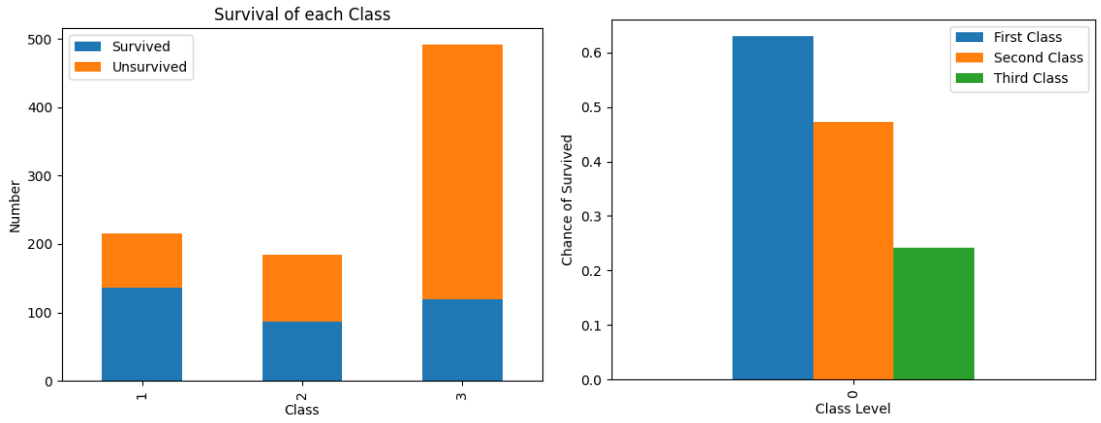
2.1. 特征提取

在进行正式的预测之前，我们首先需要提升对题目所给数据的认识。换言之，我们应当先确认各个已经给定的属性与最终乘客的存活情况之间的定性关系。

根据训练集的数据情况，我们有这些认识：不同舱位的乘客获救情况会有差异；不同性别的乘客的获救概率有显著区别等等。我们通过统计的方式以验证我们的猜想是否正确。

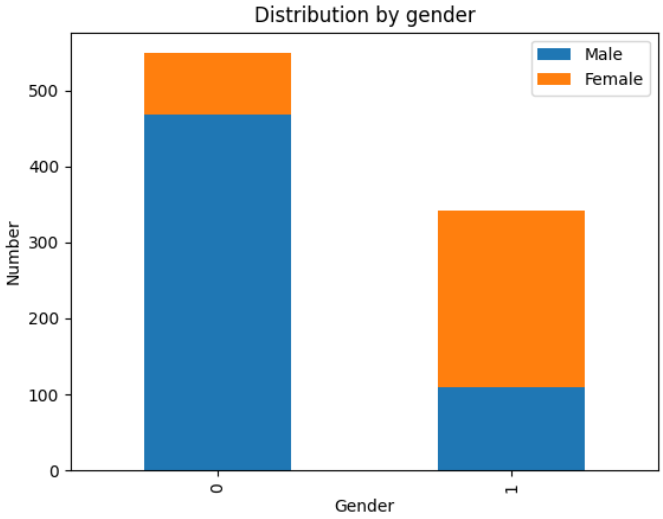
我们首先调用 matplotlib.pyplot 的绘图方法，统计每个舱位的人员的获救情况。如图 1 的获救情况所示，一等舱的乘客人员的获救概率最高，达到了一大半；二等舱的获救概率会比一等舱略小；而三等舱的获救概率则极小。这说明不同的舱位对于乘客的获救情况确实具有显著区别。

图 1 各舱位乘客的获救情况（左）各舱位乘客获救概率（右）



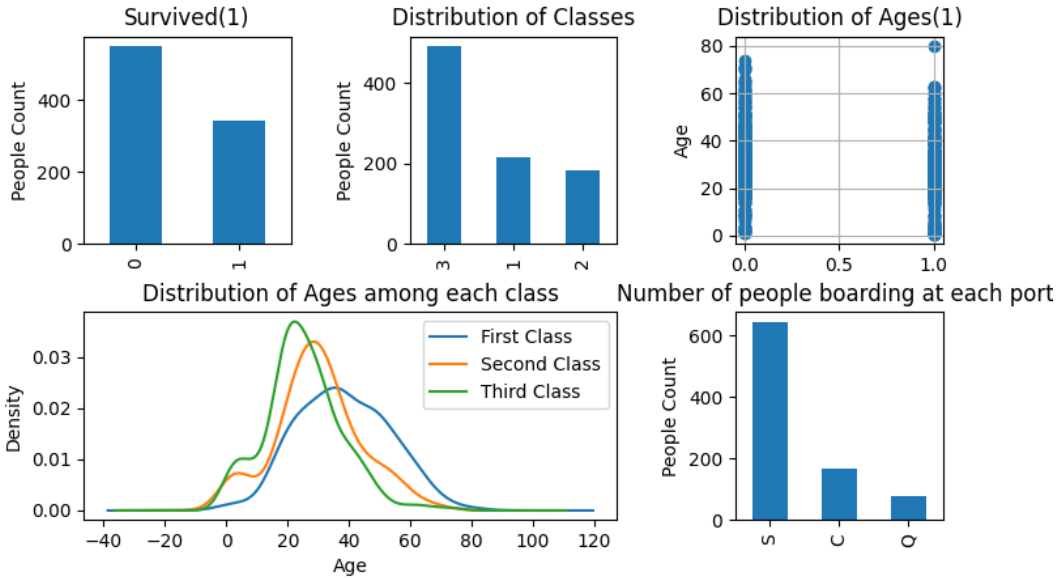
我们继续绘制乘客的获救数量与性别的关系。如图 2 乘客获救情况与性别，我们注意到未存活的人数中，男性的占比远高于女性；而在存活的名单中，女性的数量则远高于男性。根据上述分析可见，性别为女性的乘客可以有更高的几率存活。

图 2 乘客获救情况与性别



基于我们对于数据的基本认识，我们绘制一张数据统计的总表。如图 3 各属性对乘客存活状况影响的统计，我们可以看出不同的数据属性对于存活情况的影响关系，同时也可以清晰地看出训练集数据的数据分布。例如，一号舱（Class 1）内以中年人为主，而二号舱、三号舱的年龄众数则依次向年轻的方向移动；再如，登船入口以 S 口为主，这可能会影响到乘客在船舱内的居住位置，进而对他们的存活可能性有影响。

图 3 各属性对乘客存活状况影响的统计



综上，我们大致确定了数据集的数据倾向，以及对存活状况最有可能的几个影响因素。其中，年龄、性别、船舱类型被认为是各个属性中较为重要的三个属性，它们对乘客是否可以被成功获救具有较大的影响。

2.2. 特征工程

我们首先需要对数据进行预处理。在查看数据情况时，我们发现其中的 Age 属性和 Cabin 属性的缺失状况较为严重。如图 4 各数据缺失情况所示，在 891 条数据中，Age 属性存在缺失而有 714 条数据，而 Cabin 属性更是只有 204 条数据。

图 4 各数据缺失情况

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python311\pythor
<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
```

我们考虑对缺失的数据进行数据补全。首先，将数据集按照有 Age 和 Cabin 的部分和没有 Age 和 Cabin 的部分分别拆分成两个数据集。对于有 Age 和 Cabin 的数据集，我们可以使用聚类算法将数据分成若干类，对于每个类进行分别求取 Age 的中位数或众数，然后将这些值作为缺失值的填充；对于没有 Age 的数据集，我们则利用随机森林回归的方法进行填充。

对于 Age 属性的填充，事实上，前者方法更优。因为如果直接使用随机森林的方法进行年龄填充，各个数据和年龄之间的联系尚不明确，从而使得预测的年龄信息也是随机的。我们在优化前使用的是随机森林算法进行预测，而在优化后则使用直接取相同属性类的中位数进行填充。

而对于 Cabin 属性信息，我们注意到 Cabin 属性数量较多，其信息量较为庞杂但和预测量关联不大。同时，Cabin 属性的缺失量过大，有大半的信息都为缺失。综合这两点来看，很难用一种方式实现数据补全。但我们注意到，有 Cabin 信息的数据列的存活率较无 Cabin 信息的要高。因此，我们考虑将 Cabin 列简化

为有无 Cabin 信息。

在这轮的数据补全后，我们获得了已经补全的数据。如图 5 所示，每一数据组都已将各属性的值填充完毕。

图 5 补全后的训练数据

	PassengerId	Survived	Pclass	...	Fare	Cabin	Embarked
0	1	0	3	...	7.2500	No	S
1	2	1	1	...	71.2833	Yes	C
2	3	1	3	...	7.9250	No	S
3	4	1	1	...	53.1000	Yes	S
4	5	0	3	...	8.0500	No	S
..	...	...	...	...	...	...	...
886	887	0	2	...	13.0000	No	S
887	888	1	1	...	30.0000	Yes	S
888	889	0	3	...	23.4500	No	S
889	890	1	1	...	30.0000	Yes	C
890	891	0	3	...	7.7500	No	Q

[891 rows x 12 columns]

为了将原数据集更加适用与逻辑回归的模型，我们还需要对数据进行离散化和归一化的操作。这样可以使得数据在同一个标准内，减少个体特征太过明显的可能。我们调用 sklearn.preprocessing.StandardScaler().fit_transform()方法，将数据进行更改范围，并将其适应至[-1,1]区间内。

我们对 Age 属性和 Fare 属性数据进行归一化处理后，数据基本符合我们进行模型学习的规范。同时，由于舱位、性别的数据为字符串数据或数值数据，我们需要对于数据利用 pd.getdummies()方法进行离散化。例如，将舱位数据 Pclass 转换为 3 个属性，分别为 Pclass_1、Pclass_2 和 Pclass_3，以判定其是否为该舱位的乘客。这样在进行归一化和离散化后，我们的结果如图 6 所示，可见我们的数据基本符合预期。

图 6 归一化和离散化的训练数据

scaled train data is like...

	PassengerId	Survived	Age	...	Pclass_3	Age_scaled	Fare_scaled
0	1	0	22.000000	...	1	-0.561377	-0.502445
1	2	1	38.000000	...	0	0.613173	0.786845
2	3	1	26.000000	...	1	-0.267740	-0.488854
3	4	1	35.000000	...	0	0.392945	0.420730
4	5	0	35.000000	...	1	0.392945	-0.486337
..	...	...	...	...	...	...	...
886	887	0	27.000000	...	0	-0.194330	-0.386671
887	888	1	19.000000	...	0	-0.781606	-0.044381
888	889	0	16.185117	...	1	-0.988244	-0.176263
889	890	1	26.000000	...	0	-0.267740	-0.044381
890	891	0	32.000000	...	1	0.172717	-0.492378

[891 rows x 18 columns]

2.3. 逻辑回归建模

在进行模型建立之前，我们还需要对于数据进行进一步处理。我们需要从处理后的数据中取出所需要的属性列。同时，需要注意的是，我们同时忽略了一些数据列，如 PassengerId 列、Name 列。其中，在本轮初次建模中，我们没有用到 Name 列中所包含的信息，而在后续的优化过程中，我们会对这些忽略的数据列中的信息进行特征提取，并进一步地优化模型。

回到本次的初轮建模，目前我们已经将各属性列进行了预处理和符合特征工程的处理。下一步，我们将需要的特征字段用 numpy 格式转存出来。完成这些操作后，我们提取出特征属性值、幸存结果，再用 linear_model.LogisticRegression() 的逻辑回归模型进行训练。

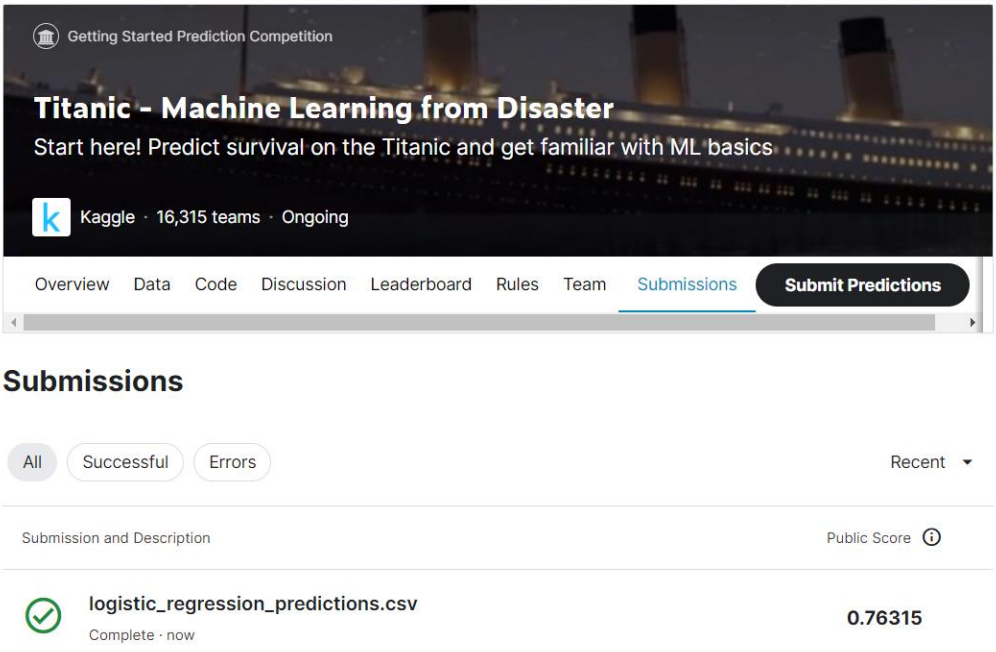
随后，我们使用相同的预处理及其它处理方式对测试集（test.csv）进行操作，使其格式和训练集一致。如图 7，我们将训练集数据操作完成后，即可对原题进行预测求解。

图 7 同等操作处理后的测试数据

```
test_data is like...
[[ 0.          0.          0.          ...  1.          0.30752581
  -0.49663711]
 [ 1.          0.          0.          ...  1.          1.25624212
  -0.5114971 ]
 [ 0.          0.          0.          ...  0.          2.39470168
  -0.46333473]
 ...
 [ 0.          0.          0.          ...  1.          0.61111503
  -0.50701688]
 [ 0.          0.          0.          ...  1.          0.0195507
  -0.49268018]
 [ 1.          1.          0.          ...  1.         -0.35612956
  -0.23626278]]
```

之后，我们使用分类器的预测函数对与测试数据进行预测，再将二分类结果保存至答案文件中。如图 8，我们将结果上传至 Kaggle，目前的正确率达到了 0.76315。

图 8 优化前逻辑回归训练结果



2.4. 特征优化

我们考虑对原代码进行进一步的优化。首先，我们需要先确认模型系数对于逻辑回归中的关联。因此我们先观察得到的模型系数与结果的相关度，相关数据如图 9 所示。

图 9 模型中各属性系数和幸存结果的相关度

	columns	coef
0	SibSp	[-0.34423360125417596]
1	Parch	[-0.10491778555378266]
2	Cabin_No	[0.0]
3	Cabin_Yes	[0.9020898484417407]
4	Embarked_C	[0.0]
5	Embarked_Q	[0.0]
6	Embarked_S	[-0.4172594665719434]
7	Sex_female	[1.9565666434028075]
8	Sex_male	[-0.6774204965304691]
9	Pclass_1	[0.34116856244445676]
10	Pclass_2	[0.0]
11	Pclass_3	[-1.1941309949200465]
12	Age_scaled	[-0.5237628097363776]
13	Fare_scaled	[0.08443567428865777]

可以看出，的确如同我们之前所预料的，性别对于成员是否获救具有很大的影响。例如，其中第 7 列中性别为女性的对于获救有较大的正相关关系，而其中第 8 列中性别为男性的乘客则更不容易获救。因此，我们希望加强性别、年龄和

船舱级别的特征；同时，由于实际数据中未见 Embark 属性对于结果的影响，我们希望削弱这些属性的贡献。

我们在第一轮训练中忽略了姓名字段的作用。在第二轮的训练里，我们需要重新审视姓名的作用。我们注意到，乘客的姓名前具有一个称谓（诸如“Mr”“Mrs”“Miss”），因此我们希望提取出这些姓名信息，对于其中乘客的信息进行进一步的处理。

我们通过这些方式以达到我们的预期：

- 增加一个 Child 字段，判断孩子的年龄是否较小，提高年龄较小成员的存活率；
- 增加 Mother 字段，如果乘客姓名的称谓里有“Mrs”的，以代指该乘客是已婚成员；
- 增加 Mixed_Status 字段，将性别和舱位进行复合，体现乘客的综合资产和性别信息；
- 去除登船地点属性（Embark 属性）；
- 改变 Age 属性的拟合填充方式，从随机森林算法改用对于称谓（如“Miss”）进行平均值填充。

在进行了这些优化后，仍然使用逻辑回归的模型，我们将该代码的预测结果上传至 Kaggle 中，如图 10 所示，这样的处理方法可以使准确率得分提高到 0.77511。

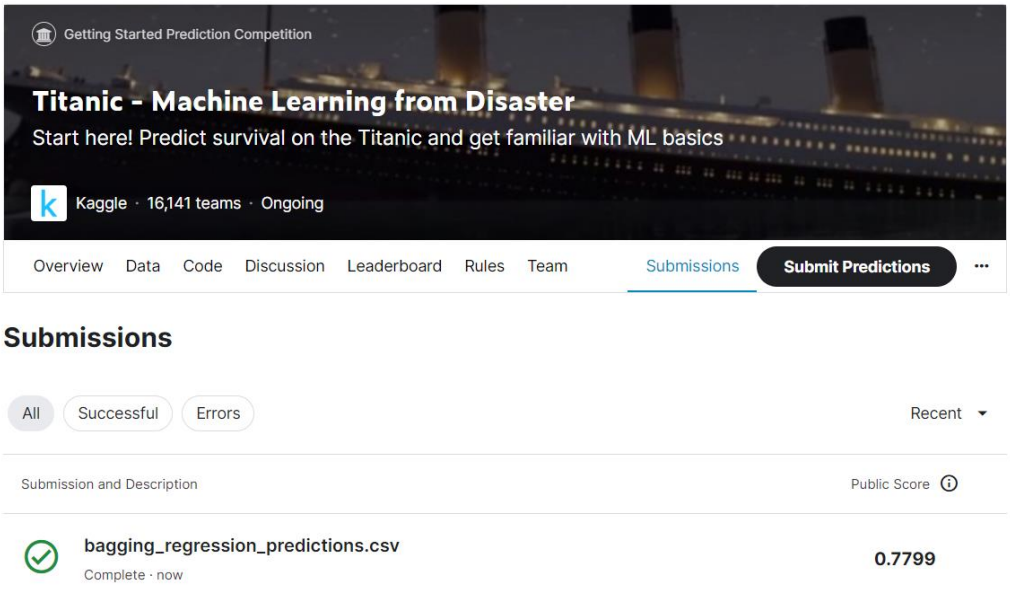
图 10 初次优化的逻辑回归模型得分

2.5. 集成学习优化

我们继续考虑对模型进行优化。我们上述使用的是逻辑回归的模型进行数据预测，而如果使用多种模型进行逐步的预测，最后将各个模型的预测结果进行投票，并票选出预测结果，准确率可能会有进一步的提高。

这是因为如果在一个模型上，训练模型出现了过拟合的情形，那么在所有的训练模型中不一定都会出现过拟合的情况。即使我们一直在试图提高 LogisticRegression 的正确率，但我们仍然可以使用 sklearn 的 Bagging 来完成集成学习。如图 11 所示，使用集成学习后，我们的正确率提高至了 0.7799。

图 11 初步模型融合后的 Kaggle 得分



但是我们并没有对于其它模型的训练做过更精细的训练，因此这种集成学习的集成程度并不高，仍有许多优化的空间。同时，由于 Kaggle 的提交次数有限，而我们的数据集数量较小，我们几乎未进行参数的调整，适当的调参也可以进一步提高准确率得分。

3. 实验环境与平台

（1）设备信息：

设备名称　CHINAMI-TV5O8C1

处理器　Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz　2.50 GHz

机带 RAM　8.00 GB (7.87 GB 可用)

设备 ID　1B3E14D6-1946-4087-B08A-780230EA6146

产品 ID　00326-10000-00000-AA970

系统类型　64 位操作系统, 基于 x64 的处理器

（2）系统信息：

版本　Windows 10 家庭版

版本号 22H2

操作系统内部版本　19045.2965

体验　Windows Feature Experience Pack 1000.19041.1000.0

（3）软件信息：

Python 版本　Python 3.11 (64-bit)

PyCharm 版本　PyCharm Community Edition 2023.1

其中，模型的训练和预测均基于本地计算机 CPU 进行。

4. 结果与分析

我们在优化前，所得到的 Kaggle 得分为 0.76315；在经过进一步的反馈特征分析后，继续使用逻辑回归模型，正确率上升至 0.77511；之后我们试图使用集成学习方法进行模型融合，得分有少量的提高至 0.7799。

实际上，我们也尝试使用了其它模型，比如随机森林模型。该模型的成绩居中，和逻辑回归模型类似。

该模型仍然可以继续优化。未来这个项目可以从这些角度进行进一步的优化，以提高 Kaggle 的准确率得分：

- 除了逻辑回归模型以外，还没有对于其它模型进行过更细致的训练；如果对于 KNN、SVM、RandomForest 模型进行综合，将会得到更好的结果；
- 由于提交次数有限，尚未对原代码进行调参，调整参数可能会进一步提高准确率；
- 特征信息还有待挖掘。例如姓名中的姓氏本身也有可能和获救情况有关，Embark 属性的登船港口可能会和获救情况有联系等等，这些都有待进一步的思考。
- 信息填充应当会有更好的方式。数据中的缺失信息，例如 Age 属性，应当可以用更可靠的方式进行估计。

综上，我们认为模型仍然有较大的上升空间。

5. 个人体会

这是我首次以个人完成机器学习的整个项目。本次项目是预测 Titanic 号邮轮的成员存活情况，它本身就有很大的意义。

在电影《泰坦尼克号》中，船长的一句“Lady first!”也真实地反映在了数据集中。我们发现，泰坦尼克号的幸存者中，女性成员的占比的确远高于男性。这样的数据，亦何尝不是一种人类文明与历史真实的体现。这样的数据，它并不只是数据；而饱含着彼时彼刻的情感和勇气。

进行数据处理的过程中，我所遇见的主要问题是数据中存在缺失。Age 属性如果直接放弃的话，将会使信息得到很大的损失。因此需要考虑，如何对这些缺失数据进行补全。在补全了相应数据，并进行了适当的特征分析后，成绩可以达到 0.76。

在进一步的分析和训练过程中，我对数据进行了进一步的处理，如加强特殊数据的重要度、直接去除不相关数据，或者直接进行集成学习。这样可以将准确率提高至 0.78，但总体上来说，上升地不显著。

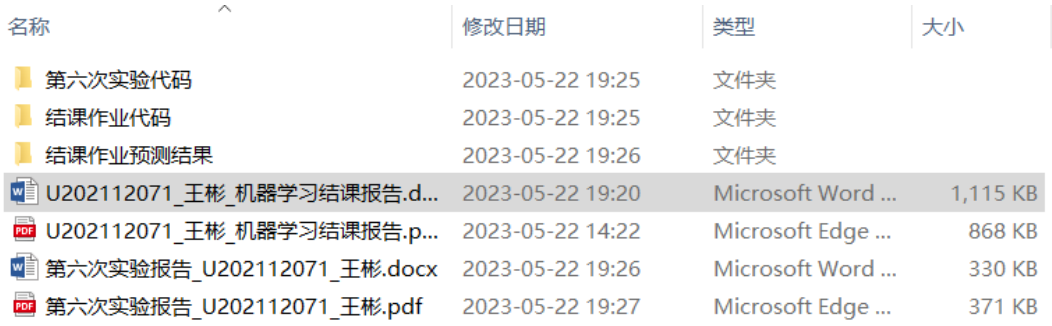
我认为可以通过更精细地训练其它模型精细集成，进而提高准确度。同时，我作为一个机器学习的入门者，对于特征工程的掌握仍然不算很熟练。“对于特征的认识决定了模型学习的上限”这句话可谓至言。

这是一次不错的机器学习入门项目，不断地提交但看不到正确率的上升的确很苦恼，但是最少还是有上升的情形还是有一定的成就感的。这一次的学习经历也告诉我：机器学习不仅仅有各种预测模型，其中同等重要甚至更加重要的是对于数据的认识、对于数据特征的分析。我想，数据科学中的数据处理，大概也需要一双敏锐的眼睛吧。

附录文件说明

附录文件如图 12 所示，分为结课报告（*.docx，*.pdf），第六次实验代码，结课作业预测结果及其代码。

图 12 结课提交文件夹



其中，“第六次实验代码”和“结课作业代码”文件夹下均存放有实验和大作业中所必须的程序代码，每个文件夹下都有 README 文件以对各个文件的用途进行解释。

“结课作业预测结果”文件夹下是两种模型对于数据集的预测，结果均已经上传至 Kaggle 中进行了测试，测试截图分别为实验报告中图 10 和图 11。