《软件实践》课程实验报告

暑期学校实验项目：高考志愿填报助手

小组名称	知识推理与图谱可视化小组						
姓　　名	陈震寰	专业	人工智能	班级	091181	学号	09118132
实验时间	2020.8.31-2020.9.23		指导教师	孔祥龙		成绩	

一、实验背景和目的

实验背景：大二学年学习了 python 编程，数据库，机器学习等相关课程，但学习的大多是理论知识，缺乏项目实践经历。使用 python 进行网站开发是对之前课程很好的回顾，能通过亲身实践加深对之前课程的理解，并熟悉项目开发的流程。本项目需要开发一个为高考后即将填志愿的学生提供咨询的网站。

实验目的：

1. 通过实践巩固所学知识。
2. 了解项目开发的过程，熟悉团队协作开发项目的方式。
3. 熟悉在码云上开发项目的操作方法。
4. 熟悉 django 框架。
5. 巩固之前所学知识。

二、小组任务和个人任务

小组任务：

我们小组最初的定位是基于 3、4 组的知识图谱进行可视化和知识推理，但在实际推进的时候，由于原定工作需要等知识图谱组的结果（实际最后一周才拿到结果），而知识图谱组的工作又基于数据库，并且知识图谱组自己进行了可视化，所以我们在孔老师的大方向指导下，经过组内探讨，讨论出了几个可行的任务：

1、依据高校分数线进行统计分析，生成相应图表，展示变化趋势；
2、根据用户输入的动态个人信息进行相应的可视化；
3、以地图为依据生成可视化界面，展示高校信息；
4、基于知识图谱的推理和问答；
5、对应的 django 框架编写。

由于小组定位的原因，我们组的工作在实际推进中大部分都要等到前面的组做好或者已经被做好了，所以我们组的工作并不是一起完成一个任务，而是分成各个小组分别完成各自的小任务。

个人任务：

1. 我主要负责根据用户输入的动态信息进行相应的可视化（此项任务由我和张立创合作完成）。首先我进行了此项任务的概念设计，并负责写了 views 中所有相关的实现功能的函数（此任务的前端界面由张立创制作，功能测试由我们共同完成）。

2. 同时，我发现数据库中各高校的专业名称各不相同，如果要利用这些专业对应的录取分数线，首先要将这些五花八门的专业名称归类到对应的一级学科，于是我编写了算法将专业名和一级学科、学科大类对应，并上传了相关文件和结果的 csv 文件。注意

到数据库 Majors 表中 firstlevelID 属性全部是 NULL，可以使用此方法在每次新增数据时获取对应的 firstlevelID。同时我发现数据库中一级学科门类不全（少了 30 条左右），我爬取了最新的教务部公布的学位授予和人才培养学科目录中的学科门类和一级学科名称，并上传了 csv 文件，希望有可能的话可以更新一下数据库（获取专业对应的一级学科时使用的是自己爬取的一级学科，不是数据库中的）。

三、个人任务需求分析

1. 根据用户输入的动态信息进行相应的可视化

此功能是根据用户动态输入的信息进行相应的可视化。用户在刚高考完出分后，可能对自己能上什么样的学校，想上什么样的学校或专业没有特别明确的认识，只有一个大概的概念，甚至不知道自己具体想要获取的信息是什么。比如，一个学生在江苏高考考了 370 分，他可能想上 985 高校，也可能想上排名较高的 211 高校；可能想上江苏的高校，也可能想上外省的高校等等，或者说在使用系统之前他根本没有想过这些具体的问题。并且，我们认为系统如果只能根据用户信息直接推荐一个或者几个学校有不合理的地方，因为这样的推荐过于局限，并且存在不可解释性。并且如果用户根据推荐结果填报志愿而没有被录取，可能会有一些不好的后果。所以我们设计的初衷就是让用户填上一些信息，根据这些信息分析用户可能想了解什么样的具体信息，然后以问题的形式显示出来，让用户自己选择，根据用户选择的具体问题，我们将相应的结果以图表的方式动态显示出来，即将数据库中用户想知道的信息显示出来，给用户一个整体上的认识，而不是直接的学校或专业推荐。

个人任务需求包括：

(a)设计用户需要填的信息，包括必填和选填。

(b)设计不同的信息填入情况对应的问题。

(c)设计不同的问题需要进行怎样的可视化。

(d)编写对应的 views 中的功能函数，包括根据输入的信息以及选填信息的填写情况生成问题的函数 InfoIntoQuestions(request)、根据前端传回用户选择的问题选择调用哪个具体的功能函数的函数 ChooseFunction(request)、生成图表函数 VisualazationCollegeScore(collegelist, collegeMinScorelist, figureName)，以及各种实现对应数据提取功能函数(12 个 QuestionsIntoAnswer 开头的函数，主要是从数据库取数据，用于生成图表)以及其他功能函数(getcolLegeminscore/GetCollEgeAverageMinScore/getSinilarity 等等)。

2. 编写算法获取各高校不同专业名称对应的一级学科列表

此任务的个人任务需求包括：

要利用数据库中专业录取分数线，就需要处理数据库中五花八门的专业名称，找到他们对应的一级学科，用户选择了目标一级学科名称后，使用 Django 模糊查询方法在 Majors 表中查找符合的专业录取分数效果并不理想，原因可能有很多专业名与对应一级学科名不是包含关系，或基本没有字段相同等等，导致如果用模糊查询，不同的一级学科名称需要使用不同的模糊查询方法，可行性很低。

所以使用 Levenshtein.ratio 方法编写算法，首先获取数据库 Majors 表中不同专业名称（共十余万条，去重后共两千余条）对应的一级学科名称。（由于数据库 firstlevel 表中一级学科

数目缺了约三十条，此任务使用的一级学科 csv 是自己爬取的），接着排序后去除专业名中的无关字符，最后使用 Levenshtein.ratio 方法判断匹配的一级学科名称。根据 Majors 表中的数据，目前采用大类招生学校相对较少，所以默认专业名称和一级学科为一一对应关系。

3. 补充：任务概念设计中用户输入信息、生成的问题以及对应的可视化

必填：1. 高考省份（下拉框）、2.高考分数（手动填写）、3.文/理科（下拉框）
选填：1. 省份倾向（下拉框） 2.专业倾向（下拉框）

(a)若选填的内容都没填

问题	函数名	实现功能
省*科分冲一冲能上什么学校？	QuestionsIntoAnswer11	生成在**省*科分数线属于[score, score+20]的大学名称及分数线(柱状图)
省*科分稳一稳能上什么学校？	QuestionsIntoAnswer12	生成在**省*科分数线属于[score−10, score+10]的大学名称及分数线(柱状图)
省*科分保一保能上什么学校？	QuestionsIntoAnswer13	生成在**省*科分数线属于[score−20, score]的大学名称及分数线(柱状图)
省*科分能上什么 985 学校/211 学校/本科学校？(这里根据分数和考试省份自动判断，比如江苏考 370 分，则既有“985 学校”对应的问题，也有“211 学校”对应的问题)	QuestionsIntoAnswer14	生成在**省*科分数线属于[score−15, score+15]的985/211/本科大学名称及分数线(柱状图)

(b)选填内容只填了省份倾向：

问题	函数名	实现功能
省*科分冲一冲能上***省的什么学校？	QuestionsIntoAnswer21	生成在**省*科分数线属于[score, score+20]的***省的大学名称及分数线(柱状图)
省*科分稳一稳能上***省的什么学校？	QuestionsIntoAnswer22	生成在**省*科分数线属于[score−10, score+10]的***省的大学名称及分数线(柱状图)
省*科分保一保能上***省的什么学校？	QuestionsIntoAnswer23	生成在**省*科分数线属于[score−20, score]的***省的大学名称及分数线(柱状图)

***省大学在**省*科的录取分数线？	QuestionsIntoAnswer24	生成***省大学在**省**科 2019 年录取分数线(柱状图)（由于数据库中有一些 2018、2017 年的数据没有，此处只用 2019 年的录取线，（但是 views 中写了获得三年平均的函数），以下几个获得录取分数线的函数同样都是 2019 年的）
省*科分能上**省的什么 985/211/本科学校?	QuestionsIntoAnswer25	生成在**省*科分数线属于[score−15, score+15]的***省的 985/211/本科大学名称及分数线(柱状图)

(c)选填内容只填了专业倾向：

问题	函数名	实现功能
**专业全国大学在本省的录取分数线？	QuestionsIntoAnswer31	生成**专业全国大学在**省的录取分数线(柱状图)
省*科*分能上哪些学校的专业？	QuestionsIntoAnswer32	生成**专业全国大学在**省的录取分数线属于[score−15, score+15]的分数线(柱状图)
省*科分冲一冲能上什么学校？	QuestionsIntoAnswer11	生成在**省*科分数线属于[score, score + 20]的大学名称及分数线(柱状图)
省*科分稳一稳能上什么学校？	QuestionsIntoAnswer12	生成在**省*科分数线属于[score−10, score+10]的大学名称及分数线(柱状图)
省*科分保一保能上什么学校？	QuestionsIntoAnswer13	生成在**省*科分数线属于[score−20, score]的大学名称及分数线(柱状图)
省*科分能上什么/985 学校/211 学校/本科学校?	QuestionsIntoAnswer14	生成在**省*科分数线属于[score−15, score+15]的***省的 985/211/本科大学名称及分数线(柱状图)

(d)选填内容都填了

问题	函数名	实现功能
专业*省的大学在本省录取分数线？	QuestionsIntoAnswer41	生成**专业***省的大学在**省的录取分数线(柱状图)
省*科*分冲一冲能上*省的什么学校？	QuestionsIntoAnswer21	生成在**省*科分数线属于[score, score+20]的***省的大学名称及分数线(柱状图)
**省*科*分稳一稳能上	QuestionsIntoAnswer22	生成在**省*科分数线属于

***省的什么学校？		[score－10, score+10]的***省的大学名称及分数线(柱状图)
省*科*分保一保能上*省的什么学校？	QuestionsIntoAnswer23	生成在**省*科分数线属于[score－20, score]的***省的大学名称及分数线(柱状图)
省*科分能上***省的什么 985 学校/211 学校/本科学校？	QuestionsIntoAnswer25	生成在**省*科分数线属于[score－15, score+15]的***省的 985/211/本科大学名称及分数线(柱状图)
**专业全国大学在本省的录取分数线？	QuestionsIntoAnswer31	生成**专业全国大学在**省的录取分数线(柱状图)

四、实验过程（需附上关键代码及相关说明）

1. 对系统进行概念设计

2. 设计必选/填项：高考省份、高考分数、文/理科，和选填/选项：省份倾向、专业倾向

3. 设计不同的选填内容完成情况对应的问题，见上表。由 InfoIntoQuestions(request)函数根据不同的选填内容完成情况，生成对应的问题：

首先从前端获取参数：

```
province = request.POST.get('province')
score = request.POST.get('score')
score = int(score)
category = request.POST.get('subject')
targetprovince = request.POST.get('Tprovince')
targetmajor = request.POST.get('Tmajor')
```

当选填内容都没填写时，首先生成前三个问题：

```
if(targetprovince == "all" and targetmajor == "all"):
    questionhead = province + category + str(score) + "分"
    question11 = questionhead + "冲一冲能上什么学校？"
    question12 = questionhead + "稳一稳能上什么学校？"
    question13 = questionhead + "保一保能上什么学校？"
    questions.append(question11)
    questions.append(question12)
    questions.append(question13)
```

在根据输入的分数生成第 4/5 个问题（比如考生在江苏考了 370 分，那么他可能想上 985，也可能想上比较好的 211，那么就生成两个问题，分别是“江苏省理科 370 分能上什么 985 学校？”、“江苏省理科 370 分能上什么 211 学校？”，但如果考生在江苏考了 410 分，那么他只可能想上 985 学校，问题也只会出现前者）：

```
if(province == "江苏"):
    if(score >= 390):
        question14 = questionhead + "能上什么985学校?"
        questions.append(question13)
    if (score >= 370 and score < 390):
        question14 = questionhead + "能上什么985学校?"
        question15 = questionhead + "能上什么211学校?"
        questions.append(question14)
        questions.append(question15)
    if (score < 370 and score >= 360):
        question14 = questionhead + "能上什么211学校?"
        question15 = questionhead + "能上什么一本学校？"
        questions.append(question14)
        questions.append(question15)
    if(score < 360):
        question14 = questionhead + "能上什么一本学校？"
        questions.append(question14)
```

同时，不同高考总分的省份能上 985/211/本科的大致分数段不同，所以生成此问题时需要对省份进行判断。

余下代码类似，此处省略。

4. 当信息输入完成并生成对应的具体问题后，由用户进行选择，前端将选择的问题传回。调用 ChooseFunction(request)函数，首先判断选填内容填写情况，根据列出对应的问题模型，调用 Levenshtein.ratio 方法（封装为 getSimilarity 函数），判断是哪个问题模型，并调用相应的函数。

比如，用户填写了选填信息中的倾向省份，但没有填写倾向专业，则函数首先使用 request.POST 方法获取参数，判断选填内容填写情况，并显示此情况对应的问题模型，分别为"**省*科**分冲一冲能上***省的什么学校？"、"**省*科**分稳一稳能上***省的什么学校？"、"**省*科**分保一保能上***省的什么学校？"、"***省大学在**省*科的分数线排名？"、"**省*科**分能上**省的什么 985/211/本科学校？"：

```
if(tprovince != "all" and tmajor == "all"):
    question11 = "**省*科**分冲一冲能上***省的什么学校？"
    question12 = "**省*科**分稳一稳能上***省的什么学校？"
    question13 = "**省*科**分保一保能上***省的什么学校？"
    question14 = "***省大学在**省*科的分数线排名？"
    question15 = "**省*科**分能上**省的什么985/211/本科学校？"
```

然后将问题模型与取得的 question 参数（用户选择的问题）进行比较，判断 question 属于哪个问题模型（getSimilarity 函数使用了 Levenshtein.ratio 方法，判断字符串相似度，相似度最大的列表索引为 question 对应的问题模型的索引），然后调用问题模型对应的函数：

```
if(tprovince != "all" and tmajor == "all"):
    question11 = "**省*科**分冲一冲能上***省的什么学校？"
    question12 = "**省*科**分稳一稳能上***省的什么学校？"
    question13 = "**省*科**分保一保能上***省的什么学校？"
    question14 = "***省大学在**省*科的分数线排名？"
    question15 = "**省*科**分能上**省的什么985/211/本科学校？"
    questionModelList = [question11, question12, question13, question14, question15]
    similarity = [getSimilarity(question, i) for i in questionModelList]
    quesindex = similarity.index(max(similarity))
```

使用 python switch 方法进行函数调用：

```
switch = {
    0: QuestionsIntoAnswer21,
    1: QuestionsIntoAnswer22,
    2: QuestionsIntoAnswer23,
    3: QuestionsIntoAnswer24,
    4: QuestionsIntoAnswer25
}
switch[quesindex](request)
```

当选填内容填写情况与上述情况相异时，同理。

4. 调用由 ChooseFunction(request)选择的函数。比如，还是上述情况，用户选择的问题是“江苏省理科 410 分冲一冲能上江苏省的什么学校？”则 ChooseFunction(request)函数调用 QuestionsIntoAnswer21(request)。QuestionsIntoAnswer21 函数首先使用 request.POST.get 方法获取参数，然后从数据库中获取参加高考省份、科类、倾向省份的 id：

```
question = request.POST.get('question')
score = request.POST.get('score')
score = int(score)
province = request.POST.get('province')
category = request.POST.get('category')
tprovince = request.POST.get('Tprovince')
```

```
#获得provinceid、categoryid和tprovinceid
provinceid = Provinces.objects.filter(provinceName = province)[0].provinceID
categoryid = Category.objects.filter(categoryname = category)[0].categoryID
tprovinceid = Provinces.objects.filter(provinceName = tprovince)[0].provinceID  #目标省份的id
```

获取在考生所在省份的录取分数在考生分数~考生分数+20 之间的大学列表：

```
AllCollegelist = Colleges.objects.filter(provinceID_id = tprovinceid).distinct()
collegeMinScorelist = []
collegelist = []
for i in AllCollegelist:
    collegeid = i.collegeID
    collegescore = GetCollegeMinScore(collegeid, provinceid, categoryid, 2019)
    print(collegeid)
    if (collegescore >= score and collegescore <= score + 20):
        collegeMinScorelist.append(collegescore)
        collegelist.append(i)
```

```
#生成可视化图表
collegeNamelist = [i.collegeName for i in collegelist]
VisualazationCollegeScore(collegeNamelist, collegeMinScorelist, "")
```

并将相关参数返回前端，此处代码省略。共写了 12 个以“QuestionsIntoAnswer”开头的函数，对应于不同的问题模型，其余 QuestionsIntoAnswer 函数虽然实现逻辑不同，但是使用的方法大同小异，由于函数过多此处不再赘述。

5. 调用 VisualazationCollegeScore 函数进行可视化：

使用 matplotlib.pyplot 进行绘图，传入的参数为大学名称列表和录取分数线列表。设置 plt.xticks(rotation = -45)使得生成 x 轴文字倾斜 45 度；在 plt.savefig()中设置 bbox_inches = 'tight' 使得保存的图片边界对应于生成的图片边界。

```
def VisualazationCollegeScore(collegelist, collegeMinScorelist, figureName):
    x = [i for i in range(len(collegelist))]
    width = 0.2
    index = np.arange(len(collegelist))

    for xx, yy in zip(x, collegeMinScorelist):
        plt.text(xx, yy + 2, str(yy), ha='center')
    figsize = (20, 10)

    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False

    plt.bar(index, collegeMinScorelist, width, color="#87CEFA")
```

```
plt.ylabel('score', fontsize = 20)
plt.xlabel('colleges', fontsize = 20)
plt.xticks(rotation = -45)
plt.title(figureName)
plt.xticks(index, collegelist, fontsize=10)
plt.yticks(fontsize=15)
plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.3)
plt.savefig('./static/images/test.png', dpi = 400, bbox_inches = 'tight')
```

6. 功能函数：getcollegeminscore/GetCollegeAverageMinScore：

由于数据库中没有给出各大学录取最低分，只给了不同专业的录取分数线，所以我编写了 getcollegeminscore 函数用于获取大学在某省份某一年最低录取分数，函数参数为 collegeid：大学 id，provinceid：省份 id，categoryid：科类 id，即文/理/综对应 id， year：年份：

```
def GetCollegeMinScore(collegeid, provinceid, categoryid, year):                #获得学校在某个省文科/理科某一年的最低录取分数
    collegeMajorScoreList = Majors.objects.filter(collegeID_id = collegeid,
                                                  provinceID_id = provinceid, categoryID_id = categoryid, year=year)
    scores = []
    for i in collegeMajorScoreList:
        scores.append(int(i.minScore))
    minscore = 0
    if len(scores):    #若列表不为空
        minscore = min(scores)
    return minscore
```

GetCollegeAverageMinScore 函数用于获取历时三年最低录取分数的均值：

```
def GetCollegeAverageMinScore(collegeid, provinceid, categoryid):          #获得学校在某个省文科/理科三年的最低录取分数
    average = (GetCollegeMinScore(collegeid, provinceid, categoryid, 2019)+
               GetCollegeMinScore(collegeid, provinceid, categoryid, 2018)+
               GetCollegeMinScore(collegeid, provinceid, categoryid, 2017))/3
    return average
```

由于数据库中有部分历史数据缺失，调用时主要使用 getcollegeminscore 函数，即只获得 2019 年录取最低分。

7. 功能函数 getSimialrity(str1,str2)

使用 Levenshtein.ratio 方法获得两个字符串之间的相似度：

```
def getSimilarity(str1, str2):
    return Levenshtein.ratio(str1, str2)
```

该函数计算莱文斯坦比，公式为 r = (sum - ldist) / sum, sum 是 str1 和 str2 字串的长度总和，ldist 为类编辑距离，即将 str1 变成 str2 需要的最少操作次数，其中删除、插入算 1 次操作，替换算 2 次操作。该函数考虑了字符串总长度，效果比较好。生成的相似度数值越大，说明两个字符串的相似程度越高。用于判断选择的问题属于哪个问题模型，以及判断某个专业名称是否属于某学科大类。

在相关功能函数中嵌入了此方法，比如在 QuestionsIntoAnswer41 中若专业名和一个一级学科之间的相似度>0.5，则判断该专业属于这个一级学科（观察之前生成的 csv 文件发现一般情况下，相似度大于 0.5 时该专业就属于这个一级学科）

```
for i in similaritylist:
    if (i > 0.5):
        targetlist.append(tobjectslist[similaritylist.index(i)])
```

8.2. 编写算法获取各高校不同专业名称对应的一级学科列表

使用 Django 模糊查询方法在 Majors 表中查找符合的专业录取分数效果并不理想，原因可能有很多专业名与对应一级学科名不是包含关系，或基本没有字段相同等等。首先获取数据库 Majors 表中不同专业名称（共十余万条，去重后共两千余条）对应的一级学科名称。（由于数据库 firstlevel 表中一级学科数目缺了约三十条，此任务使用的一级学科 csv 是自己爬取的，已提交至小组仓库中），接着进行去重并排序：

```
data = pd.read_csv("Majors.csv", encoding="gbk")
majors = data["majorName"].tolist()

#去重
majors = set(majors)
majors = list(majors)
majors.sort()
dfMajors = pd.DataFrame(majors, columns=['major'])
```

专业名称中有很多对于判断专业所属一级学科无关的字符，首先去掉这些字符，使用 string.digits 方法去除数字，使用正则表达式 re.sub 方法去除无关符号或无关的单个文字，由于 re.sub 方法会替换原字符串中模式字符串的所有子串，所以在去除无关短语时，使用

string.replace 方法：

```
def removeUnrelatedStr(string):
    #去除数字
    remove_digits = string.maketrans('', '', digits)
    string = string.translate(remove_digits)
    #去除无关字符/文字
    string = re.sub(r'[+十“”，类（）批()？?+、含#H班【年】类]*', '', string)
    string = re.sub(r'[一二三四五六七八九十]*', '', string)
    string = re.sub(r'[ABCDE]*', '', string)
    #去除无关短语
    string = string.replace('一体化', '')
    string = string.replace('预备班', '')
    string = string.replace('预科班','')
    string = string.replace('中外合作办学', '')
    string = string.replace('实验班', '')
    string = string.replace('双语', '')
    string = string.replace('普通', '')
    string = string.replace(' ', '')
    string = string.replace('教学','')
    string = string.replace('合作','')
    string = string.replace('国防生','')
    string = string.replace('免费','')
    string = string.replace('公费','')
    string = string.replace('基地班','')
    string = string.replace('卓越','')
    string = string.replace('国家','')
    string = string.replace('试点','')
    string = string.replace('单列','')
    string = string.replace('民族','')
    string = string.replace('国家专项','')
    string = string.replace('提前','')
    return string
```

最后使用 Levenshtein.ratio 方法判断匹配的一级学科名称。根据 Majors 表中的数据，目前采用大类招生学校相对较少，所以默认专业名称和一级学科为一一对应关系。对于个别特殊情况进行判断并单独处理：

```
for i in Majors:
    firstSimilarity = []
    for j in firstLevelDisciplineList:
        similarity = getSimilarity(i, j)
        firstSimilarity.append(similarity)
    index = firstSimilarity.index(max(firstSimilarity))
    firstLevelName = firstLevelDisciplineList[index]
    if(max(firstSimilarity) == "人工智能"):
        firstLevelName = "计算机科学与技术"
    if(i.find("语") != -1 and i.find("汉") == -1 and i.find("双语") == -1 and i.find("中国") == -1):
        firstLevelName = "外国语言文学"
    if(i.find("会计") != -1):
        firstLevelName = "工商管理"
    if(max(firstSimilarity)==0.0):
        firstLevelName = "！！！"
    first.append(firstLevelName)
```

最后再匹配一级学科对应的学科门类，生成“matchedFirstLevel6.csv”，最终准确率可达 95%以上，手动修改了约 2%的数据。

虽然项目中不能使用 csv 文件，但是 views 中相关函数内嵌入了此方法。注意到数据库 Majors 表中 firstlevelID 属性全部是 NULL，可以使用此方法在每次新增数据时获取对应的 firstlevelID。

五、实验结果与分析

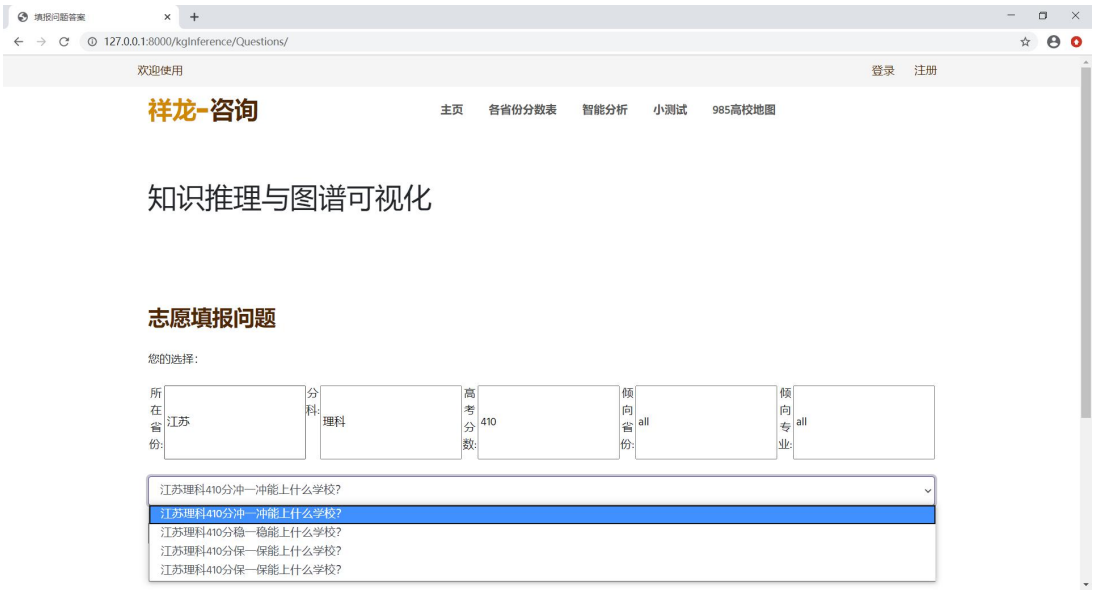
1. 实验结果

1.1 根据用户输入的动态个人信息进行相应的可视化

项目主页如下所示：

必填项包括省份，文/理科，使用下拉框，成绩为手动填写，可以上调/下调。选填项为倾向省份和倾向专业（数据库 firstlevel 表中对应一级学科）。

选择了相关信息后，进入问题生成页面：

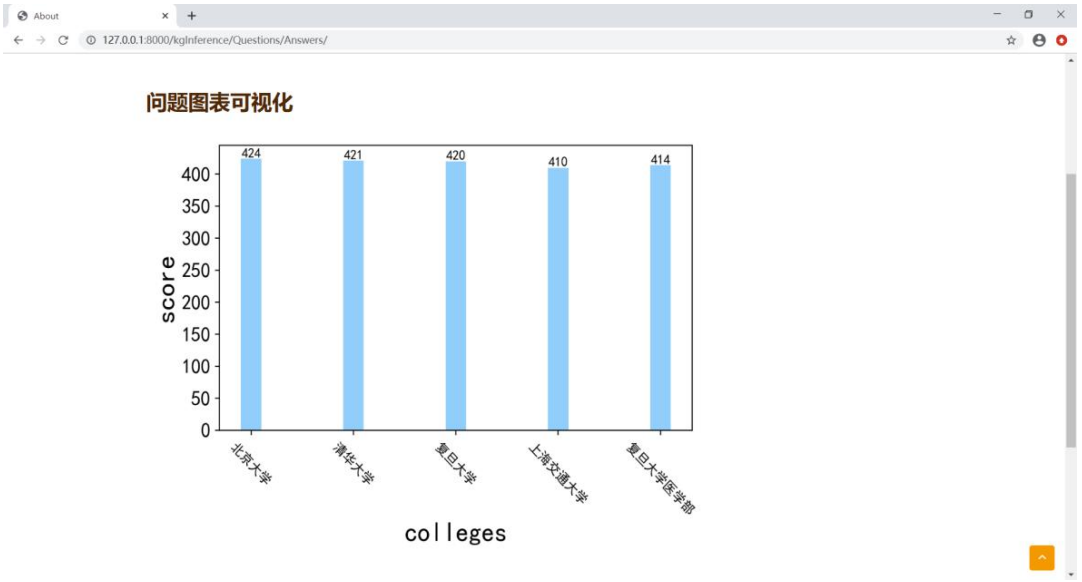


选择问题后点击“提交，进入可视化页面”，将在下面展示。

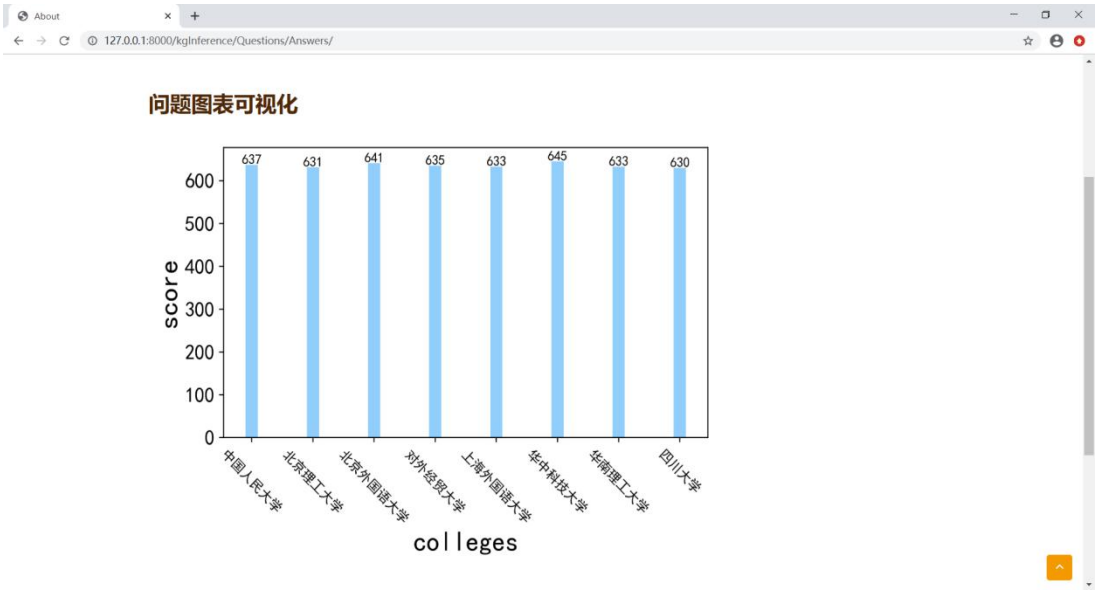
1.1.1 不输入选填信息

选择“江苏”、“理科”、“410 分”，选择问题“江苏理科 410 分冲一冲能上什么学校？”：

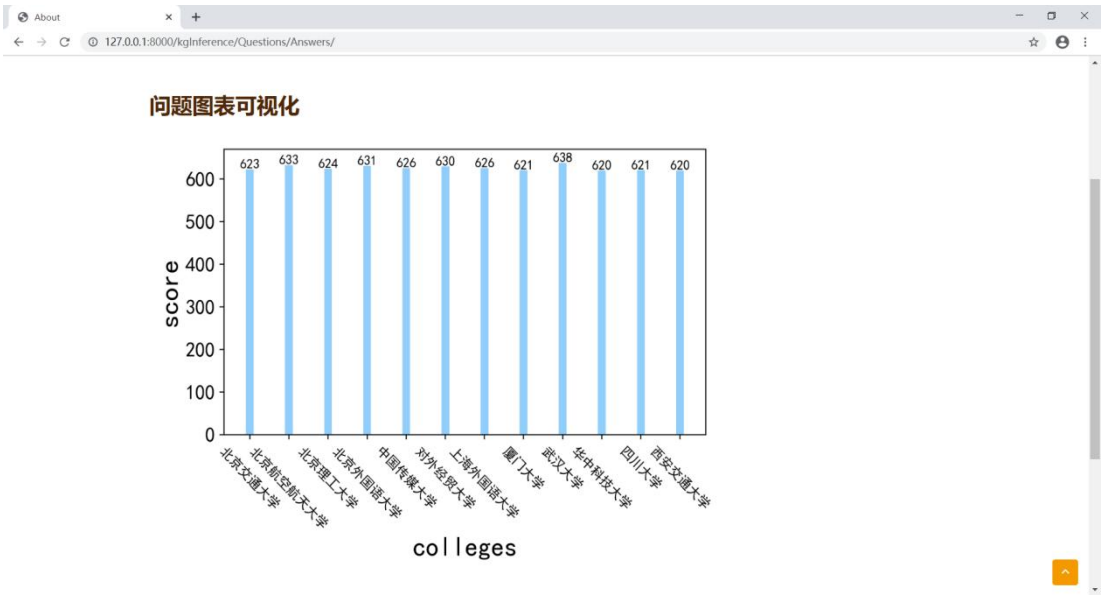
显示相应可视化结果：



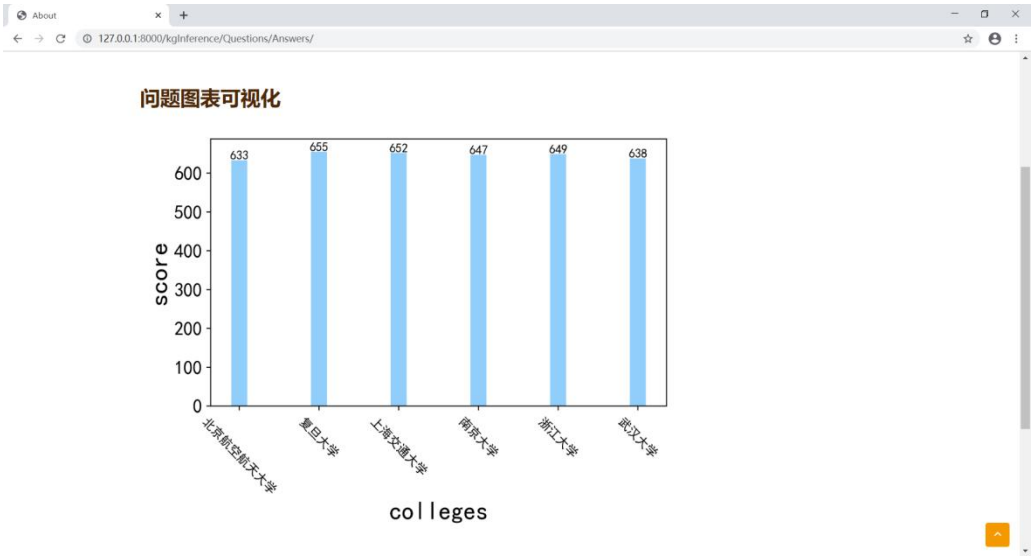
选择“山东”、“理科”、“640 分”，选择问题“山东理科 640 分稳一稳能上什么学校？”：



选择“北京”、“文科”、“640 分”，选择问题“北京文科 640 分保一保能上什么学校？”：

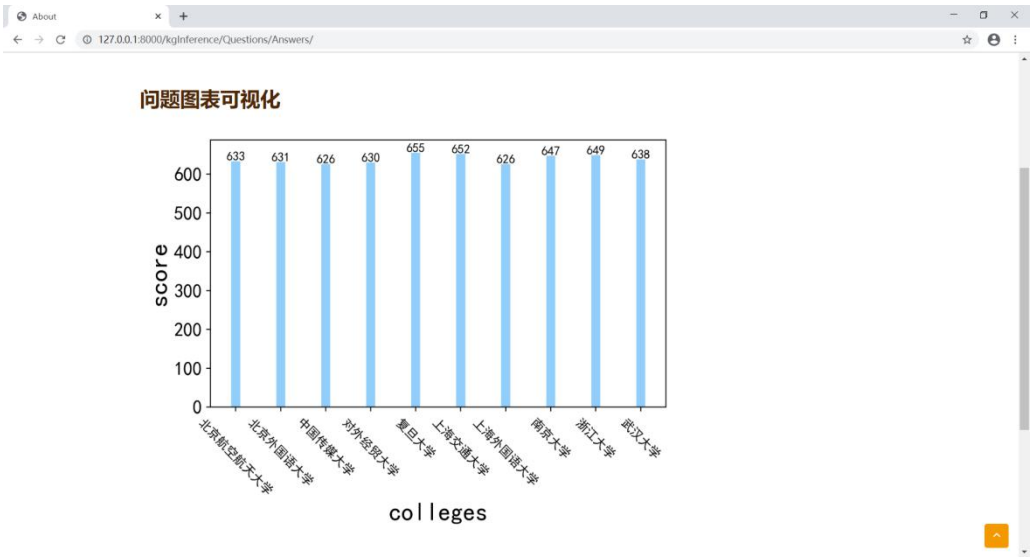


选择“北京”、“文科”、“640 分”，选择问题“北京文科 640 分能上什么 985 学校？”：



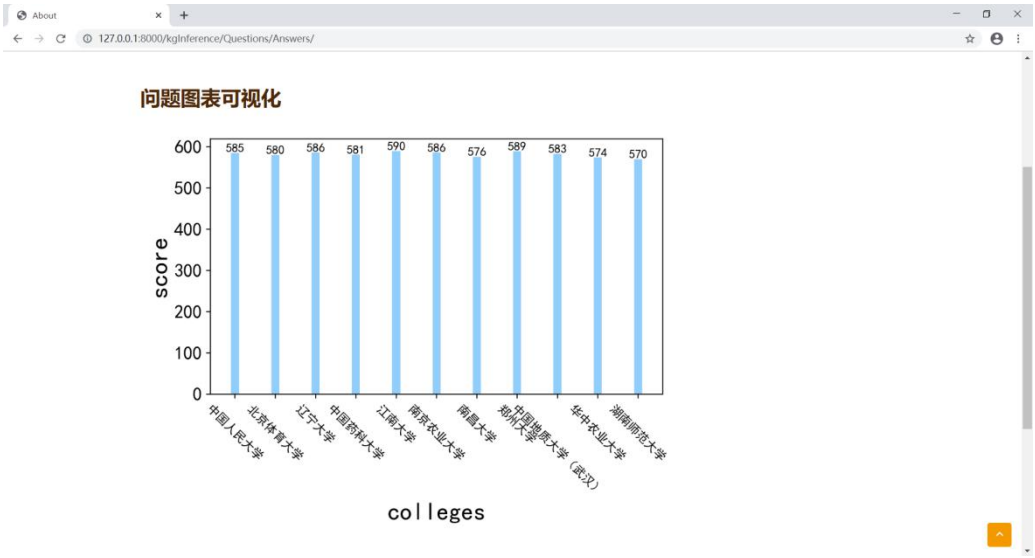
可见生成[score-15,score+15]范围内的 985 院校。因为选择的范围和“保一保”问题不同，所以图中推荐的 985 学校不完全等同于上一张图中的 985 学校。

选择“北京”、“文科”、“640 分”，选择问题“北京文科 640 分能上什么 211 学校？”：



可见显示的学校增加了，不仅有刚才图中的 985 学校，还有符合条件的 211 非 985 学校。

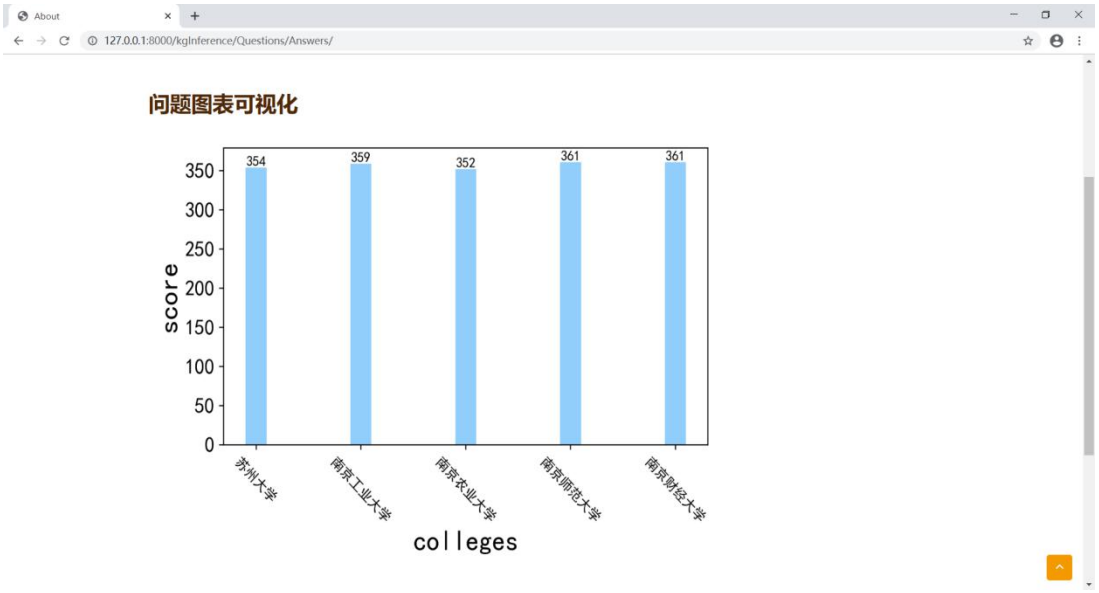
选择“北京”、“文科”、“570 分”，此时不会出现“北京文科 570 分能上什么 211 学校？”、“北京文科 570 分能上什么 985 学校？”这两个问题，选择“北京文科 570 分能上什么本科学校？”：



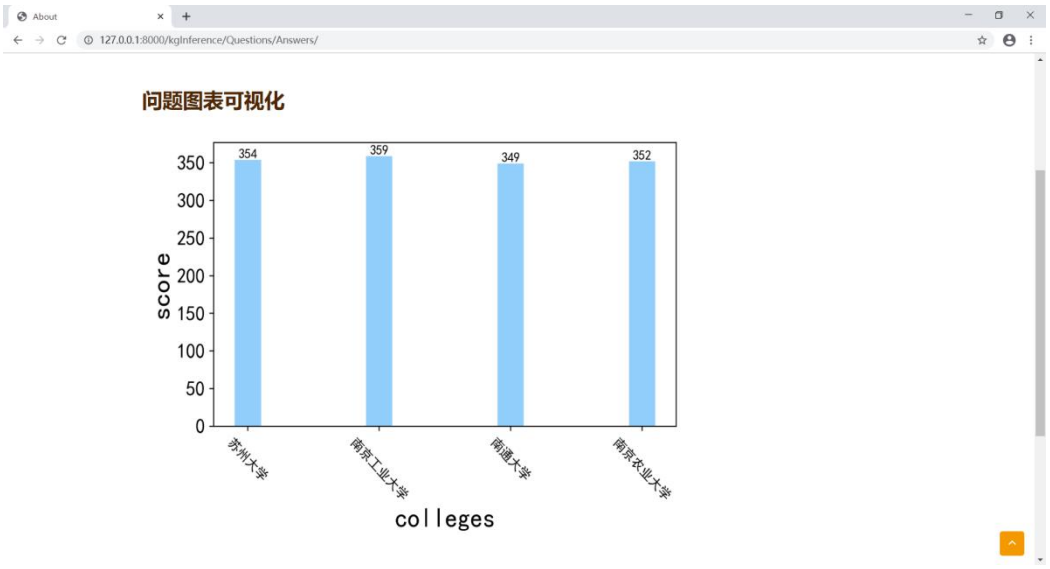
1.1.2 选填信息中选上倾向省份

选择“江苏”、“理科”、“350 分”，倾向省份选择“北京”，选择问题“江苏理科 350 分冲一冲能上北京的什么学校？”：

选择“江苏”、“理科”、“360 分”，倾向省份选择“江苏”，选择问题“江苏理科 360 分稳一稳能上江苏的什么学校？”：

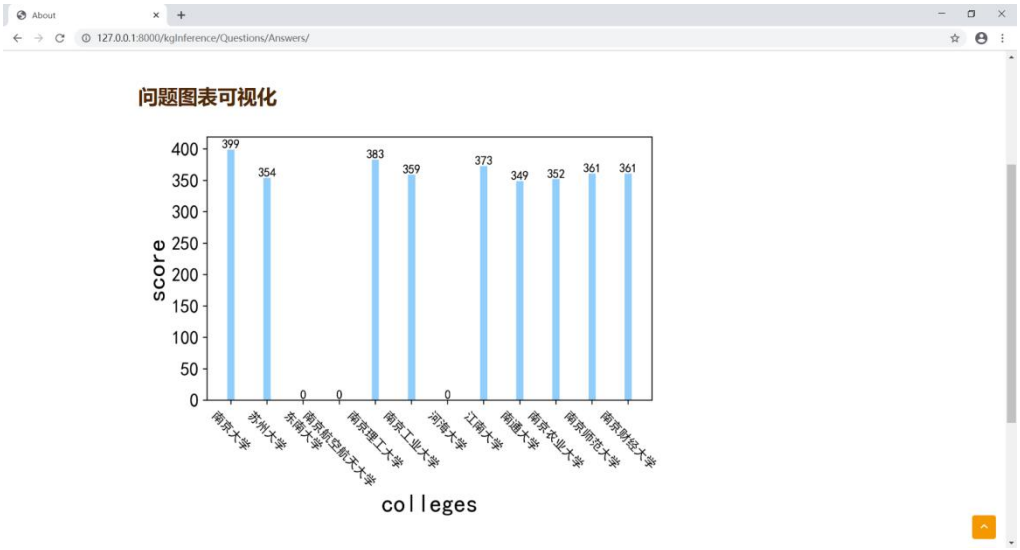


选择“江苏”、“理科”、“360 分”，倾向省份选择“江苏”，选择问题“江苏理科 360 分保一保能上江苏的什么学校？”：



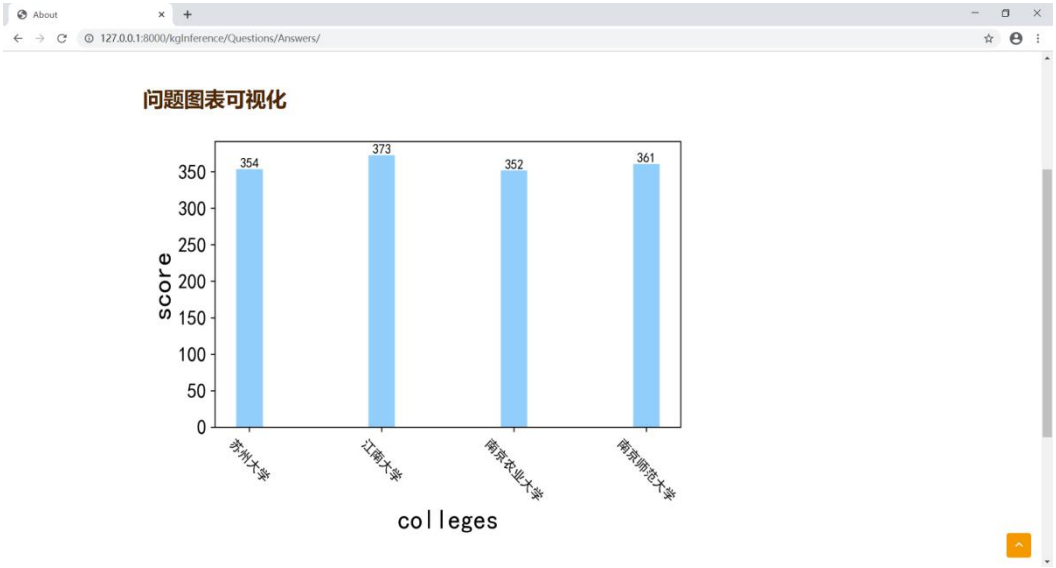
可见，同样输入，相比上一个问题，少了“南京师范大学”、“南京财经大学”，多了“南通大学”。

选择“江苏”、“理科”、“360 分”，倾向省份选择“江苏”，选择问题“江苏的大学在本省理科的录取分数线排名”：

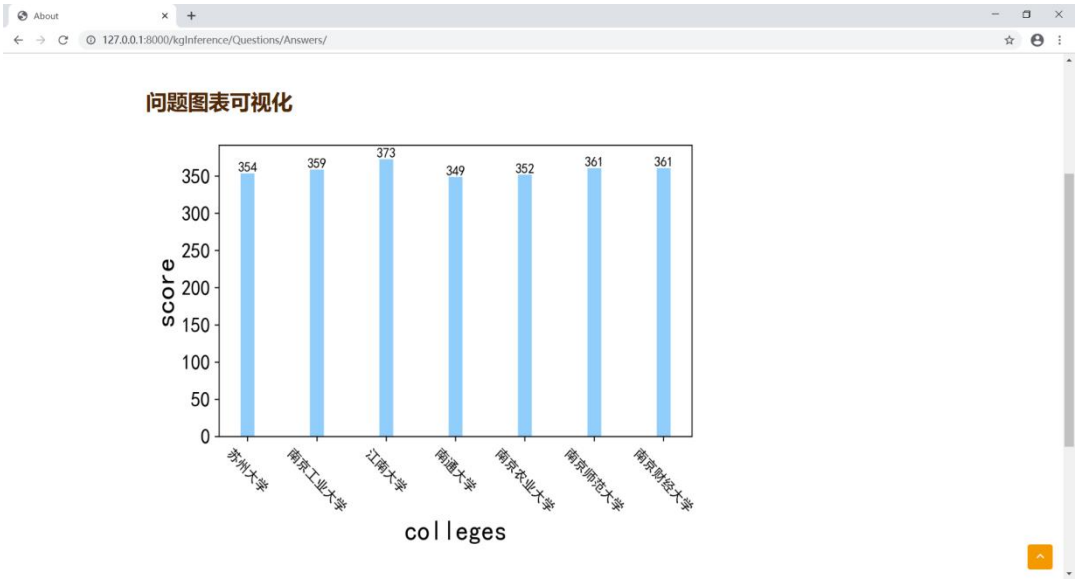


由于数据库中数据不全，部分学校的分数无法查到。

选择“江苏”、“理科”、“360 分”，倾向省份选择“江苏”，选择问题“江苏理科 360 分能上江苏的什么 211 学校？”：



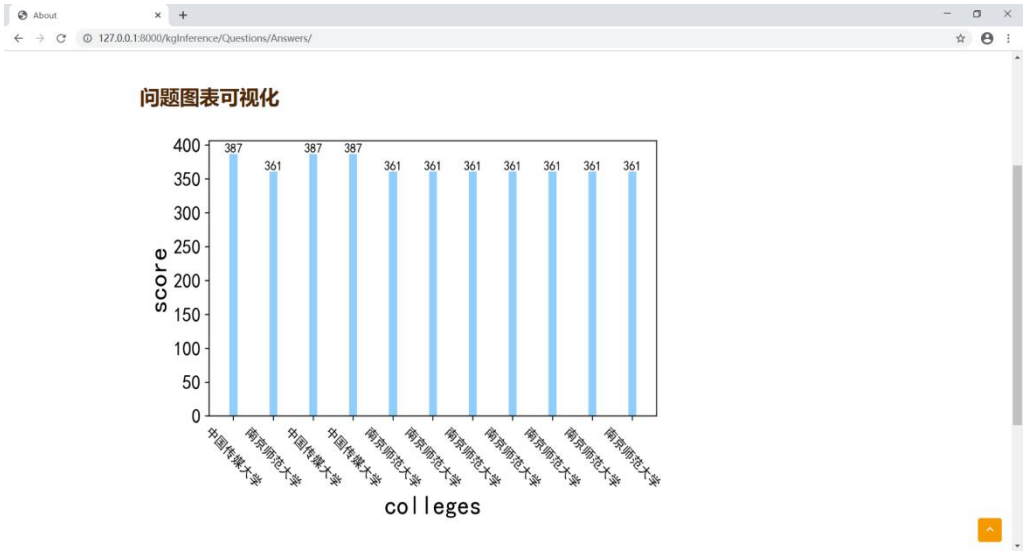
选择“江苏”、“理科”、“360 分”，倾向省份选择“江苏”，选择问题“江苏理科 360 分能上江苏的什么本科学校？”：



可见结果图中既包含上一张图中的 211 学校，也包含符合条件的本科非 211 学校。

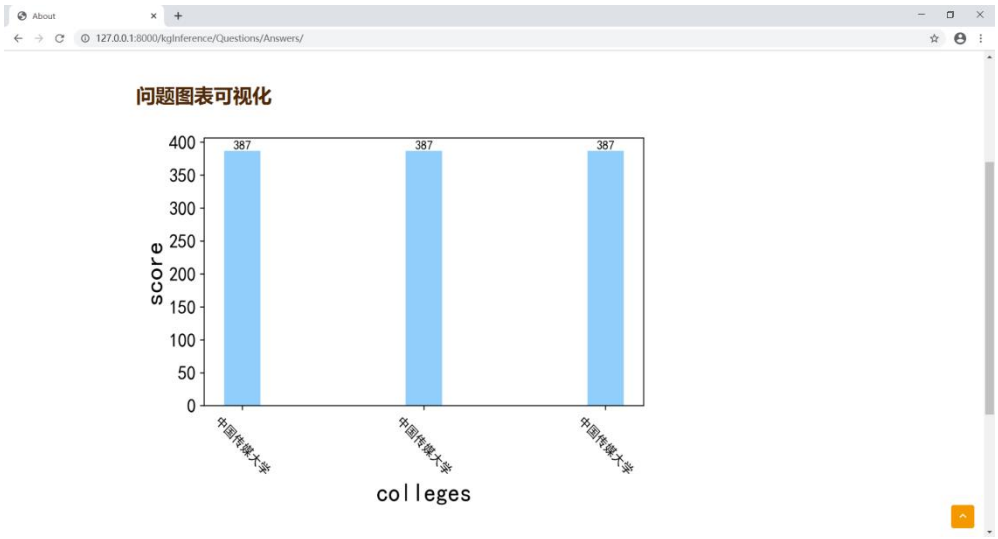
1.1.3 选填信息中选上倾向专业

选择“江苏”、“理科”、“380 分”，倾向专业选择“新闻传播学”，选择问题“新闻传播学专业全国大学在本省的录取分数线？”：



由于南京师范大学 2019 年开设多个新闻传播学（一级学科）下设的专业，所以此处有多条记录。

选择“江苏”、“理科”、“380 分”，倾向专业选择“新闻传播学”，选择问题“江苏理科 380 分能上哪些学校的新闻传播学专业？”：



选填信息中选上倾向专业而未选倾向省份时，另外三个问题模型和之前的一样，此处不再展示。

1.1.4 选填信息中倾向专业和倾向省份都选了

选择“江苏”、“理科”、“410 分”，倾向省份选择“江苏”，倾向专业选择“教育学”，选择问题“教育学专业江苏省的大学在江苏省的录取分数线？”

由于江南大学 2019 年开设多个教育学（一级学科）下设的专业，所以此处有多条记录。

选填信息中选上倾向专业同时选上倾向省份时，另外几个问题模型和之前的一样，此处不再展示。

2. 分析

2.1 根据用户输入的动态个人信息进行相应的可视化

优点：根据用户填入的个人信息生成问题，经过选择后生成对应图表，具有引导作用，引导用户选择自己想知道的信息，生成的图表比较直观、易于接受，比较人性化。使用下拉框的方式比较方便。

缺点：有时获得的符合条件的数据比较多，导致柱状图比较密，效果不理想，views 中写了根据数据条数自动生成多张图，每张图最多显示十条数据的函数，但是由于我们没有解决前端自动生成数量不等的图片时出现的 bug，所以没有实现此功能。

2.2 找到对应一级学科（消歧）：

优点：计算开销小，速度快，准确率比较高（手动修改错误数据前估计准确率 95%以上），且算法鲁棒性比较强。可以用此方法获取数据库 Majors 表中 firstlevelID 属性。在部分函数中嵌入了相关方法，相比 django 自带模糊查询效果更好，因为专业和对应一级学科之间并非都是包含关系，使用 django 模糊查询方法会丢弃很多有效数据。

缺点：无法达到接近百分之百的准确率，仍有一些语义判断的错误，后来手动修改了约 2%的结果。可以使用神经网络获得非常高的准确率，但是考虑到使用比较深的神经网络在准确率提高空间不大的同时输出结果所需的计算量大幅增加，所以没有使用神经网络。未来随着各高校相继采用大类招生手段，每个专业名可能要对应多个一级学科，而不是一一对应，所以算法还有很大改进空间。

六、实验总结与心得体会

在本次实验中，我在孔老师的指导下第一次参与了一个大型项目的开发，体验了软件开发的大致流程。由于任务定位的原因，我们组的工作相对松散，并不是整组一起完成一个大任务，而是分成各个小组分别完成不同的小任务。

我们小组的任务是完成根据用户输入的动态个人信息进行相应的可视化功能。开发过程中，我充分了解并体会了基于码云的项目开发模式，学习了 django 框架下项目开发流程和在 django 框架下后端程序的编写，虽然没有写 html，但是也学习了前端开发的基本方法和思想。运用这些知识我对我们组的项目进行了概念设计，并和张立创合作完成了根据用户输入的动态个人信息进行相应的可视化的任务的开发和测试。在实际操作过程中，我体会到了团队合作开发项目的困难，虽然我们小组基本完成了前后端分离，但是后端代码需要在前端测试，导致实际操作中效率并不高。总体来说虽然遇到了很多困难，但我们小组最终完成了最初概念设计时设定的任务，功能也都能在本地实现。

通过本次课程实践，我感受到真正的项目开发并不像我们在课堂上学一门语言或学一些算法那么简单。实际的操作，小组间以及小组成员间关于工作内容、新的想法、代码 bug 等问题的及时沟通交流非常重要。如何在一个大项目中提高个人工作效率与整体工作效率是一个值得思考的问题。本次课程时间对于我来说是一次宝贵的实践经历，为以后的学习和工作打下了基础，令我受益匪浅。