**试题名称：1.1 业务和数据分析采集**

1. **请根据此需求分析需要采集什么样的数据，分析维度需包含采集种类、采集规则和采集要求。内容提交至“提交资料/题目 1/业务分析.docx”文件中。**

使用场景：RPA模拟人进入网页获取验证码图片->数字验证码识别模型返回图片预测结果->RPA填写后提交表单。

需求分析：基于对网页数字验证码的分析，需要文本验证码（其中数字的主要）

1. 采集种类：

* 4位或6位数字验证码
* 涉及简单数字计算的数字验证码
* 数字和字母混合验证码：同时包含数字和字母的验证码。

1. 采集规则

* 多样性：存在不同背景、不同字体、不同颜色、排序无规律的验证码，从而增加模型的泛化能力
* 噪声处理：基于验证码的安全机制主要针对对抗分割和对抗识别，对抗分割特点的安全机制增加了单个字符识别的难度，比如用字符粘连、加入空心线描绘字符轮廓、字符变长、噪点或噪声线等增加；对抗识别旨在增加识别字符串的难度，例如字符设置不同大小、旋转或扭曲等手段。需要基于这两点尽可能获取多种图片验证码，以尽可能全面覆盖数据预处理。
* 标签：对采集后的数字验证码图片进行标注，标注结果经过验证，以便用于监督学习

1. 采集要求

* 数量：至少采集10,000张验证码图像，以确保数据量足够训练一个高精度的模型。
* 质量：采集的图像需要清晰，无模糊、失真等问题。
* 格式：图像格式统一为PNG或JPG，尺寸统一为固定大小（例如100x40像素）。

**（2）至少写出两种采集方法，并描述如何通过该采集方法采集需要的数据。内容提交至“提交资料/题目 1/业务分析.docx”文件中。**

**方法一：网页抓取**

使用爬虫技术从目标网页上抓取图片并进行相应的数据标注

1. **步骤**：
   * **目标网站选择**：选择具有公开验证码功能的网站，如注册页面、登录页面等。
   * **编写爬虫程序**：使用Python编写爬虫程序，自动访问目标网站并抓取验证码图像。
   * **保存图像和标签**：将抓取到的验证码图像保存到本地，并记录对应的验证码文本。
2. **工具**：
   * **Scrapy**：用于网页数据的抓取。
   * **Selenium**：用于处理动态加载的网页元素，特别是生成验证码的场景。
3. **示例代码**：
4. import requests
5. from bs4 import BeautifulSoup
6. from PIL import Image
7. from io import BytesIO
8. def download\_captcha(url, save\_path):
9. response = requests.get(url)
10. img = Image.open(BytesIO(response.content))
11. img.save(save\_path)
12. url = "http://example.com/captcha"
13. save\_path = "captchas/captcha\_1.png"
14. download\_captcha(url, save\_path)

参考文献：

代码参考：<https://developer.aliyun.com/article/666608>

import requests

import os

import time

from lxml import etree

def get\_Page(url,headers):

response = requests.get(url,headers=headers)

if response.status\_code == 200:

# print(response.text)

return response.text

return None

def parse\_Page(html,headers):

html\_lxml = etree.HTML(html)

datas = html\_lxml.xpath('.//div[@class="captcha\_images\_left"]|.//div[@class="captcha\_images\_right"]')

item= {}

# 创建保存验证码文件夹

file = f"./test\_img"#'D:/\*\*\*\*\*\*'

if os.path.exists(file):

os.chdir(file)

else:

os.mkdir(file)

os.chdir(file)

for data in datas:

# 验证码名称

name = data.xpath('.//h3')

# print(len(name))

# 验证码链接

src = data.xpath('.//div/img/@src')

# print(len(src))

count = 0

for i in range(len(name)):

# 验证码图片文件名

filename = name[i].text + '.jpg'

img\_url = 'https://captcha.com/' + src[i]

response = requests.get(img\_url,headers=headers)

if response.status\_code == 200:

image = response.content

with open(filename,'wb') as f:

f.write(image)

count += 1

print('保存第{}张验证码成功'.format(count))

time.sleep(1)

def main():

url = 'https://captcha.com/captcha-examples.html?cst=corg'

headers = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.146 Safari/537.36'}

html = get\_Page(url,headers)

parse\_Page(html,headers)

if \_\_name\_\_ == '\_\_main\_\_':

main()

验证码生成：<https://captcha.com/captcha-examples.html?cst=corg>

**方法二：手动生成**

观察网页验证码特点并利用已有的验证码生成库，手动生成多种样式的验证码图像。

1. **步骤**：
   * **选择生成库**：选择一个功能丰富的验证码生成库，如captcha库。
   * **编写生成脚本**：编写Python脚本，批量生成不同样式的验证码图像。
   * **保存图像和标签**：将生成的验证码图像保存到本地，并记录对应的验证码文本。
2. **工具**：
   * **captcha库**：用于生成验证码图像。
3. **示例代码**：

python

import os

def check\_labels(image\_dir, label\_file):

with open(label\_file, 'r') as f:

labels = f.readlines()

for label in labels:

image\_name, text = label.strip().split(',')

image\_path = os.path.join(image\_dir, image\_name)

if not os.path.exists(image\_path):

print(f"Image {image\_name} not found!")

image\_dir = "processed\_captchas"

label\_file = "labels.txt"

check\_labels(image\_dir, label\_file)

参考文献：

1. <https://segmentfault.com/a/1190000022812969>
2. <https://mp.weixin.qq.com/s?__biz=MzU3MzQxMjE2NA==&mid=2247497242&idx=1&sn=ec08afcfd5e5533de3158d4e2c96f36b&chksm=fcc0b031cbb73927623bd7e4846ddaaa95722aff57c47f9ed560e234cb7c1c01d761cece193b&scene=21#wechat_redirect>
3. <https://blog.csdn.net/Ephemeroptera/article/details/89478727#:~:text=%E9%AA%8C%E8%AF%81%E7%A0%81%E6%8A%80%E6%9C%AF%E4%BD%9C%E4%B8%BA%E4%B8%80%E7%A7%8D%EE%80%80%20%E5%8F%8D%E8%87%AA%E5%8A%A8%E5%8C%96%20%EE%80%80%E6%8A%80%E6%9C%AF%EF%BC%8C%E4%BD%BF%E5%BE%97%E5%BE%88%E5%A4%9A%E7%A8%8B%E5%BA%8F%E7%9A%84%E8%87%AA%E5%8A%A8%E5%8C%96%E5%B7%A5%E4%BD%9C%E6%AD%A2%E6%AD%A5%E3%80%82%EE%80%81%EE%80%81>
4. <https://blog.csdn.net/weixin_46641057/article/details/112620697>
5. <https://cloud.tencent.com/developer/news/362106>
6. 采集标注样本参考（通过网页返回结果确定是否标注成功）：

<https://github.com/kerlomz/captcha_trainer>

1. https://www.cnblogs.com/beer/p/5672678.html
2. <https://developer.aliyun.com/article/1053621>
3. 下载网页：验证码科普：

file:///Users/matianjia/Downloads/%E9%80%89%E6%8B%A9%E6%B5%8F%E8%A7%88%E5%99%A8%E6%89%93%E5%BC%8080636648.html

训练AI模型精准输出验证码<-模型预测验证码结果，RPA自动填写<-模型训练&优化<-训练数据预处理、数据标注

**试题名称：1.1 数据清理和处理**

## 数据清洗

\*请对captcha文件夹中的验证码图片进行清洗，输出最短边小于20的验证码图片名和总数量，并删除这些图片\*

代码一：

import cv2

import os

s=time.time()

def clean\_captcha\_images1(folder\_path,min\_length=20):

deleted\_files = []

for filename in os.listdir(filepath):

if filename.endswith(".png"):

file\_path = os.path.join(filepath, filename)

# print(file\_path)

img = cv2.imread(file\_path)

# print(img.shape) #高度\*宽度\*通道

i\_h,i\_w = img.shape[0],img.shape[1]

if min(i\_h,i\_w) < min\_length:

os.remove(file\_path) #删除指定路径的文件

deleted\_files.append(filename)

print(f"删除的图像名称:{filename}")

print(f"共删除{len(deleted\_files)}张尺寸不合格的脏数据")

filepath = f"./captcha"

s=time.time()

clean\_captcha\_images1(filepath)

e=time.time()

print(f"{e-s}s")

代码二：

from PIL import Image

def clean\_captcha\_images2(folder\_path):

deleted\_files = [] # 存储被删除文件的名称

for filename in os.listdir(folder\_path):

if filename.endswith('.png'): # 确保处理的是PNG格式的图片

file\_path = os.path.join(folder\_path, filename)

with Image.open(file\_path) as img:

width, height = img.size

# print(width,height)

if min(width, height) < 20:

os.remove(file\_path)

deleted\_files.append(filename)

# 输出删除的文件名称和总数量

for name in deleted\_files:

print(f"删除的图像名称：{name}")

print(f"共删除{len(deleted\_files)}张尺寸不合格的脏数据")

folder\_path = 'captcha'

s= time.time()

clean\_captcha\_images2(folder\_path)

e=time.time()

print(f"{(e-s)}s")

**试题名称：2.1数据标注和训练数据划分**

(1) 请根据此需求制定标注规范，标注规范需考虑如下内容。内容提交至“提交资料/ 题目2/模型训练.docx”文件中。

* 标注方法：采用什么标注方法;
* 标注范围：哪些情况需要标注，哪些情况不需要标注；
* 标注规则：如标注哪些验证码数据，样式不一致的数据，验证码严重遮挡的情况等；
* 特殊情况反馈途径：若遇到举棋不定的标注情况如何进行反馈；
* 交付的数据格式：限定数据交付数据。

1. 标注方法

* 定义：数据标注员进行数据标注时的环境和规程，包含标注对象定义、所用标注工具和标注平台、标注格式、标注前的准备工作、标注后的处理工作等。
* 方法：

全人工标注：基于百张验证码图片集的情况，由数据标注员经过标注培训后采用具备易操作性、规范性、高效性的标注工具按照既定的数据标注规范手动输入验证码文本作为标注结果

半自动化标注：基于千张～万张大量验证码图片集的情况，由经验丰富的数据标注员使用已有的OCR工具对图片进行预标注后，再由人工对结果进行审核。

1. 标注范围

* 需要标注的情况：人工可肉眼识别出、清晰且未经过严重遮挡、且可以识别出图片上所有字符的验证码图片
* 不需要标注的情况：图像质量差，如模糊不清、过度裁剪或严重遮挡影响字符识别的验证码图像。

1. 标注规则

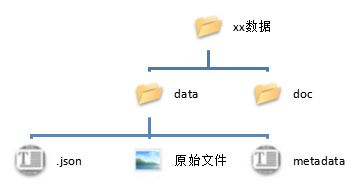
* 标注结果保持一致性：确保所有验证码图像标注的风格和格式一致（例如，所有字符均为大写或小写）。
* 对样式不一致的数据确保标注结果的准确性：对于字体、颜色和背景高度不一的验证码，标注时需要特别注意，确保字符识别正确。
* 验证码严重遮挡：对于严重遮挡的验证码，除非能准确识别出所有字符，否则应从标注集中排除。

1. 特殊情况反馈途径

* 特殊反馈渠道：通过电子邮件或内部系统提交不确定或有争议的图像，由专门团队讨论决定后续操作，比如采取多人标注或者集中处理该类特殊情况。
* 内部审核：标注员遇到不确定的情况应提交至审核队列，由经验更丰富的审核员决定如何处理。
* 规则定期更新：针对存疑数据或当前规则未覆盖的数据标注情况，经记录处理后需及时调整规则细则并同步参与者，并保证数据标注方及时、充分理解。

1. 交付的数据格式

* 交付数据存储结构



* **图像文件**：JPEG格式，命名为图像ID.jpeg。
* **标注文件**（label.txt）：txt文件，包括所有数据标注结果，每行格式为图像ID,验证码文本，例如1234.jpg,ABCD。

参考文献：

<https://blog.csdn.net/pdcfighting/article/details/120376321>

(2) 划分训练集与测试集，打开“划分数据“项目中data\_spilt.ipynb文件，需要完成如下任务：

待划分的数据集存储在“captcha”中，包含图像文件夹image和标注文件label.txt

1  按照 8:2 的比例将数据集划分为训练集与测试集, 以如下格式进行存储；image文件夹存储图像文件，label.txt存储标注文件。

2  将训练集和测试集的数量记录在“提交资料/题目2模型训练.docx”文件中；将此部分代码截图保存至“提交资料/题目2模型训练阶段.docx”文件中。下载划分好的数据集，将其保存至“提交资料/题目2/划分数据集”中

3、技能要求

（1）标注数据理解

（2）标注情况分析

（3）标注方法掌握

（4）编码能力

（5）数据理解能力

4、质量指标

（1）标注相关问题的回答全面性

（2）数据集划分的准确性

代码一：

import os

import shutil

import random

import time

# 读取图像文件夹和标注文件

image\_root\_path = f"./captcha/"

label\_path = f"./captcha/label.txt"

# 建立训练和测试数据的目录及标注文件

data\_root = f"./data/"

#创建文件夹，copy图片，重写标注文件

def create\_copy\_rewrite(image\_root\_path,data\_root,img\_dict,filenames,flag="train"):

print(f"{flag} data start...")

data\_path = os.path.join(data\_root,flag)

#创建train文件夹，test可复用

if not os.path.exists(data\_path):

os.makedirs(data\_path)

#训练图片写入train文件夹，test可复用

#写入train\_label,test\_label可复用

with open(os.path.join(data\_path,"label.txt"),"w") as fw:

for img in filenames:

shutil.copy(os.path.join(image\_root\_path,img),data\_path)

img\_data = img + " " + img\_dict[img] +"\n"

fw.write(str(img\_data))

def split\_dataset(image\_root\_path,label\_path,data\_root,train\_ratio = 0.8):

test\_ratio = 1-train\_ratio

#用字典存入验证码图片及标注结果

#key:图片，value:label;'fobkwqm4.png': '2483'

img\_dict={}

with open(os.path.join(image\_root\_path,"label.txt"),"r") as fr:

content = fr.readlines()

for img\_label in content:

img = img\_label.replace("\n","").split(" ")

img\_dict[img[0]] = img[1]

total\_filenames = list(img\_dict.keys())

train\_filenames = random.sample(total\_filenames,int(len(total\_filenames)\*train\_ratio)) #以80%随机切分训练集和数据集

test\_filenames = list(set(total\_filenames) - set(train\_filenames))

print(f"总图片数量:{len(total\_filenames)},训练集图片数量：{len(train\_filenames)},测试集图片数量:{len(test\_filenames)}")

print(f"训练集第一张图片:{train\_filenames[:1]},测试集第一张图片：{test\_filenames[:1]}")

create\_copy\_rewrite(image\_root\_path,data\_root,img\_dict,train\_filenames,flag="train")

create\_copy\_rewrite(image\_root\_path,data\_root,img\_dict,test\_filenames,flag="test")

image\_root\_path = "./captcha/"

label\_path = "./captcha/label.txt"

data\_root = "./data/"

s=time.time()

split\_dataset(image\_root\_path, label\_path, data\_root)

e=time.time()

print(f"总处理时长...{(e-s)}秒")

代码二：

def split\_dataset(image\_root\_path, label\_path, data\_root, train\_ratio=0.8):

# 读取标注文件

with open(label\_path, 'r') as file:

lines = file.readlines()

# 随机化数据集

random.shuffle(lines)

# 计算训练数据集的大小

train\_size = int(len(lines) \* train\_ratio)

# 创建训练集和测试集的目录

train\_dir = os.path.join(data\_root, 'train')

test\_dir = os.path.join(data\_root, 'test')

os.makedirs(train\_dir, exist\_ok=True)

os.makedirs(test\_dir, exist\_ok=True)

# 创建训练集和测试集的标注文件

train\_label\_path = os.path.join(train\_dir, 'label.txt')

test\_label\_path = os.path.join(test\_dir, 'label.txt')

train\_label\_file = open(train\_label\_path, 'w')

test\_label\_file = open(test\_label\_path, 'w')

# 分配文件到训练集和测试集

for i, line in enumerate(lines):

image\_name, label = line.strip().split(' ')

src\_path = os.path.join(image\_root\_path, image\_name)

if i < train\_size:

# 复制到训练集

shutil.copy(src\_path, train\_dir)

train\_label\_file.write(line)

else:

# 复制到测试集

shutil.copy(src\_path, test\_dir)

test\_label\_file.write(line)

# 关闭标注文件

train\_label\_file.close()

test\_label\_file.close()

# 输出数据集信息

total\_images = len(lines)

train\_images = train\_size

test\_images = total\_images - train\_size

print(f'验证码总数为：{total\_images}')

print(f'训练数据集数量：{train\_images}')

print(f'测试数据集数量：{test\_images}')

# 指定路径和参数

image\_root\_path = "./captcha/"

label\_path = "./captcha/label.txt"

data\_root = "./data/"

s=time.time()

split\_dataset(image\_root\_path, label\_path, data\_root)

e=time.time()

print(f"总处理时长...{(e-s)}秒")

大致思路：创建data下的train/test目录->读取label并且随机化数据，按比例切分训练和测试集->生成train\_label和test\_label,如果在train\_size以内则shutil.copy图片并写入label，否则shutil.copy到测试集及写入label->关闭标注文件

**试题名称：2.2模型训练和测试验证**

（看一下模型结构&增加数据预处理部分代码）

https://blog.csdn.net/qq\_37995260/article/details/100015034?utm\_medium=distribute.pc\_relevant.none-task-blog-2~default~baidujs\_baidulandingword~default-1-100015034-blog-103949063.235^v43^pc\_blog\_bottom\_relevance\_base9&spm=1001.2101.3001.4242.2&utm\_relevant\_index=4

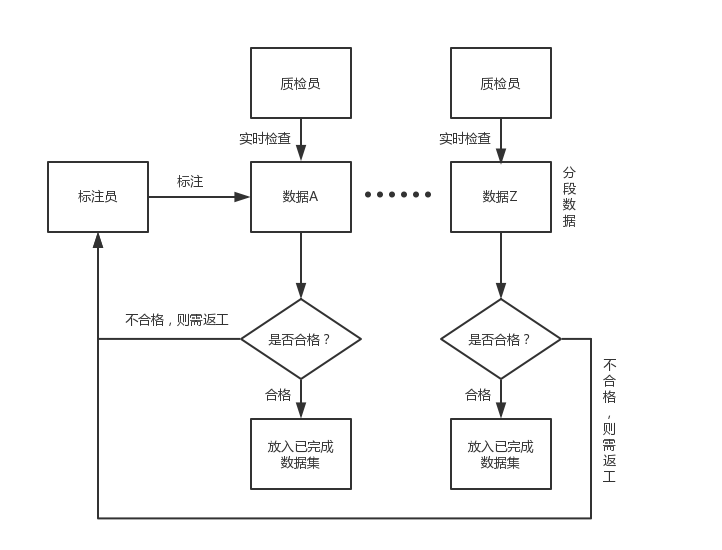
**试题名称：3.1数据质检和流程设计**

如何对验证码数据集标注结果进行质检，至少描述 2 种质检方法和流程，并应描述 每种方法的优劣势。内容提交至“提交资料/题目 3/智能系统设计.docx”文件中。

1. **面向数据标注过程的质量控制**

质量控制面向过程，确保标注过程可控，并产生预期的结果。在标注过程中，需要对数据质量及其行为进行规范和检测，及时预警反馈，查明低质量数据原因，以此控制标注数据的质量

**流程：**



**优势：**

* 能够及时发现问题并解决问题。
* 能够有效减少标注过程中重复错误的重复出现。
* 能够保证整体标注任务的流畅性。
* 能够实时掌握数据标准的任务进度。

**劣势：**

对于人员的配备及管理要求较高。

1. **面向数据标注结果的质量控制**
2. **人工质检**

**流程：**

* **抽样审核**：从标注的数据集中随机抽取一定比例（如10%）的数据进行人工审核。
* **全面复审**：对于关键应用或高风险区域的数据，执行100%的人工复审。
* **错误反馈**：审核员标识出错误或不确定的标注，并提供正确的标注。
* **修正和再训练**：根据反馈修正标注数据，并可能需要重新训练模型以提高精度。

**优势：**

* **准确性高**：人工质检可以准确识别复杂或模糊情况下的标注错误。
* **灵活性强**：人工审核可以根据实际情况灵活调整标注规则和质检重点。

**劣势：**

* **成本高**：需要大量的人力资源，特别是对大规模数据集进行全面复审时。
* **效率低**：人工审核速度慢，可能成为项目进度的瓶颈。

**2. 自动质检（基于规则或模型）**

**流程：**

* **规则检验**：定义一系列自动检验规则（例如，标注的字符长度应与验证码长度一致）。
* **模型验证**：利用一个已训练的OCR模型对标注结果进行验证，比较模型识别结果与人工标注是否一致。
* **异常检测**：使用统计方法或机器学习模型识别出标注数据中的异常或离群值。
* **反馈修正**：自动修正明显的错误，并将可疑数据标记供进一步人工审核。

**优势：**

* **效率高**：自动化流程大大加快了质检速度，适合大规模数据集。
* **成本低**：减少了对人工资源的依赖，降低了质检成本。

**劣势：**

* **准确性有限**：自动化工具可能无法处理复杂或模糊的标注场景。
* **灵活性差**：对于规则的适应性和灵活性有限，可能需要定期更新和调整

**试题名称：3.2数据处理**

（1）打开“智能系统设计”文件夹，将‘captcha’文件夹中图像样本数据格式修改为“jpg”。将此部分代码截图保存至“提交资料/题目 3/ 智能系统设计.docx”文件中。

（2）打开“智能系统设计”文件夹，将‘captcha’文件夹中图像样本名称重新定义，并利用数字编号。数据从“0000”开始计数，顺序累加，如0000.jpg，0001.jpg，0002.jpg….以此类推直至结束。将此部分代码截图保存至“提交资料 /题目 2 业务分析.docx”文件中。下载处理好的数据集，将其保存至“提交资料/题目 3/智能系统设计”文件夹中。

1. 技能要求
2. 编码能力
3. 数据处理能力
4. 质量指标
5. 代码的正确性
6. 数据处理结果的准确性

代码一：

import cv2

import os

data\_dir = './captcha'

# 获取captcha文件夹中的所有图片文件路径

img\_list = os.listdir(data\_dir)

#改格式

for old\_name in os.listdir(data\_dir):

if old\_name.endswith("png"):

new\_name = old\_name.replace("png","jpg")

img = cv2.imread(os.path.join(data\_dir,old\_name))

cv2.imwrite(os.path.join(data\_dir,new\_name),img)

os.remove(os.path.join(data\_dir,old\_name))

# 对图片文件进行重命名

for i, old\_name in enumerate(img\_list):

new\_name = "{:04d}.jpg".format(i)

os.rename(os.path.join(data\_dir, old\_name), os.path.join(data\_dir, new\_name))

代码二：

import os

from PIL import Image

data\_dir = './captcha'

# 获取captcha文件夹中的所有图片文件路径

img\_list = os.listdir(data\_dir)

# 转换图像格式为jpg并重命名

for i, old\_name in enumerate(img\_list):

# 定义新文件名

new\_name = '{:04d}.jpg'.format(i)

# 指定旧文件完整路径

old\_path = os.path.join(data\_dir, old\_name)

# 指定新文件完整路径

new\_path = os.path.join(data\_dir, new\_name)

# 打开旧文件

with Image.open(old\_path) as img:

# 转换格式并保存新文件

img.convert('RGB').save(new\_path, 'JPEG')

# 删除旧文件

os.remove(old\_path)

print(f"Processed {len(img\_list)} images.")

**试题名称：人工智能训练师（三级）-4-4.1人工智能训练师培训能力考核**

1. 注意事项
2. 看下四级、五级大纲，看他们需要掌握的范围和内容，每一步的指导有针对性的增加培训方法
3. 看下我司的数据挖掘规范
4. 重点讲解的知识点

现有答案的基础上增加内容：

1. 人工智能当前的行业趋势和未来方向
2. 对应行业业务基础知识、数据结构及数据流向

**1. 人工智能基础知识**

* **定义和历史**：介绍人工智能的基本概念、历史发展，以及主要学派和理论。
* **主要领域**：解释机器学习、深度学习、自然语言处理、计算机视觉等关键领域。
* **应用实例**：讨论AI在各行业中的实际应用，如医疗、金融、自动驾驶等。

**2. 数据采集与处理**

* **数据采集方法**：介绍如何从不同来源收集数据，包括网络爬虫、APIs、传感器数据等。
* **数据清洗技巧**：教授如何处理缺失值、异常值、重复数据等常见问题。
* **数据转换**：讲解数据标准化、归一化等处理技术，以及如何转换数据格式以适应分析和模型训练需求。

**3. 机器学习和深度学习基础**

* **算法介绍**：解释监督学习、无监督学习、强化学习等机器学习类型及其常用算法。
* **模型训练**：讲解如何选择模型、设置参数、训练模型以及如何评估模型性能。
* **工具和库**：介绍常用的机器学习和深度学习框架，如TensorFlow, PyTorch, scikit-learn等。

**4. 数据标注与质量控制**

* **标注工具与方法**：介绍常用的数据标注工具和方法，特别是针对特定类型数据（如图像、文本）的标注策略。
* **质量控制流程**：讨论如何确保数据标注的准确性和一致性，包括设置标注指南、进行交叉检验等。

**5. 伦理和法律问题**

* **伦理原则**：介绍在进行AI研究和开发时需要考虑的伦理问题，如偏见、隐私、透明度和责任。
* **法律合规**：讲解与人工智能相关的主要法律和规范，特别是数据保护法（如GDPR）和版权法。

1. **未来趋势**

* **AI应用新领域及发展现状**：讲解当前AI领域的新趋势和新应用，如大模型及落地场景

**7. 案例研究和实战练习**

* **案例分析**：提供实际案例分析，帮助学员理解理论如何应用于实际问题解决。
* **实践项目**：设计小型项目，让学员亲自动手实践从数据采集到模型部署的整个流程。

**8. 业务知识**

* **业务背景：**包括基于公司所在行业的业务背景知识、目标模型所在的业务流程
* **数据流向：**目标模型对应的数据结构及数据流向，包括输入、输出及上下游对接

**试题名称：人工智能训练师（三级）-4-4.2人工智能训练师指导能力考核**

人工智能训练师五级/初级工、四级/中级工解决数据采集、处理问题、解决数据标注问题使用的工具有哪些？

Ref：

四级、五级工初级了解：

<https://www.thepaper.cn/newsDetail_forward_15593533>

**数据采集工具**

1. **Python**:
   * **外部网页数据采集**：BeautifulSoup 和 Scrapy 用于网页爬虫；requests 用于API调用。
2. **数据库技术**:
   * **SQL**：用于管理结构化数据，其中大批量数据当前主要在集群上进行处理，包括Hive、Impala、Spark SQL等
   * **NoSQL数据库**：如 MongoDB，用于处理大规模、非结构化或半结构化数据。

**数据处理工具**

1. **Python**:
   * **库**：Pandas 用于数据操作和分析；NumPy 用于数值计算；Matplotlib、Seaborn用于可视化分析
2. **数据库技术**:
   * **SQL**：用于处理结构化数据，其中大批量数据当前主要在集群上进行处理，包括Hive、Impala、Spark SQL等，处理数据汇总
   * **NoSQL数据库**：如 MongoDB，用于处理大规模、非结构化或半结构化数据。
3. **数据清洗工具**:
   * **OpenRefine**：一种强大的工具，用于清理和转换数据，使其更加一致和整洁。
   * **Talend**：一个数据集成工具，提供广泛的数据清洗和转换功能。
4. **编程环境和IDE**:
   * **Jupyter Notebook**：提供一个交互式的编程环境，非常适合数据分析和机器学习的探索性研究。
   * **PyCharm** 或 **Visual Studio Code**：这些IDE支持Python和其他编程语言，提供代码编辑、调试和版本控制功能。

**数据标注工具**

1. **LabelImg**:
   * 一个开源的图像标注工具，适用于图像中物体检测的标注，可以导出为XML或TXT格式。
2. **LabelMe**:
   * 一种图形图像注释工具，适用于创建机器学习训练数据集。
3. **CVAT (Computer Vision Annotation Tool)**:
   * 一个为计算机视觉任务设计的开源标注工具，支持视频和静态图像标注。
4. **Prodigy**:
   * 一款付费的注解工具，支持自然语言处理和计算机视觉任务，能够实现快速有效的数据标注。
5. **Doccano**:
   * 一个开源项目，用于为机器学习项目注释文本数据，支持多种标注类型，包括情感分析和实体识别。