

**Harbin Institute of Technology, Weihai**

**大数据技术与实践**

**报告**

**学 号： 181110315**

**姓 名： 王少博**

|  |
| --- |
| 教师评语：  报告成绩： |

**目录**

[1. Task1：Pandas 、NumPy and Matplotlib 3](#_Toc56414444)

[1.1. 实验简介 3](#_Toc56414445)

[1.2. 问题描述 3](#_Toc56414446)

[1.3. 实验结果 3](#_Toc56414447)

[2. Task2：End-to-end Machine Learning project——房价预测 4](#_Toc56414448)

[2.1. 实验简介 4](#_Toc56414449)

[2.2. 问题描述 4](#_Toc56414450)

[2.3. 实验结果 5](#_Toc56414451)

[3. Task3：Using Decision Trees and Random Forest 6](#_Toc56414452)

[3.1. 实验简介 6](#_Toc56414453)

[3.2. 问题描述 6](#_Toc56414454)

[3.3. 实验结果 7](#_Toc56414455)

[4. Task4：ANN and CNN——人工神经网络和卷积神经网络 7](#_Toc56414456)

[4.1. ANN进行客户是否会离开银行的预测 7](#_Toc56414457)

[4.1.1. 实验简介 7](#_Toc56414458)

[4.1.2. 问题描述 7](#_Toc56414459)

[4.1.3. 实验结果 8](#_Toc56414460)

[4.2. CNN对猫狗的图片进行分辨 8](#_Toc56414461)

[4.2.1. 实验简介 8](#_Toc56414462)

[4.2.2. 问题描述 8](#_Toc56414463)

[4.2.3. 实验结果 9](#_Toc56414464)

[5. Task5：ANN for MNIST——使用ANN进行手写数字识别 10](#_Toc56414465)

[5.1. 实验简介 10](#_Toc56414466)

[5.2. 问题描述 10](#_Toc56414467)

[5.3. 实验结果 10](#_Toc56414468)

[6. 遇到的问题和解决方案 11](#_Toc56414469)

# Task1：Pandas, NumPy and Matplotlib

## 实验简介

这次实验主要是NumPy库、Pandas库和Matplotlib库的基础操作。都是一些基本的操作，这在之前已经很熟悉了。

## 问题描述

问题1主要是对Pandas.DataFrame的使用，我们需要先将数据进行初始化，然后在输出数据的基本信息即可。

问题2主要是对Pandas.DataFrame数据的获取，Pandas.DataFrame将数据以矩阵的形式存储在内存中，同时每行每列都有标签，通过行列标号来获取结果。

问题3主要是根据Pandas.DataFrame的标签来获取对应的行列数据

问题4和5主要是对NumPy库的简单使用，NumPy库是用于矩阵操作的一个库，这里主要是对矩阵的初始化，获取一些行列的数据值的操作。

问题6是对Matplotlib的简单使用，使用Matplotlib的plot进行简单的图像绘画。

## 实验结果

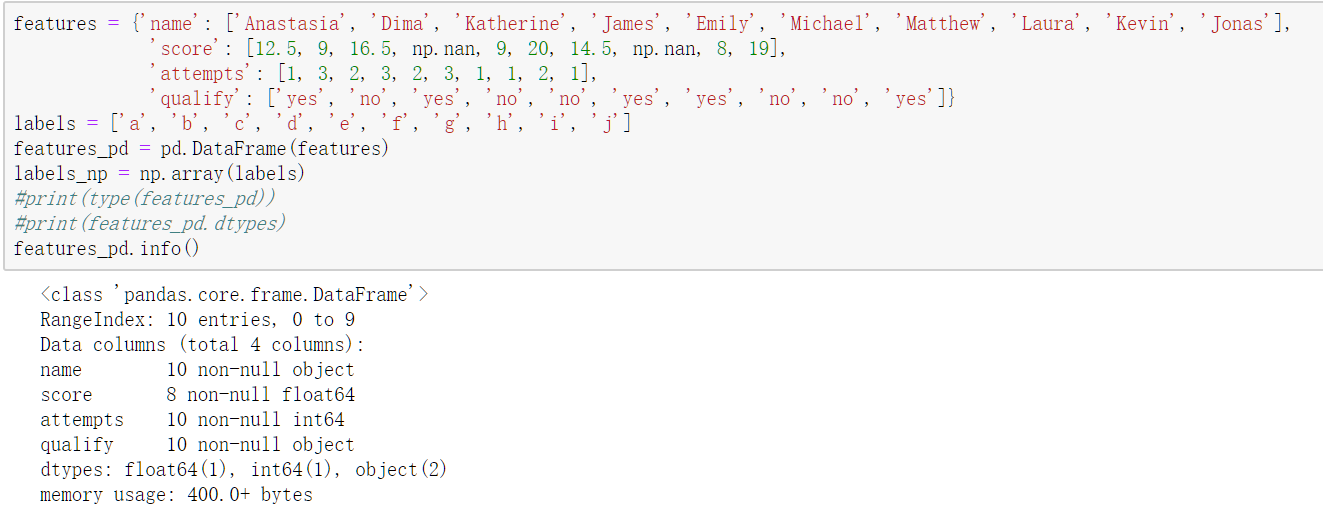


图1-1 问题1运行结果

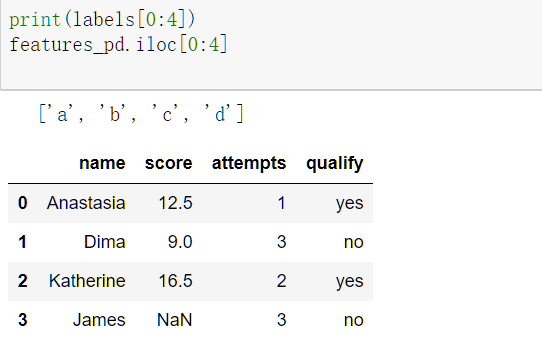


图1-2 问题2运行结果



图1-3 问题5运行结果

# Task2：End-to-end Machine Learning project——房价预测

## 实验简介

这次的实验主要包括了两个部分，第一个部分是学习部分，先通过教程学习基础的一些模块的使用，以及数据的读取，处理，和模型的使用的等一些知识。然后第二部分是通过学习到的知识，从数据读取处理到模型的预测的一整套流程，这里使用了SVM，随机森林，随机搜索，网格搜索等机器学习的算法进行数据的树立。

## 问题描述

第一部分学习，主要从获取数据开始，需要了解所给数据中各部分的含义，并查看数据各部分包含的内容。然后是数据的可视化，通过对数据的各种处理，可视化，可以让我们对数据的一些特性有一些别的了解，通过对数据的可视化，我们可以对房价的地区分布等特征有一个更加清楚的了解。之后是对数据处理，以让数据适合算法进行计算，例如这里的一些标签是非数字类型的，是字符串，那样就需要通过一些编码算法将这些字符串转换成二进制编码（或者说独热编码）来表示，然后准备好了数据之后，我们开始将这些数据放入模型进行预测，这里使用的有线性回归模型，决策树模型，以及随机森林和随机搜索。

第二部分是自己使用一些模型进行完整的放假预测，这里处理完数据之后，使用了SVM，网格搜索和随机搜索等，同时通过对数据的一些别的预处理例如以提高预测的正确率。

问题1：阅读pdf，复制黏贴代码，运行并观察结果。

问题2：使用超参数kernel设置为linear的 sklearn.svm.SVR训练模型，并通过均方根误差RMSE评估其性能。

问题3：使用RandomizedSearchCV训练模型。

## 实验结果

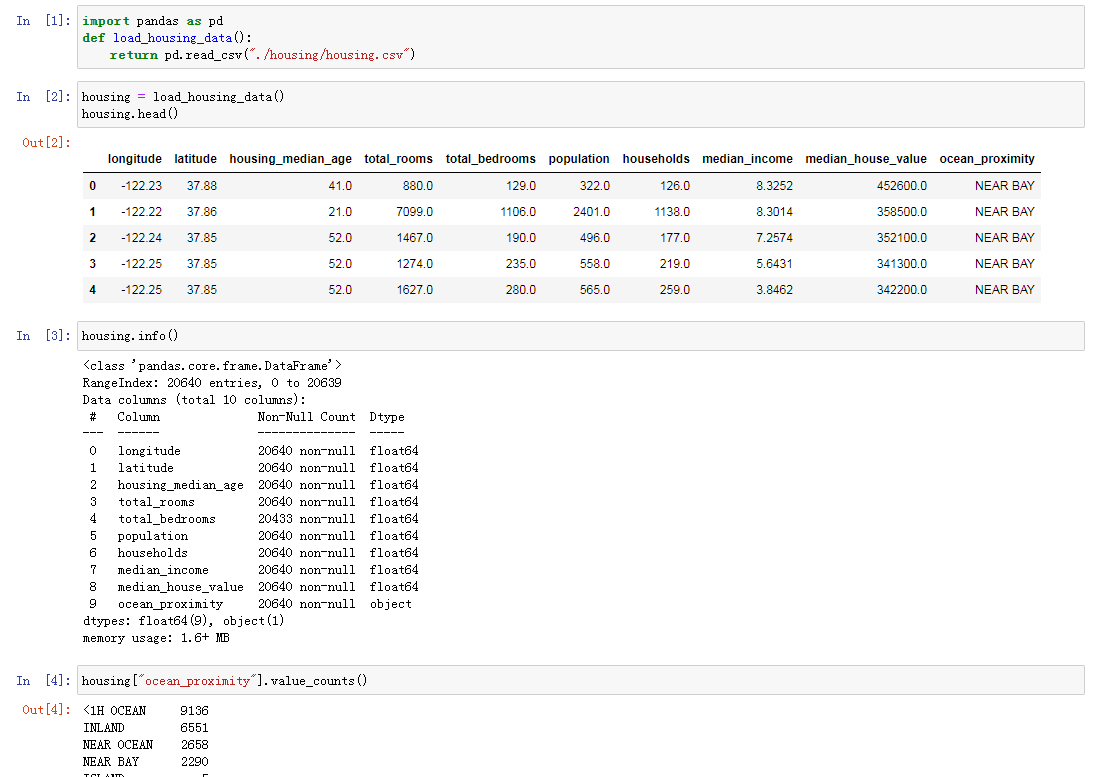


图2-1 问题1主体代码

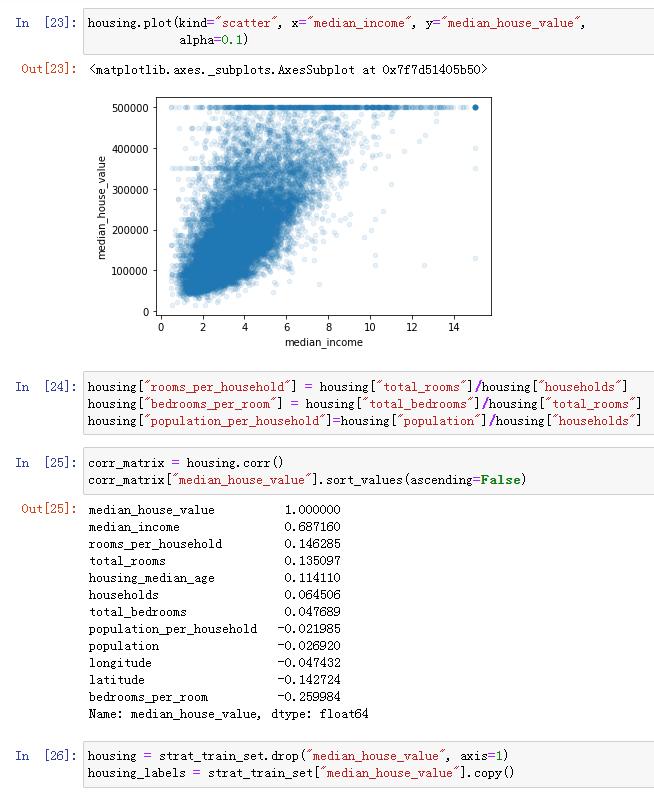


图2-2 问题1主体代码



图2-3 问题1主体代码（3）



图2-4 问题2主体代码（1）

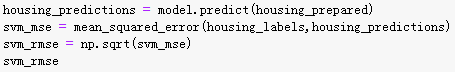


图2-5 问题2主体代码（2）

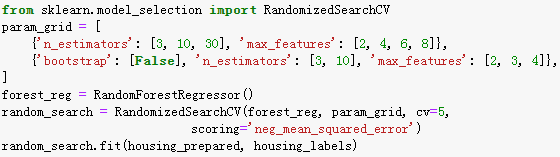


图2-6 问题3主体代码

**运行结果**

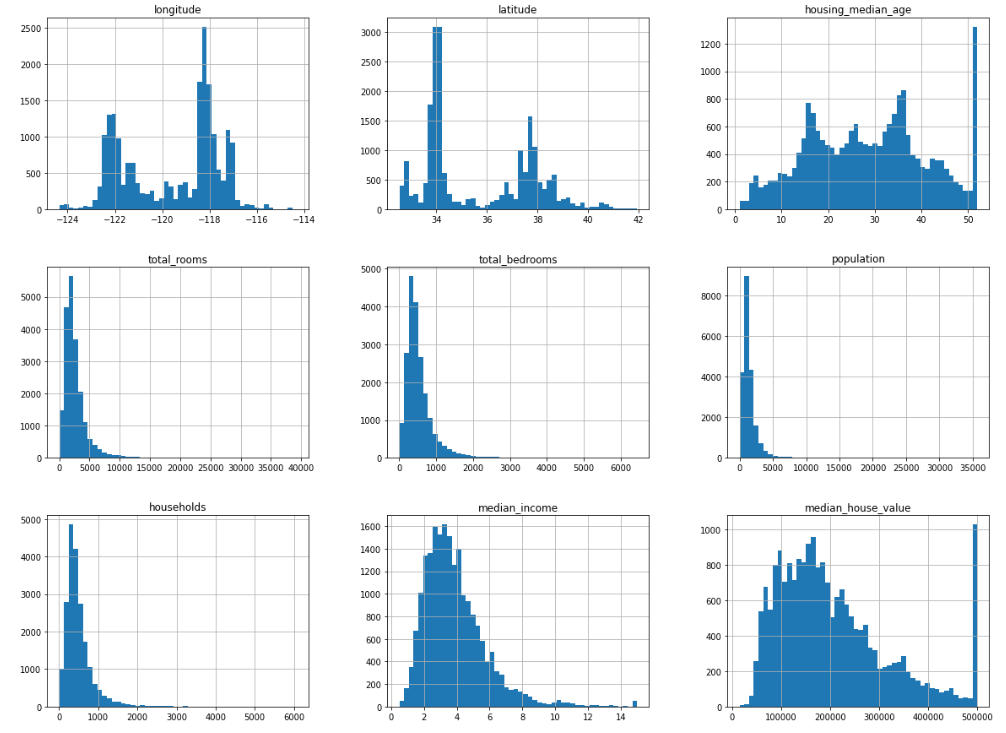


图2-7 问题1运行结果（1）

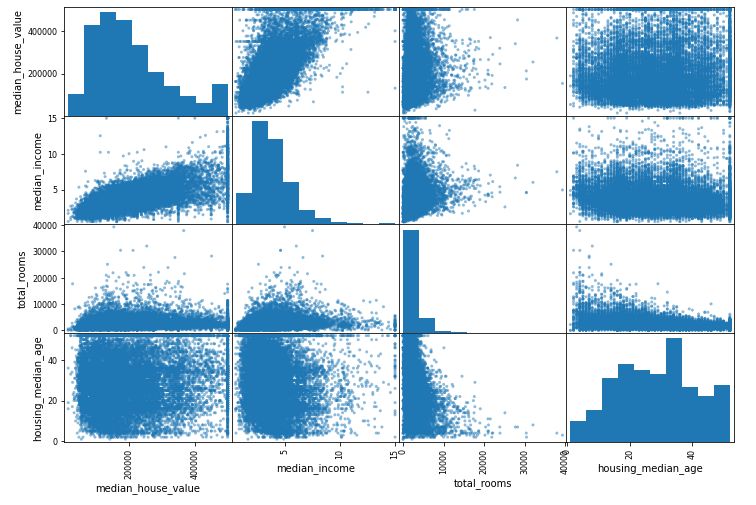


图2-8 问题1运行结果（2）

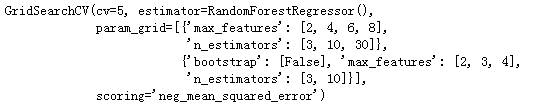


图2-9 问题1运行结果（3）

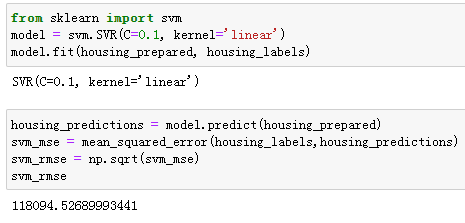


图2-10 问题2运行结果

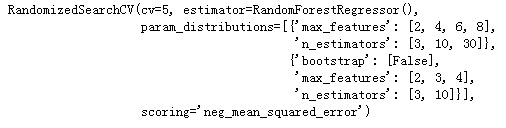


图2-11 问题3运行结果

# Task3：Using Decision Trees and Random Forest

## 实验简介

主要根据第二部分学习的知识，在这里使用随机森林和决策树对影响狗粮变质的因素进行查找。这里给出了一份含有四种添加剂A、B、C、D的狗粮的变质信息，然后通过这些数据进行计算是哪种添加剂加速了狗粮的变质。

## 问题描述

首先把数据读取到内存中，然后调用相关的模型方法进行预测，最后得到的是添加剂C是影响狗粮变质的主要因素。这里使用了决策树和随机森林进行预测，最后得到的是结果都表明是添加剂C，一般来说，决策树的准确率要高于随机森林，但是这里的数据比较少，需要预测的结果的类别也比较少，所以实际两者表现的准确率类似。

## 实验结果

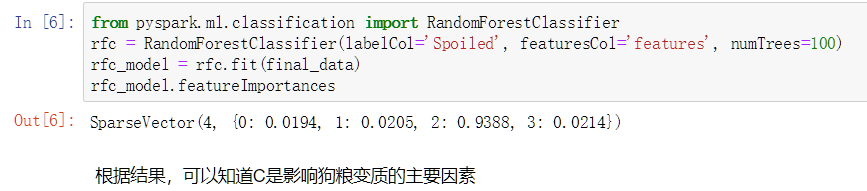


图3-1 随机森林的预测结果

# Task4：ANN and CNN——人工神经网络和卷积神经网络

## ANN进行客户是否会离开银行的预测

### 实验简介

因为客户的数据中一些信息是非数字表示的，所以先把这些数字转换成独热编码，然后通过pytorch构建一个三层（包括输入层）的人工神经网络来对训练数据进行训练，然后输入要预测的数据，预测结果。

### 问题描述

首先使用模块sklearn.preprocessing中的LabelEncoder,和OneHotEncoder对非数字的数据进行独热编码，然后对数据进行归一化，这样可以提高数据的紧凑性，防止数据的混乱度过大。

然后通过pytorch构建一个人工神经网络，实际上是两层线性回归构成的，然后为了防止过拟合，中间加入了随机失活层，同时使用ReLu作为激活函数，通过反向传播的方式进行模型参数的更新，最终得到训练好的模型。

最后通过将待预测数据进行归一化之后，然后输入到训练好的模型中，对结果进行预测，最终得到这个人会离开银行。

这里预测结果经过实际对比之后，感觉是错误的，但是模型却得到了这个结果，我认为有以下一些原因。

通过对数据中的其他一些数据的预测，发现大部分时候是是正确的，同时模型的训练准确率只有85%左右，这里的结果符合误差的范围。为什么训练的准确率这么低呢？首先这里的数据的也就是特征比较少，只有11个，虽然人工神经网络的大小已经降低了，但是不可避免的出现过拟合现象，导致泛化效果比较差。同时最主要的是因为数据中的特征太少，类比上图片之后可以发现，图片一般都是有几百个甚至上千的特征可以用来提取，这里相比，可以用来使用的特征太少。最后，考虑到训练的数据比较少，训练的数据里面不可避免的会有错误的现象，同时这里预测的是人会不会离开银行，对于人的行为本身就是很难确定的，即使初始条件相同，我们没法确保两个人的决定是相同的，所以这里预测出错和训练的正确率低是无法避免的现象。

### 实验结果

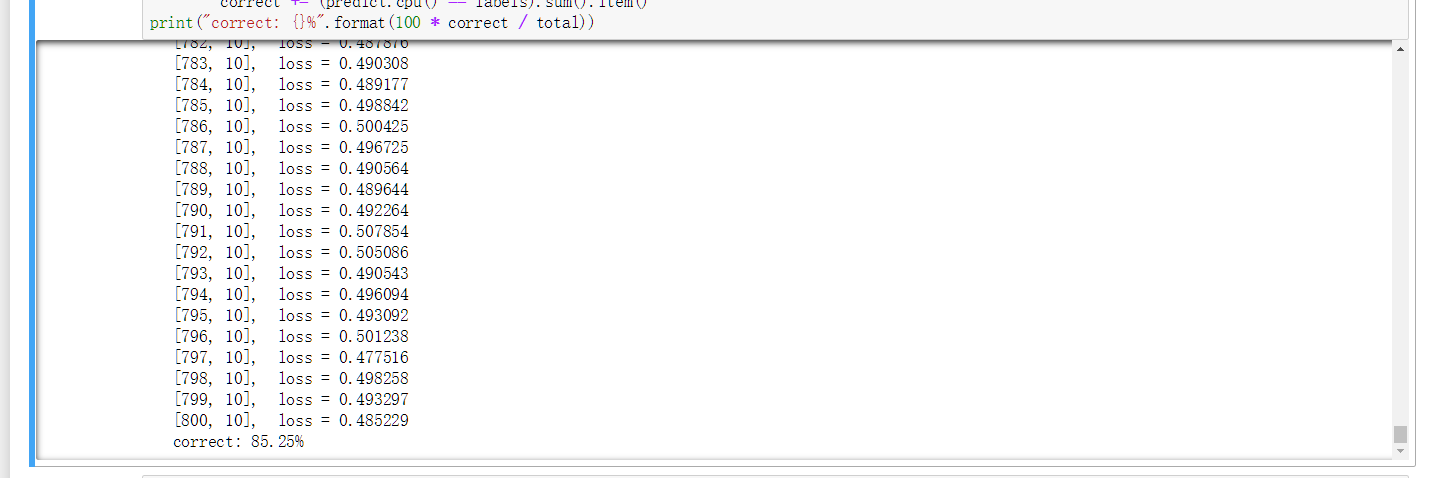


图4-1 部分训练结果截图

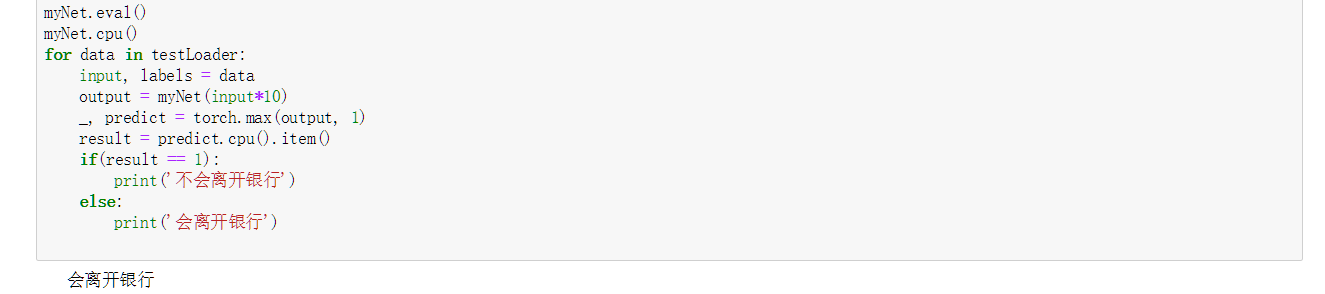


图4-2 预测结果截图

## CNN对猫狗的图片进行分辨

### 实验简介

这里主要是使用卷积神经网络对猫狗的图片进行分辨。主要的思路是先把数据读取进来，并对图片进行裁剪，因为这里的图片大小不一致，会无法输入网络中，然后把模型的训练结果进行保存，最后读取模型的训练结果，进行预测即可。

### 问题描述

首先读取数据，但是在读取数据的时候发现，实际的图片太多，有10000张图片，本身图片是碎片化的存储在硬盘上面，因为为了得到较好的模型结果，需要对模型进行多次训练，如果每次都从硬盘上面读取碎片化的数据，读取的时间比较长，极度影响训练效率。所以先把所有的图片读取到内存中，然后将训练图片和验证图片分别转换成对应格式的二进制文件存储到硬盘上，这样再次读取数据的时候，硬盘可以使用顺序读取的方式读取文件，然后加快了数据的读取速度。

其次是数据的预处理，首先是插值剪切，因为网络的输入数据大小是固定的，但是图片的大小是不固定的，我们只有先将图片裁剪到合适的大小才能输入网络。虽然都是猫和狗的图片，但是图片中的猫和狗的姿势都是不确定的，有的是站立，有的是躺着，有的是仰面朝天，为了保证模型的泛化性，我们需要做的是将图片读取过来之后，在送入模型训练之前，对图片进行随机翻转，然后再输入网络中。

之后是模型的建立，虽然自己搭建模型进行从头开始的训练也能得到很好的效果，因为这里就只有两种类别，而且训练的输入图片很多，但是为了节省时间，这里使用了预训练模型RestNet18，RestNet18严格的来说是残差神经网络，是卷积神经网络的改进版本，可以有效降低卷积神经网络的退化现象，加快训练速度。因为RestNet18的输出分类数量不是2类，所以我们需要将最终的线性输出层进行修改，修改成2种结果输出。然后进行训练，这里使用的是远程服务器上的GPU（RTX2080ti）进行训练。通过多次调试，发现训练10个epoch，得到的结果最好，大约是97.5%（这里因为是在服务器上训练，所以截图没有了）。

最后就是图片的预测，通过从服务器上下载训练好的模型，然后在本地读取模型后对所给的两种图片进行预测，可以得到正确的预测结果。

### 实验结果



图4-3 预测结果

# Task5：ANN for MNIST——使用ANN进行手写数字识别

## 实验简介

这里主要是先下载训练数据，然后读取数据，之后构建相应的人工神经网络进行训练，最后进行预测即可。

## 问题描述

这里使用pytorch官方的类进行MNIST数据的下载和读取，然后构建神经网络进行训练，在构建神经网络的时候，我使用了一个隐藏层的神经网络和两个隐藏层的神经网络进行分别训练，然后对比结果，最终我决定采用一个隐藏层的的神经网络，因为经过我的测试，一个隐藏层的神经网络在调整参数等一些方面有优势，同时也可以加快训练速度，减少过拟合。

在神经网络的构建上面，我采用了800个隐藏神经元的构造，使用70%的随机失活层。经过训练，在测试集上面，准确率达到了97.5%。

## 实验结果

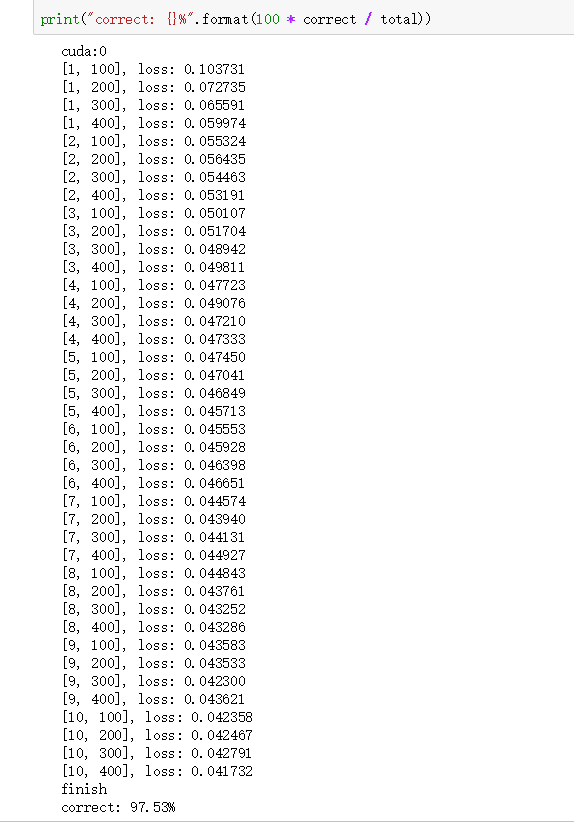


图5-1 训练过程

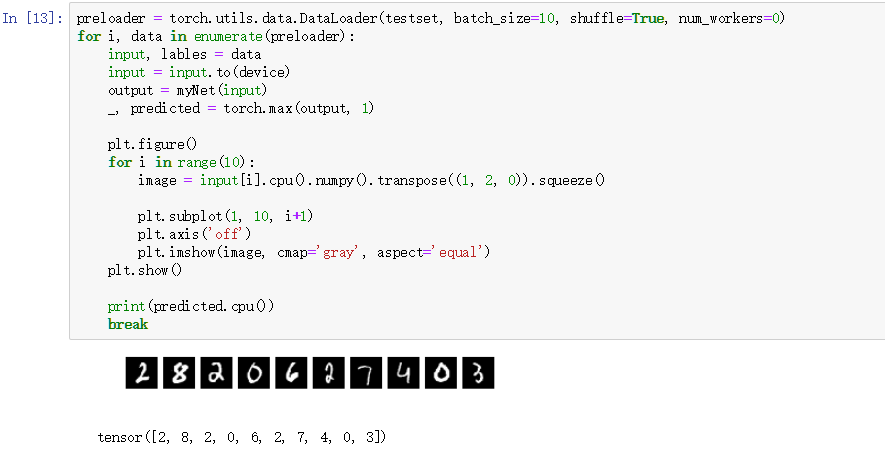


图5-2 预测结果

# 遇到的问题和解决方案

Task1中基本没有遇到什么问题，只是对Pandas稍微有点不熟悉，不知道怎么输出和要求对应的格式的数据，最后在百度的时候找到了。

Task2中根据教程中给的代码进行运行，但是运行到最后的时候，有一个自定义的类MyLabelBinarizer报错，后来百度之后发现是因为安装的库的版本问题，这里修改了一些方法的输入参数，可以正常运行了。在进行下面的训练模型的时候，由于以前没有使用过sklearn模块中的一些方法，所以写起来比较费力，最后从GitHub上面找到了一份教程代码，然后根据教程代码，我模仿着写的最后的模型预测，基本没有遇到太多的问题。

Task3中，根据百度到的pyspark.ml.classification中的RandomForestClassifier等一些模型的调用的使用方法，我将数据读取到内存中，然后进行预测，基本没有什么问题。

Task4中，对于问题1中的是否会离开银行的预测，一开始正确率一直很低，后来我发现是因为数据的特征太少，然后模型太大，导致训练收敛结果一直无法收敛，然后验证的正确率一直很低。后来修改了模型的大小之后，发现正确率达到了99%，但是在预测给的数据的时候，发现预测错了。经过排查，发现在模型训练和验证的时候，使用了同一个训练数据集，导致了模型实际上是严重过拟合了。修改了之后，发现正确率一直在85%左右，无法提升，最后分析后认为是数据数量太少，数据的特征太少，导致正确率一直很低。所以最后的结果属于正常现象。

问题2中，对猫狗进行预测，一开始遇到的问题就是数据读取问题，每次读取读取都需要花费五六分钟，所以考虑到把数据改成二进制文件存储成一个大文件。但是使用电脑处理的时候，16GB的内存无法满足需求，最后使用了服务器进行的处理，同时进行训练的时候，发现笔记本电脑的GTX960m显卡运行速度太慢，最后使用的是服务器RTX2080ti的显卡进行学习，然后通过将模型下载到笔记本，进行的数据预测。

Task5中遇到的问题是使用神经网络进行训练的时候，正确率一直在20%左右徘徊，最后通过增加隐藏的神经元个数，同时加入激活函数ReLu，把正确率提升到了80%以上，但是根据数据集作者给出的一个不同模型对应的预测正确率的表，我的模型还没有达到表中对应的模型的正确率。最后我加入了随机失活层，把正确率提升到了97.5%。同时我也测试了双隐藏层的神经网络，最后发现实际效果不如单层的，同时参数调整教困难，所以放弃了双隐藏层的神经网络。