印度美食分析与相似性研究项目报告

项目报告作者：刘付雄

时间：2023年10月16日

目录

[1.项目背景 3](#_Toc148278853)

[2.项目目标 3](#_Toc148278854)

[3.项目现状 4](#_Toc148278855)

[4.解决思路及流程 4](#_Toc148278856)

[5.项目实现 5](#_Toc148278857)

[5.1库包需求 5](#_Toc148278858)

[5.2查看数据结构 5](#_Toc148278859)

[5.3数据预处理 6](#_Toc148278860)

[5.3.1缺失值检测与处理 6](#_Toc148278861)

[5.3.2重复值检测与处理 7](#_Toc148278862)

[5.3.3异常值检测与处理 7](#_Toc148278863)

[5.3.4特征处理 7](#_Toc148278864)

[5.4数据探索性分析 8](#_Toc148278865)

[5.4.1 name特征 8](#_Toc148278866)

[5.4.2 diet特征 8](#_Toc148278867)

[5.4.3 flavor\_profile特征 9](#_Toc148278868)

[5.4.4 course特征 10](#_Toc148278869)

[5.4.5 state特征 11](#_Toc148278870)

[5.4.6 region特征 12](#_Toc148278871)

[5.4.7 ingredients特征 13](#_Toc148278872)

[5.5构造新数据指标 13](#_Toc148278873)

[5.6数据分析 13](#_Toc148278874)

[5.7相似性分析 14](#_Toc148278875)

[5.8开发美食推荐系统 14](#_Toc148278876)

[5.9 分析总结 14](#_Toc148278877)

[5.10项目总结 14](#_Toc148278878)

# 1.项目背景

项目背景如下：

印度是一个多民族、多文化的国家，其美食文化很有丰富性，印度拥有众多不同的地区、民族和文化，每个地区都有自己独特的美食文化，包括各种香料、食材和烹饪方法。印度美食在全球范围内越来越受欢迎，许多人对印度美食的口味和风格感兴趣，加上如今疫情开放，经济发展，旅游业逐渐恢复生机。通过研究分析印度美食，帮助游客更好地理解不同地区的菜肴特色，为他们提供更好的服务。

# 2.项目目标

基于上述存在的数据问题，主要实现以下这些目标：

1. 构建新的数据指标：总处理时长；
2. 寻找印度菜共同点（味道、用料、类型、平均总处理时长）；
3. 分析印度菜的特色（各类型数量占比、味道、用料、总处理时长）
4. 分析印度各地区菜肴的特色；
5. 分析印度各地区菜肴的特色；
6. 分析各菜肴类型特色；
7. 分析素食和非素食的特色。
8. 为印度当地文旅局总结各地美食特色，便于推广美食并发展当地的饮食业，根据各地的美食特色，通过线上宣传推广的方式，来扩大印度美食的影响力，从而吸引更多的游客前往印度旅游；
9. 制定顾客去印度旅游的美食攻略

根据分析结果，可以让旅客从自身的饮食喜好和口味偏好、自身的时间安排、各地区菜肴的特色、各邦的菜肴特色等方面制定美食攻略（去印度的哪个区吃？吃什么？）。

# 3.项目现状

印度美食种类繁多，并且有许多相似的菜肴。通过查看数据结构可知：

1. 数据集中的数据指标共有9个，分别为：

表 1数据字段说明

表 2

|  |  |
| --- | --- |
| 字段名 | 说明 |
| name | 菜肴名称 |
| ingredients | 主要使用食材 |
| diet | 饮食类型 - 素食或非素食 |
| prep\_time | 准备时间 |
| cook\_time | 烹饪时间 |
| flavor\_profile | 风味特征，包括菜品是否辣、甜、苦等 |
| course | 菜肴类型 - 开胃菜、主菜、甜点等 |
| state | 菜肴著名或起源的州 |
| region | 国家所属地区 |

数据指标除了准备时间、烹饪时间外实际意义均为字符，有些杂乱，需要总结分析出各个指标的情况。在此基础上新增一个指标：总处理时长—准备时间与烹饪时间的总和；

1. 数据的缺失值较多，对分析过程以及目标实现有些影响；
2. 进行相似度分析时，数据预处理、选择合适的特征表示和相似度计算方法都非常重要，这将直接影响到最后的结果。因此，在实际应用中需要仔细设计和调整相似度分析的流程和参数，以提高准确性和可靠性。

# 4.解决思路及流程

流程如下图所示：

# 5.项目实现

## 5.1库包需求

|  |  |
| --- | --- |
| 语言/库名 | 版本 |
| python | 3.9.12 |
| numpy | 1.24.3 |
| pandas | 1.4.2 |
| matplotlib | 3.5.3 |
| pyecharts | 1.9.1 |
| seaborn | 0.12.2 |
| sklearn | 1.0.2 |
| tkinter | 8.6 |

## 5.2查看数据结构

查看数据结构是指在进行数据分析之前，对原始数据进行基础的探索性分析。

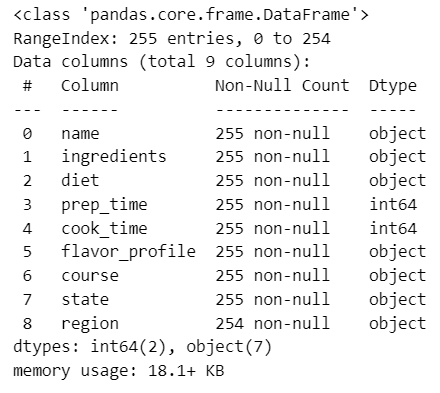
|  |
| --- |
| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  df = pd.read\_csv('/数据/indian\_food.csv')  df.head() |

输出结果如下:



|  |
| --- |
| # 查看数据情况  df.info() |

输出结果如下：



## 5.3数据预处理

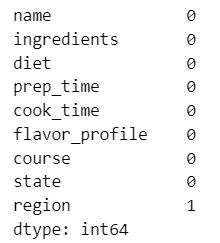
数据预处理是指在进行数据分析之前，对原始数据进行清洗、转换和集成等一系列处理，以便于进行更高效、准确的数据分析。

该项目需要做的数据预处理内容包括：检测和纠正数据中的异常值、缺失值、重复值和错误值等。

### 5.3.1缺失值检测与处理

|  |
| --- |
| #检测缺失值:Null。region有一个null值  df.isnull().sum() |

输入结果如下：



检测结果：region列含有一个null值，根据该列的实际含义，我们将其转换为其它地区：Other region。代码如下：

|  |
| --- |
| df.fillna('Other region') |

### 5.3.2重复值检测与处理

|  |
| --- |
| # 检测重复值  df.duplicated().sum() |

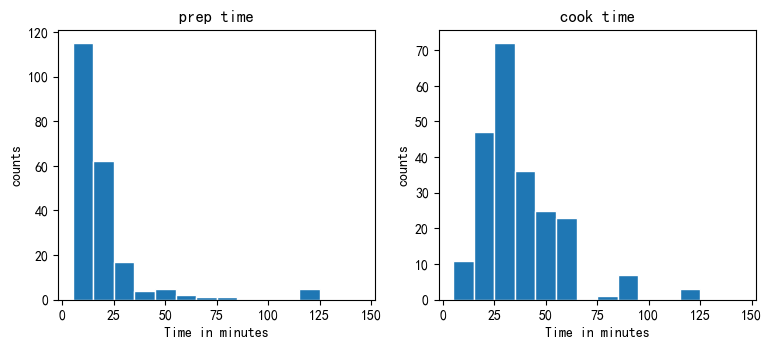
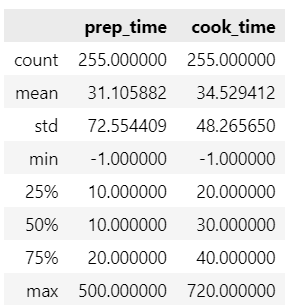
输出结果为：0。没有重复值

### 5.3.3异常值检测与处理

这里只针对连续型变量作异常值的检测，离散型变量需在数据探索性分析中检测。

|  |
| --- |
| plt.figure(figsize=(9,3.5))  plt.subplot(121)  plt.hist(df['prep\_time'],np.arange(5,150,10),edgecolor='w')  plt.title("prep time")  plt.ylabel("counts")  plt.xlabel("Time in minutes")  plt.subplot(122)  plt.hist(df['cook\_time'],np.arange(5,150,10),edgecolor='w')  plt.title("cook time")  plt.ylabel("counts")  plt.xlabel("Time in minutes")  plt.show() |

输出结果如下：

准备时间和烹饪时间最小值是-1，其应该视作异常值，先将-1值转为nan值。

### 5.3.4特征处理

针对ingredients 列，其值是包含多个食材的英文名称，其原始值存在一些问题，会影响我们接下来的数据分析。当我们将所有食材放在一个列表，并且去重时，有425种，但在列表中观察每个食材时，会发现许多食材名称前或后会有空格、同一食材名称大小写不一致，这就导致了统计食材数量产生了偏差。我们解决掉这些问题，并且针对含有多个单词的食材，将中间空格转换为“\_”，以便后续数据分析。（说明：这些处理我们并不在原特征上做变化，也不构造新的特征，而是在需要分析食材时才做处理。）处理代码如下：

|  |
| --- |
| ingredientsList=[]  for i in range(0,len(df)):      ingredientsList.append(df['ingredients'][i].split(","))  flat\_list = []  for sublist in ingredientsList:      for item in sublist:          flat\_list.append(item)  countIngredients=[]  ingred=[]  for i in flat\_list:      str1 = i.lstrip()      str1 = str1.lower()      str2 = str1.rstrip()      ingred.append(str2)  ingred1 = [item.replace(' ', '\_') for item in ingred] |

输出结果如下：

|  |
| --- |
| ['maida\_flour','yogurt','oil','sugar',……'maida','mustard\_oil'] |

## 5.4数据探索性分析

进行探索性数据分析，目的是发现数据中的模式、趋势、异常值和关联关系，以帮助理解数据的特征和可用性。

### 5.4.1 name特征

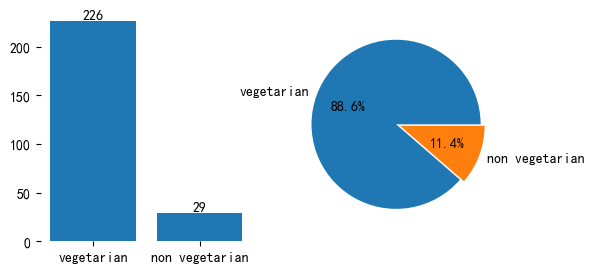
|  |
| --- |
| #共255种美食  len(set(df['name'])) |

输出结果：255，共255种印度美食。

### 5.4.2 diet特征

|  |
| --- |
| df\_diet = df['diet'].value\_counts().reset\_index()  plt.figure(figsize=(6,3))  plt.title("饮食类型分布")  plt.subplot(121)  p1=plt.bar(df\_diet["index"],df\_diet["diet"])  ax=plt.gca()  ax.spines['top'].set\_color('none')#顶部没了  ax.spines['right'].set\_color('none')#右边没了  ax.spines['left'].set\_color('none')  ax.spines['bottom'].set\_color('none')  plt.bar\_label(p1,label\_type='edge')  plt.subplot(122)  plt.pie(df\_diet["diet"],explode=(0.05,0), labels=df\_diet["index"],autopct='%.1f%%',)#显示百分比  plt.show() |

输出结果如下：

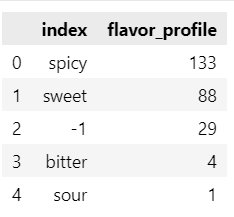


由图可知，255种美食中，有226种素食美食，29种非素食美食，素食美食占88.6%，非素食美食占11.4%。

### 5.4.3 flavor\_profile特征

|  |
| --- |
| df\_flavor = df['flavor\_profile'].value\_counts().reset\_index()  df\_flavor |

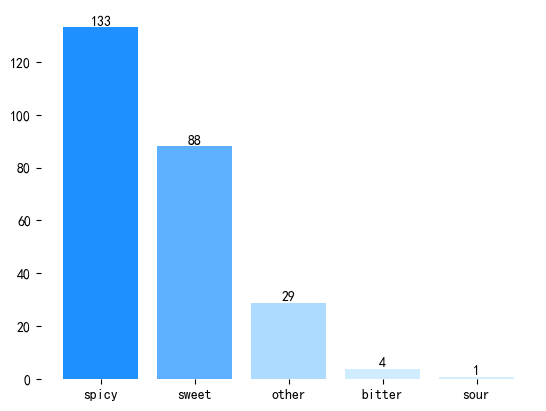
输出结果如下：



可以看到，风味特征有一个“-1”值，该值视为异常值,根据该列的实际含义,我们将其转换为其他风味:other。再分析各个风味。代码如下：

|  |
| --- |
| df['flavor\_profile'] = df['flavor\_profile'].replace('-1','other')  df\_flavor = df['flavor\_profile'].value\_counts().reset\_index()  from matplotlib import cm  from matplotlib.colors import ListedColormap, LinearSegmentedColormap  def bar\_color(data):      blue\_colors = ['#D4EEFF', '#1E90FF']  # 淡蓝色到深蓝色      cmap = LinearSegmentedColormap.from\_list('my\_cmap', blue\_colors)      norm = plt.Normalize(data.min(), data.max())      norm\_y = norm(data)      color = cmap(norm\_y)  #      return color  p1=plt.bar(df\_flavor["index"],df\_flavor["flavor\_profile"],color=bar\_color(df\_flavor["flavor\_profile"]))  ax=plt.gca()  ax.spines['top'].set\_color('none')#顶部没了  ax.spines['right'].set\_color('none')#右边没了  ax.spines['left'].set\_color('none')  ax.spines['bottom'].set\_color('none')  plt.bar\_label(p1,label\_type='edge')  plt.show() |

输出结果如下：

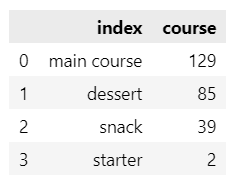


风味为spicy的美食最多，其次是sweet，sour的美食最少。

### 5.4.4 course特征

|  |
| --- |
| df\_course = df['course'].value\_counts().reset\_index()  df\_course |

输出结果如下：

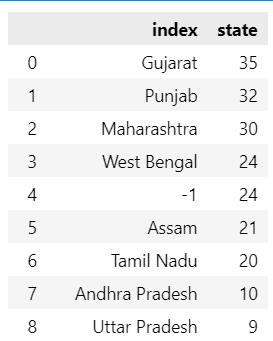


main course类型的美食最多，starter类型的美食最少。

### 5.4.5 state特征

|  |
| --- |
| df\_state = df['state'].value\_counts().reset\_index()  df\_state |

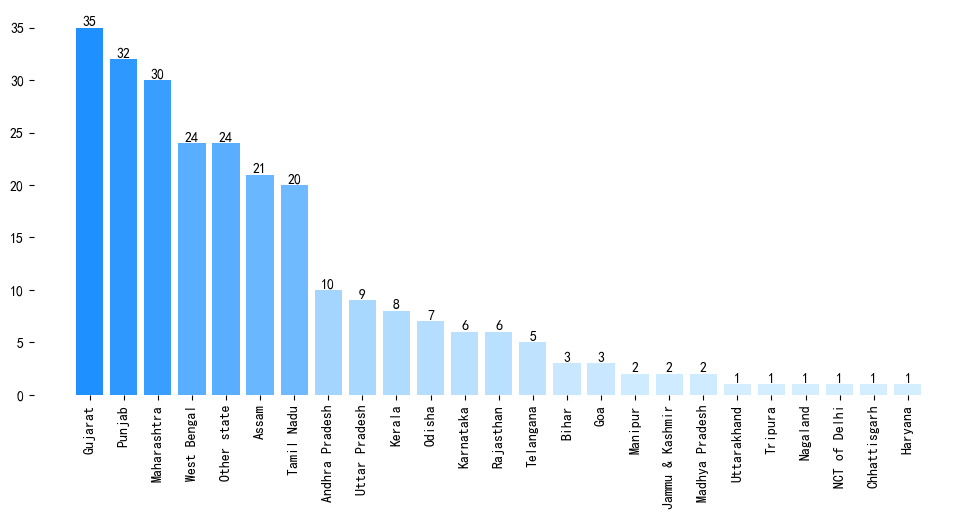
输出部分结果如下：



可以看到,州名中有“-1”值,将其视为异常值,根据该特征实际含义,将其填补为其他州:Other state。并分析各州美食数量。代码如下：

|  |
| --- |
| df['state'] = df['state'].replace('-1','Other state')  df\_state =df['state'].value\_counts().reset\_index()  plt.figure(figsize=(12,5))  p1=plt.bar(df\_state["index"],df\_state["state"],color=bar\_color(df\_state["state"]))  ax=plt.gca()  ax.spines['top'].set\_color('none')#顶部没了  ax.spines['right'].set\_color('none')#右边没了  ax.spines['left'].set\_color('none')  ax.spines['bottom'].set\_color('none')  plt.bar\_label(p1,label\_type='edge')  plt.xticks(rotation=90)  plt.show() |

输出结果如下：

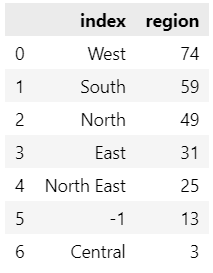


Gujarat州的美食最多，其次是Punjab州，而Uttarakhand、Tripura、Nagaland、NCT of Delhi、Chhattisgarh、Haryana州的美食都只有一种。

### 5.4.6 region特征

|  |
| --- |
| df\_region = df['region'].value\_counts().reset\_index()  df\_region |

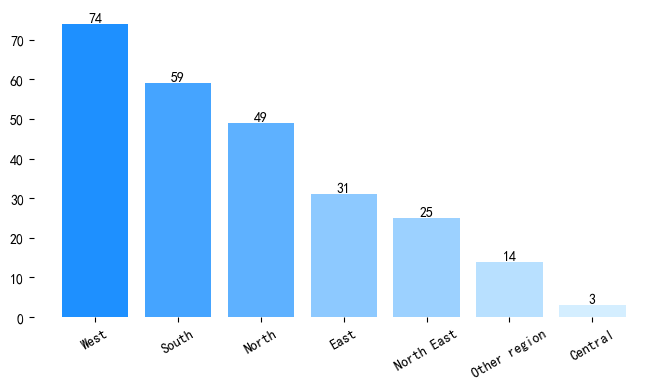
输出结果如下：



同样的,将-1值填补为其它地区:Other region。并分析各地区的美食数量，代码如下：

|  |
| --- |
| df['region'] =  df['region'].replace('-1','Other region')  df\_region = df['region'].value\_counts().reset\_index()  plt.figure(figsize=(8,4))  p1=plt.bar(df\_region["index"],df\_region["region"],color=bar\_color(df\_region["region"]))  ax=plt.gca()  ax.spines['top'].set\_color('none')#顶部没了  ax.spines['right'].set\_color('none')#右边没了  ax.spines['left'].set\_color('none')  ax.spines['bottom'].set\_color('none')  plt.bar\_label(p1,label\_type='edge')  plt.xticks(rotation=30)  plt.show() |

输出结果如下：



西部地区的美食最多，有74种，总部地区美食最少，只有3种。

### 5.4.7 ingredients特征

|  |
| --- |
| ingredients = set()  for item in df['ingredients']:      ingredients.update(str(item).lower().split(","))  ingred=[]  for i in ingredients:      str1 = i.lstrip()      str2 = str1.rstrip()      # print(str2)      ingred.append(str2)  #总共365种食材  len(set(ingred)) |

输出结果：365，255种美食共用了365种食材。

## 5.5构造新数据指标

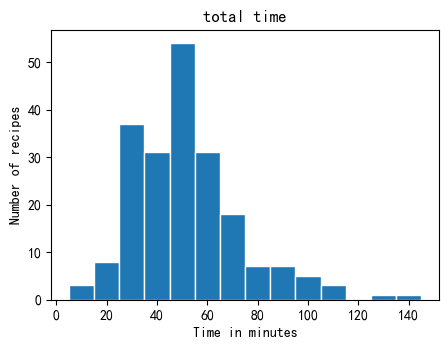
针对美食的准备时间和烹饪时间，我们添加一列，为美食所花费总时间：total\_time。代码如下：

|  |
| --- |
| df['total\_time'] = df['prep\_time']+df['cook\_time'] |

总花费时间分布：

|  |
| --- |
| plt.figure(figsize=(5,3.5))  plt.hist(df['total\_time'],np.arange(5,150,10),edgecolor='w')  plt.title("total time")  plt.ylabel("Number of recipes")  plt.xlabel("Time in minutes")  plt.show() |

输出结果如下：

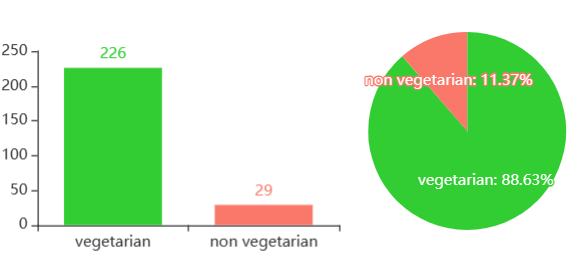


## 5.6数据分析

### 5.6.1分析素食和非素食的特色

|  |
| --- |
| def barPage() -> Bar:      x\_data = df\_diet['index'].to\_list()      bar1 = (          Bar()          .add\_xaxis(x\_data)          .add\_yaxis(              "",          df\_diet['diet'].to\_list(),              category\_gap="35%",   #柱子间隔              label\_opts=opts.LabelOpts(is\_show=True),          )      )      bar1.set\_series\_opts(label\_opts=opts.LabelOpts(position="top"))      bar1.set\_global\_opts(          title\_opts=opts.TitleOpts(title="",pos\_left='16%',title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)),          visualmap\_opts=opts.VisualMapOpts(              type\_="color",              min\_=min(df\_diet['diet'].to\_list()),              max\_=max(df\_diet['diet'].to\_list()),              range\_color = ['#FA786A','#32CD32'],              is\_show=False,          )      )      grid = Grid()      grid.add(bar1,grid\_opts=opts.GridOpts(width='40%', height='40%',pos\_left="5%", pos\_right="20%"), is\_control\_axis\_index=True)#width调整粗细      return grid  def piePage() -> Pie:      data\_pair = [(row['index'], row['diet']) for \_, row in df\_diet.iterrows()]      # data\_pair = sorted(data\_pair, key=lambda x: x[1], reverse=True)      colors=['#32CD32', '#FA786A']      pie2 = (Pie()          .add('', data\_pair,              radius=["0%", "60%"],              )              .set\_colors(colors)  # 设置颜色          .set\_global\_opts(title\_opts=opts.TitleOpts(title="",                                                      pos\_left = "43%",pos\_top = "5%"),                              legend\_opts=opts.LegendOpts(is\_show=False),)          .set\_series\_opts(label\_opts=opts.LabelOpts(formatter="{b}: {d}%",position='inside')) # 显示在图形的标签形式          )      return pie2  page = (      Page(layout=Page.DraggablePageLayout)      .add(          barPage(),          piePage(),      )  )  page.render('素食和非素食美食种数组合图.html')  Page.save\_resize\_html("素食和非素食美食种数组合图.html",   # 上面的HTML文件名称                        cfg\_file="chart\_config\_2.json",  # 保存的json配置文件                        dest="new\_素食和非素食美食种数组合图.html")  # 新HTML文件名称 |

输出结果如下：



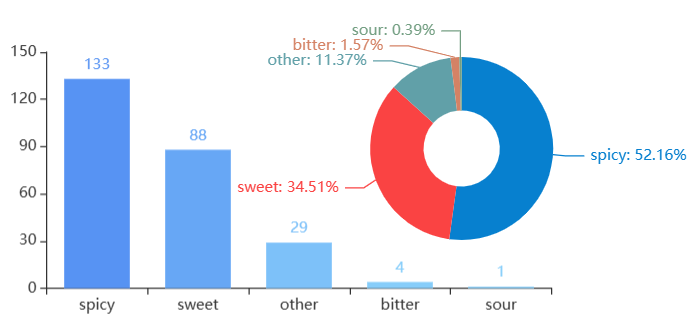
我们可以看出印度美食中素食菜品占据了绝大多数，有226种，占88.63%，而非素食菜品仅有29种，占11.37%。这显示出了印度素食主义文化的深厚影响。

素食是印度文化的一个重要组成部分，因为根据印度教、耆那教和锡克教的教义，禁止食用肉类。这种文化传统不仅影响了印度本土饮食文化，也影响到了全球的素食和素食主义文化。

### 5.6.2分析各菜肴风味特色

|  |
| --- |
| df\_flavor = df['flavor\_profile'].value\_counts().reset\_index()  def barPage() -> Bar:      x\_data = df\_flavor['index'].to\_list()      bar1 = (          Bar()          .add\_xaxis(x\_data)          .add\_yaxis(              "",          df\_flavor['flavor\_profile'].to\_list(),              category\_gap="35%",   #柱子间隔              label\_opts=opts.LabelOpts(is\_show=True),          )      )      bar1.set\_series\_opts(label\_opts=opts.LabelOpts(position="top"))      bar1.set\_global\_opts(          title\_opts=opts.TitleOpts(title="",pos\_left='16%',title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)),          visualmap\_opts=opts.VisualMapOpts(              type\_="color",              min\_=min(df\_flavor['flavor\_profile'].to\_list()),              max\_=max(df\_flavor['flavor\_profile'].to\_list()),              range\_color = ['#87CEFA','#5793f3'],              is\_show=False,          )      )      grid = Grid()      grid.add(bar1,grid\_opts=opts.GridOpts(width='50%', height='45%',pos\_left="5%", pos\_right="20%"), is\_control\_axis\_index=True)#width调整粗细      return grid  def piePage() -> Pie:      data\_pair = [(row['index'], row['flavor\_profile']) for \_, row in df\_flavor.iterrows()]      data\_pair      colors=['#0780cf', '#fa4343','#61a0a8', '#d48265', '#749f83']      pie2 = (Pie()          .add('', data\_pair,              radius=["25%", "60%"],              )              .set\_colors(colors)  # 设置颜色          .set\_global\_opts(title\_opts=opts.TitleOpts(title="",                                                      pos\_left = "43%",pos\_top = "5%"),                              legend\_opts=opts.LegendOpts(is\_show=False),)          .set\_series\_opts(label\_opts=opts.LabelOpts(formatter="{b}: {d}%",position='outside')) # 显示在图形的标签形式          )      return pie2  page = (      Page(layout=Page.DraggablePageLayout)      .add(          barPage(),          piePage(),      )  )  page.render('各种风味的美食种数组合图.html')  Page.save\_resize\_html("各种风味的美食种数组合图.html",   # 上面的HTML文件名称                        cfg\_file="chart\_config\_3.json",  # 保存的json配置文件                        dest="new\_各种风味的美食种数组合图.html")  # 新HTML文件名称 |

输出结果如下：

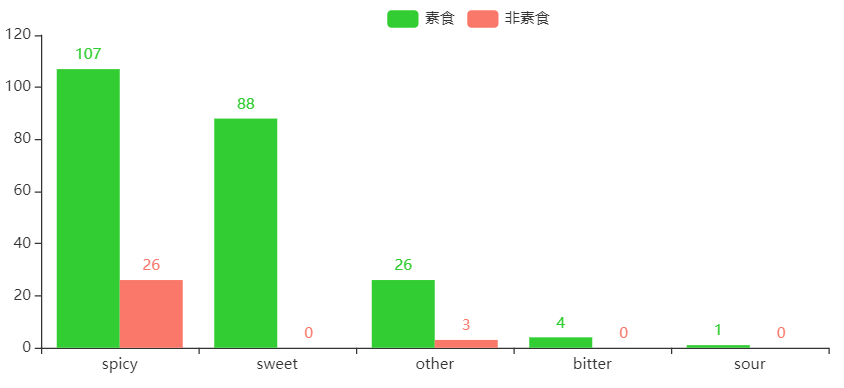


根据可视化图像，有133种美食属于辛辣口味，88种美食属于甜味口味，29种美食属于其它口味，4种美食属于苦味口味，只有1种美食属于酸味口味，。这反映了印度菜肴普遍具有浓郁的香料和辛辣调味的特点。辣味在印度饮食中被广泛使用，为菜品带来独特的风味和火辣的口感；不仅如此，印度还有着丰富的甜点文化，这些甜点通常包含各种奶制品、果干、坚果和糖浆等成分；而对于其它风味的美食，这可能是指一些特殊口味或者结合多种味道的菜品，如酸辣、咖喱、咖啡等。这些菜品可能具有独特的风味组合。

进一步分析各风味素食和非素食美食种数：

|  |
| --- |
| df\_flavor\_diet = pd.pivot\_table(data=df,index='flavor\_profile',columns='diet',values='name',aggfunc='count')  df\_flavor\_diet.fillna(0,inplace=True)  df\_flavor\_diet.sort\_values('vegetarian',ascending=False,inplace=True)  # 绘制分组柱形图  x\_data = df\_flavor\_diet.index.to\_list()  bar = Bar()  bar.add\_xaxis(x\_data)  bar.add\_yaxis('素食', df\_flavor\_diet['vegetarian'].tolist(),gap="0%",color='#FA786A')  bar.add\_yaxis('非素食',df\_flavor\_diet['non vegetarian'].tolist(),gap="0%",color='#32CD32')  # bar.set\_series\_opts(label\_opts={'position': 'inside'})  bar.set\_global\_opts(      title\_opts=opts.TitleOpts(title="",                                pos\_left='30%',pos\_top="1%",                                title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)                                ),      xaxis\_opts=opts.AxisOpts(name\_rotate=60,axislabel\_opts={"rotate":0},is\_show=True),  )  bar.set\_series\_opts(      label\_opts=opts.LabelOpts(          position="top",          formatter=JsCode(              "function(params){return params.value.toFixed(0);}"          ),      ),  )  grid = Grid()  grid.add(bar,grid\_opts=opts.GridOpts(width='70%', height='50%',pos\_top="5%",pos\_left="12%", pos\_right="20%"), is\_control\_axis\_index=True)#width调整粗细  grid.render\_notebook() |

输出结果如下：

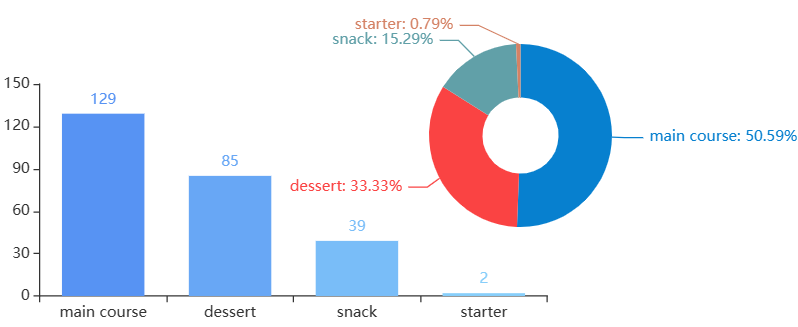


可以看到，无论是素食还是非素食，风味最多的都是辛辣口味。素食有许多甜味口味的美食，而非素食口味却没有。

### 5.6.3分析各菜肴类型特色

|  |
| --- |
| df\_course = df['course'].value\_counts().reset\_index()  def barPage() -> Bar:      x\_data = df\_course['index'].to\_list()      bar1 = (          Bar()          .add\_xaxis(x\_data)          .add\_yaxis(              "",          df\_course['course'].to\_list(),              category\_gap="35%",   #柱子间隔              label\_opts=opts.LabelOpts(is\_show=True),          )      )      bar1.set\_series\_opts(label\_opts=opts.LabelOpts(position="top"))      bar1.set\_global\_opts(          title\_opts=opts.TitleOpts(title="",pos\_left='16%',title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)),          visualmap\_opts=opts.VisualMapOpts(              type\_="color",              min\_=min(df\_course['course'].to\_list()),              max\_=max(df\_course['course'].to\_list()),              range\_color = ['#87CEFA','#5793f3'],              # is\_piecewise=True,              is\_show=False,          )      )      grid = Grid()      grid.add(bar1,grid\_opts=opts.GridOpts(width='50%', height='40%',pos\_left="5%", pos\_right="20%"), is\_control\_axis\_index=True)#width调整粗细      return grid  def piePage() -> Pie:      data\_pair = [(row['index'], row['course']) for \_, row in df\_course.iterrows()]      colors=['#0780cf', '#fa4343','#61a0a8', '#d48265', '#749f83']      pie2 = (Pie()          .add('', data\_pair,              radius=["25%", "60%"],              )              .set\_colors(colors)  # 设置颜色          .set\_global\_opts(title\_opts=opts.TitleOpts(title="",                                                      pos\_left = "43%",pos\_top = "5%"),                              legend\_opts=opts.LegendOpts(is\_show=False),)          .set\_series\_opts(label\_opts=opts.LabelOpts(formatter="{b}: {d}%",position='outside')) # 显示在图形的标签形式          )      return pie2  page = (      Page(layout=Page.DraggablePageLayout)      .add(          barPage(),          piePage(),      )  )  page.render('各类型美食种数组合图.html')  Page.save\_resize\_html("各类型美食种数组合图.html",   # 上面的HTML文件名称                        cfg\_file="chart\_config\_4.json",  # 保存的json配置文件                        dest="new\_各类型美食种数组合图.html")  # 新HTML文件名称 |

输出结果如下：

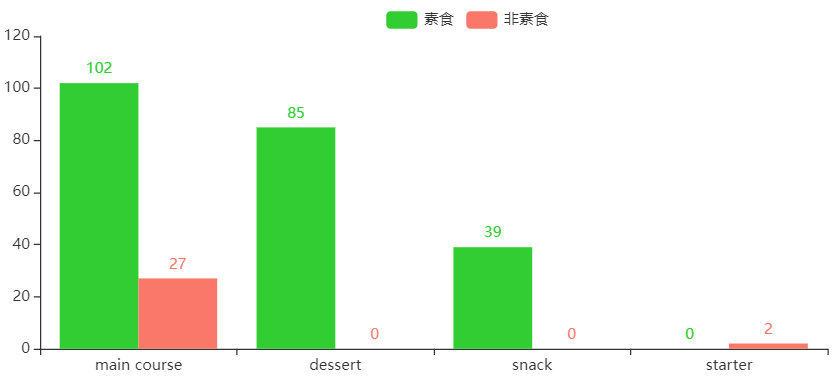


由图中可以看出，255种印度美食中，主食的占比最高50.59%，有129种，其次是甜点，占比33.33%，有85种，小吃占比15.29%，有39种，占比最小的是开胃菜0.79%，只有2种。可以知道，主菜通常是印度餐中的主要部分，提供了大量的蛋白质、碳水化合物和蔬菜；印度的甜点种类繁多，这85种甜点可能包括了各种饼干、蛋糕、布丁、凝胶、奶昔和传统的印度甜点，如古拉和拉斯马利。这些甜点通常会使用大量的糖、牛奶和香料，口感丰富；对于小吃，通常是较轻的食物，可以在任何时间食用。可能包括炸食、烤食、沙拉和各种小点心；开胃菜通常是在正餐开始前食用的，以激发食欲，所以种类不太多。

我们进一步分析各菜肴类型素食和非素食美食数量。

|  |
| --- |
| df\_course\_diet = pd.pivot\_table(data=df,index='course',columns='diet',values='name',aggfunc='count')  df\_course\_diet.fillna(0,inplace=True)  df\_course\_diet.sort\_values('vegetarian',ascending=False,inplace=True)  # 绘制分组柱形图  x\_data = df\_course\_diet.index.to\_list()  bar = Bar()  bar.add\_xaxis(x\_data)  bar.add\_yaxis('素食', df\_course\_diet['vegetarian'].tolist(),gap="0%",color='#FA786A')  bar.add\_yaxis('非素食',df\_course\_diet['non vegetarian'].tolist(),gap="0%",color='#32CD32')  # bar.set\_series\_opts(label\_opts={'position': 'inside'})  bar.set\_global\_opts(      title\_opts=opts.TitleOpts(title="",                                pos\_left='30%',pos\_top="1%",                                title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)                                ),      xaxis\_opts=opts.AxisOpts(name\_rotate=60,axislabel\_opts={"rotate":0},is\_show=True),  )  bar.set\_series\_opts(      label\_opts=opts.LabelOpts(          position="top",          formatter=JsCode(              "function(params){return params.value.toFixed(0);}"          ),      ),  )  grid = Grid()  grid.add(bar,grid\_opts=opts.GridOpts(width='70%', height='50%',pos\_top="5%",pos\_left="12%", pos\_right="20%"), is\_control\_axis\_index=True)#width调整粗细  grid.render\_notebook() |

输出结果如下：

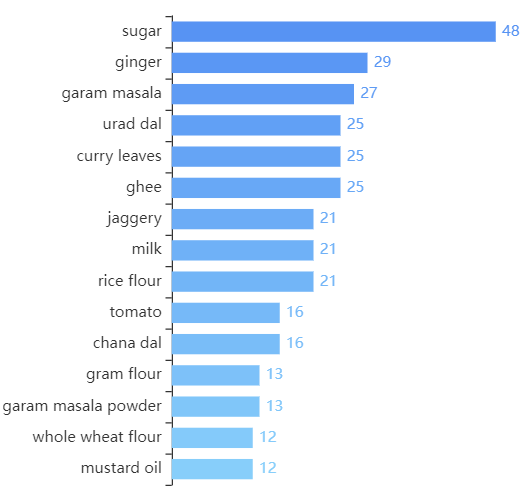


可以看出，无论是素食还是非素食，类型最多的都是主食，素食有85种美食是甜点，而非素食却没有这种类型的美食，说明印度非素食美食是不吃甜的。

### 5.6.5分析印度美食食材使用

|  |
| --- |
| ingredientsList=[]  for i in range(0,len(df)):      ingredientsList.append(df['ingredients'][i].split(","))  flat\_list = []  for sublist in ingredientsList:      for item in sublist:          flat\_list.append(item)  countIngredients=[]  ingred=[]  for i in flat\_list:      str1 = i.lstrip()      str1 = str1.lower()      str2 = str1.rstrip()      ingred.append(str2)  for i in ingred:     countIngredients.append(ingred.count(i))  flat\_small=map(lambda x:x.lower(), ingred)  Ingredients\_df = pd.DataFrame(list(zip(flat\_small, countIngredients)),                 columns =['Ingredient', 'Occurrence'])  sorted=Ingredients\_df.sort\_values(by=['Occurrence'], ascending=False)  unique\_ingredients=sorted.drop\_duplicates()  unique\_ingredients  ingred1 = [item.replace(' ', '\_') for item in ingred]  import stylecloud  from IPython.display import Image  stylecloud.gen\_stylecloud(text=" ".join(ingred1), collocations=False,                            font\_path='C://Windows//Fonts//simhei.ttf',##########设置字体,否则中文不显示                            icon\_name='fas fa-cloud',####图标形状                            size=1000,                            output\_name='./食材词云.png')  Image(filename='./食材词云.png')  ingredients\_top15 = unique\_ingredients.head(15)  ingredients\_top15.sort\_values('Occurrence',inplace=True)  x\_data = ingredients\_top15['Ingredient'].to\_list()  bar1 = (      Bar()      .add\_xaxis(x\_data)      .add\_yaxis(          "",          ingredients\_top15['Occurrence'].to\_list(),          category\_gap="35%",   #柱子间隔          label\_opts=opts.LabelOpts(is\_show=True),      )      .reversal\_axis()  )  bar1.set\_series\_opts(label\_opts=opts.LabelOpts(position="right"))  bar1.set\_global\_opts(      title\_opts=opts.TitleOpts(title="",pos\_left='16%',title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)),      xaxis\_opts=opts.AxisOpts(name\_rotate=60,axislabel\_opts={"rotate":45},is\_show=False),      visualmap\_opts=opts.VisualMapOpts(          type\_="color",          min\_=0,          max\_=14,          range\_color = ['#87CEFA','#5793f3'],          is\_show=False,      )  )  grid = Grid()  grid.add(bar1,grid\_opts=opts.GridOpts(width='30%', height='75%',pos\_left="15%", pos\_right="20%"), is\_control\_axis\_index=True)#width调整粗细  grid.render\_notebook() |

输出结果如下：



从词云图和条形图中可以看出，使用量较大的的配料是sugar，ginger，garam masala，ghee等。糖的使用量最大，48种美食都使用了糖，这可能是因为甜味在印度美食中扮演着重要角色。印度有多种糖类，如jaggery（一种未精炼的棕糖），它们在许多传统食谱中都有使用；姜的使用量也很大，这可能是因为姜不仅为食物增添了风味，还有一定的药用价值。在印度料理中，姜常被用来制作腌料、酱汁和咖喱；加拉姆香料是一种常见的印度混合香料，通常包括丁香、肉桂、孜然、豆蔻等香料。它为印度菜肴提供了复杂的香气和味道；与此同时，我们还发现有各种混合的面粉使用的也比较多，印度的很多美食都含有或多或少的各种面粉。

我们进一步分析素食和非素食美食的食材使用量：

统计食材总数、素食食材总数以及非素食食材总数：

|  |
| --- |
| df\_hun = df[df['diet']=='non vegetarian']  df\_su = df[df['diet']=='vegetarian']  df\_su.reset\_index(inplace=True)  df\_hun.reset\_index(inplace=True)  ingredients = set()  for item in df['ingredients']:      ingredients.update(str(item).lower().split(","))  ingred=[]  for i in ingredients:      str1 = i.lstrip()      str2 = str1.rstrip()      ingred.append(str2)  len(set(ingred))  # 素食食材有多少？  ingredients = set()  for item in df\_su['ingredients']:      ingredients.update(str(item).lower().split(","))  ingred=[]  for i in ingredients:      str1 = i.lstrip()      str2 = str1.rstrip()      ingred.append(str2)  len(set(ingred))  # 非素食食材有多少？  ingredients = set()  for item in df\_hun['ingredients']:      ingredients.update(str(item).lower().split(","))  ingred=[]  for i in ingredients:      str1 = i.lstrip()      str2 = str1.rstrip()      ingred.append(str2)  len(set(ingred)) |

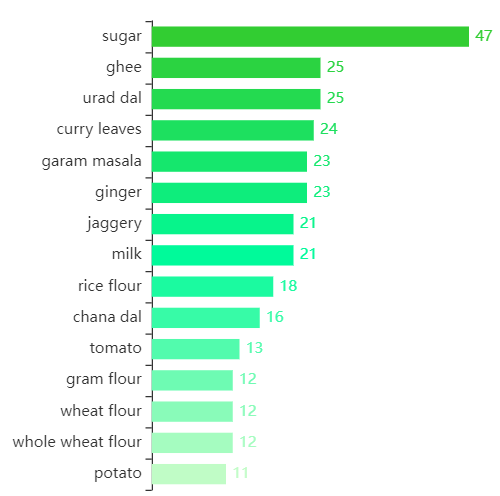
输出结果：共365种食材，素食使用了321种食材，非素食使用了96种食材。这也是由于素食美食较多，而非素食美食较少而导致两者食材差距过大。

素食和非素食食材词云图和食材使用量条形图：

|  |
| --- |
| df\_hun = df[df['diet']=='non vegetarian']  df\_su = df[df['diet']=='vegetarian']  df\_su.reset\_index(inplace=True)  df\_hun.reset\_index(inplace=True)  ingredientsList=[]  for i in range(0,len(df\_su)):      ingredientsList.append(df\_su['ingredients'][i].split(","))  flat\_list = []  for sublist in ingredientsList:      for item in sublist:          flat\_list.append(item)  countIngredients=[]  ingred=[]  for i in flat\_list:      str1 = i.lstrip()      str1 = str1.lower()      str2 = str1.rstrip()      ingred.append(str2)  for i in ingred:     countIngredients.append(ingred.count(i))  flat\_small=map(lambda x:x.lower(), ingred)  Ingredients\_df = pd.DataFrame(list(zip(flat\_small, countIngredients)),                 columns =['Ingredient', 'Occurrence'])  sorted=Ingredients\_df.sort\_values(by=['Occurrence'], ascending=False)  unique\_ingredients=sorted.drop\_duplicates()  unique\_ingredients  ingred1 = [item.replace(' ', '\_') for item in ingred]  import stylecloud  from IPython.display import Image  stylecloud.gen\_stylecloud(text=" ".join(ingred1), collocations=False,                            font\_path='C://Windows//Fonts//simhei.ttf',##########设置字体,否则中文不显示                            icon\_name='fas fa-cloud',####图标形状                            size=1000,                            output\_name='./素食食材词云1.png')  Image(filename='./素食食材词云1.png')  ingredients\_top15 = unique\_ingredients.head(15)  ingredients\_top15.sort\_values('Occurrence',inplace=True)  x\_data = ingredients\_top15['Ingredient'].to\_list()  bar1 = (      Bar()      .add\_xaxis(x\_data)      .add\_yaxis(          "",      ingredients\_top15['Occurrence'].to\_list(),          category\_gap="35%",   #柱子间隔          label\_opts=opts.LabelOpts(font\_weight=10,is\_show=True),      )      .reversal\_axis()  )  bar1.set\_series\_opts(label\_opts=opts.LabelOpts(position="right"))  bar1.set\_global\_opts(      title\_opts=opts.TitleOpts(title="素食食材排行",pos\_left='16%',title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)),      xaxis\_opts=opts.AxisOpts(name\_rotate=60,axislabel\_opts={"rotate":45},is\_show=False),      # yaxis\_opts=opts.AxisOpts(is\_show=False),      visualmap\_opts=opts.VisualMapOpts(          type\_="color",          # min\_=min(ingredients\_top15['Occurrence'].to\_list()),          # max\_=max(ingredients\_top15['Occurrence'].to\_list()),          min\_=0,          max\_=14,          range\_color = ['#C0FCC6','#00FA9A','#32CD32'],          # is\_piecewise=True,          is\_show=False,      )  )  grid = Grid()  grid.add(bar1,grid\_opts=opts.GridOpts(width='30%', height='75%',pos\_left="15%", pos\_right="20%"), is\_control\_axis\_index=True)#width调整粗细  grid.render\_notebook()  ingredientsList=[]  for i in range(0,len(df\_hun)):      ingredientsList.append(df\_hun['ingredients'][i].split(","))  flat\_list = []  for sublist in ingredientsList:      for item in sublist:          flat\_list.append(item)  countIngredients=[]  ingred=[]  for i in flat\_list:      str1 = i.lstrip()      str1 = str1.lower()      str2 = str1.rstrip()      ingred.append(str2)  for i in ingred:     countIngredients.append(ingred.count(i))  flat\_small=map(lambda x:x.lower(), ingred)  Ingredients\_df = pd.DataFrame(list(zip(flat\_small, countIngredients)),                 columns =['Ingredient', 'Occurrence'])  sorted=Ingredients\_df.sort\_values(by=['Occurrence'], ascending=False)  unique\_ingredients=sorted.drop\_duplicates()  unique\_ingredients  import stylecloud  from IPython.display import Image  stylecloud.gen\_stylecloud(text=" ".join(ingred1), collocations=False,                            font\_path='C://Windows//Fonts//simhei.ttf',##########设置字体,否则中文不显示                            icon\_name='fas fa-cloud',####图标形状                            size=800,                            output\_name='./非素食食材词云.png')  Image(filename='./非素食食材词云.png')  ingredients\_top15 = unique\_ingredients.head(15)  ingredients\_top15.sort\_values('Occurrence',inplace=True)  x\_data = ingredients\_top15['Ingredient'].to\_list()  bar1 = (      Bar()      .add\_xaxis(x\_data)      .add\_yaxis(          "",      ingredients\_top15['Occurrence'].to\_list(),          category\_gap="35%",   #柱子间隔          label\_opts=opts.LabelOpts(is\_show=True),      )      .reversal\_axis()  )  bar1.set\_series\_opts(label\_opts=opts.LabelOpts(position="right"))  bar1.set\_global\_opts(      title\_opts=opts.TitleOpts(title="非素食食材排行",pos\_left='16%',title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)),      xaxis\_opts=opts.AxisOpts(name\_rotate=60,axislabel\_opts={"rotate":45},is\_show=False),      # yaxis\_opts=opts.AxisOpts(is\_show=False),      visualmap\_opts=opts.VisualMapOpts(          type\_="color",          # min\_=min(ingredients\_top15['Occurrence'].to\_list()),          # max\_=max(ingredients\_top15['Occurrence'].to\_list()),          min\_=0,          max\_=14,          range\_color = ['#E5B97F','#E9967A','#FA786A'],          # is\_piecewise=True,          is\_show=False,      )  )  grid = Grid()  grid.add(bar1,grid\_opts=opts.GridOpts(width='30%', height='75%',pos\_left="15%", pos\_right="20%"), is\_control\_axis\_index=True)#width调整粗细  grid.render\_notebook() |

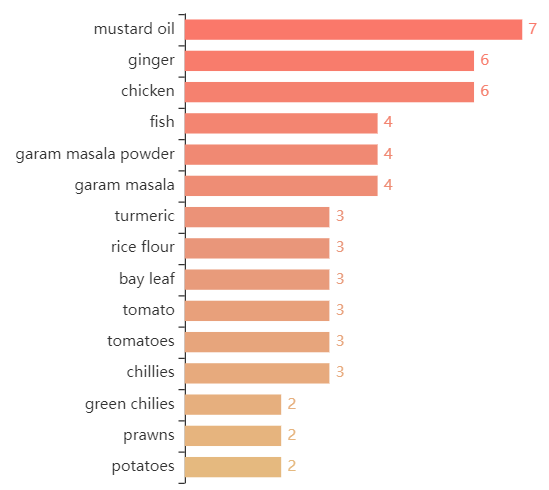
输出结果如下：

素食食材：



素食食材中，使用量较大的是sugar，ghee，urad dal等，而在48种使用了糖的美食中，有47种都是素食美食，酥油（ghee）也都是用在了素食上。

非素食食材：



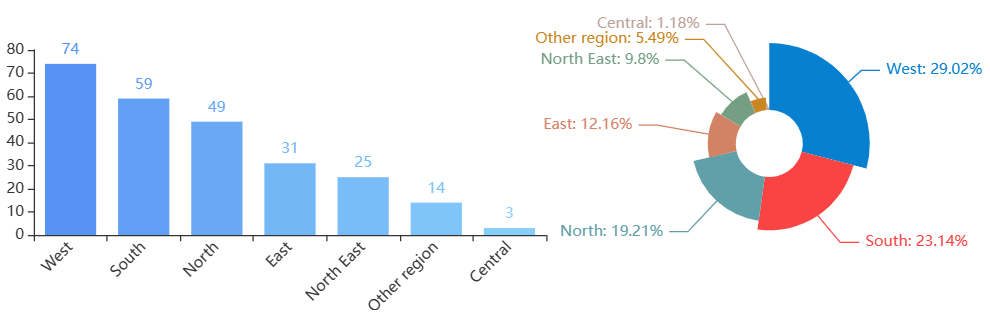
非素食食材中，使用量较大的与素食食材使用大相径庭，使用量较大的是mustard oil（芥末油），ginger（姜），chicken等。

### 5.6.5分析印度各地区菜肴的特色

分析各地区美食数量：

|  |
| --- |
| df\_region = df['region'].value\_counts().reset\_index()  #图形绘制  x\_data = df\_region['index'].to\_list()  def barPage() -> Bar:      bar1 = (          Bar()          .add\_xaxis(x\_data)          .add\_yaxis(              "",          df\_region['region'].to\_list(),              category\_gap="30%",   #柱子间隔              label\_opts=opts.LabelOpts(is\_show=True),          )      )      bar1.set\_global\_opts(          title\_opts=opts.TitleOpts(title="",pos\_left='16%',title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)),          xaxis\_opts=opts.AxisOpts(name\_rotate=60,axislabel\_opts={"rotate":45}),          visualmap\_opts=opts.VisualMapOpts(              type\_="color",              min\_=min(df\_region['region'].to\_list()),              max\_=max(df\_region['region'].to\_list()),              range\_color = ['#87CEFA','#5793f3'],              # is\_piecewise=True,              is\_show=False,          )      )      return bar1  def piePage() -> Pie:      colors=['#0780cf', '#fa4343', '#61a0a8', '#d48265', '#749f83', '#ca8622', '#bda29a']      pie2 = (Pie()          .add('', data\_pair,              radius=["20%", "60%"],              center=["50%", "50%"],#圆心位置，第一个坐标点为横轴位置，第二个坐标点为纵轴位置              rosetype="radius"              )              .set\_colors(colors)  # 设置颜色          .set\_global\_opts(title\_opts=opts.TitleOpts(title="",                                                      pos\_left = "43%",pos\_top = "5%"),                              legend\_opts=opts.LegendOpts(is\_show=False),)          .set\_series\_opts(label\_opts=opts.LabelOpts(formatter="{b}: {d}%",position='outside')) # 显示在图形的标签形式          )      return pie2  page = (      Page(layout=Page.DraggablePageLayout)      .add(          barPage(),          piePage(),      )  )  page.render('各地区美食种数组合图.html') |

输出结果如下

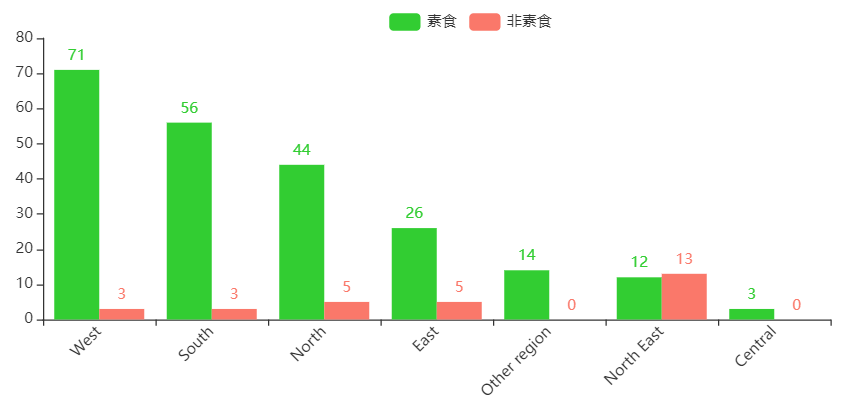


西部美食最多，有74种，其次是南部地区，有59种，美食种类最少的是中部地区，只有3种美食。

分析各地区素食和非素食美食数量：

|  |
| --- |
| df\_diet\_region = pd.pivot\_table(data=df,index='region',columns='diet',values='name',aggfunc='count')  df\_diet\_region.fillna(0,inplace=True)  df\_diet\_region.sort\_values('vegetarian',ascending=False,inplace=True)  # 绘制分组柱形图  x\_data = df\_diet\_region.index.to\_list()  bar = Bar()  bar.add\_xaxis(x\_data)  bar.add\_yaxis('素食', df\_diet\_region['vegetarian'].tolist(),gap="0%",color='#FA786A')  bar.add\_yaxis('非素食',df\_diet\_region['non vegetarian'].tolist(),gap="0%",color='#32CD32')  # bar.set\_series\_opts(label\_opts={'position': 'inside'})  bar.set\_global\_opts(      title\_opts=opts.TitleOpts(title="",                                pos\_left='15%',pos\_top="1%",                                title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)                                ),      xaxis\_opts=opts.AxisOpts(name\_rotate=60,axislabel\_opts={"rotate":45},is\_show=True),  )  bar.set\_series\_opts(      label\_opts=opts.LabelOpts(          position="top",          formatter=JsCode(              "function(params){return params.value.toFixed(0);}"          ),      ),  )  grid = Grid()  grid.add(bar,grid\_opts=opts.GridOpts(width='70%', height='45%',pos\_top="5%",pos\_left="12%", pos\_right="20%"), is\_control\_axis\_index=True)#width调整粗细  grid.render\_notebook() |

输出结果如下：

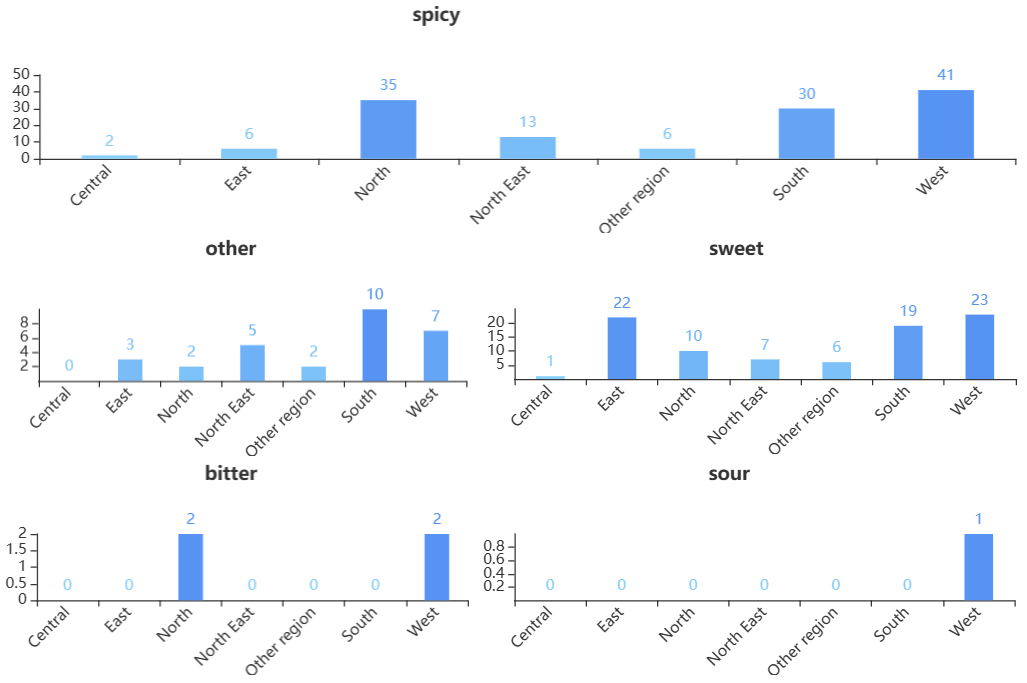


素食美食与上述各地区美食分布类似，而非素食美食种类最多的是东北地区，有13种美食。

分析各地区美食风味：

|  |
| --- |
| df\_flavor\_region = pd.pivot\_table(data=df,index='region',columns='flavor\_profile',values='name',aggfunc='count')  df\_flavor\_region.fillna(0,inplace=True)  x\_data = df\_flavor\_region.index.to\_list()  def barPage1() -> Bar:      bar1 = (          Bar()          .add\_xaxis(x\_data)          .add\_yaxis(              "",              df\_flavor\_region['other'].to\_list(),              category\_gap="60%",   #柱子间隔              label\_opts=opts.LabelOpts(is\_show=True),              color='#5793f3'          )      )      bar1.set\_global\_opts(          title\_opts=opts.TitleOpts(title="other",pos\_left='40%',title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)),          xaxis\_opts=opts.AxisOpts(name\_rotate=60,axislabel\_opts={"rotate":45},is\_show=True),          visualmap\_opts=opts.VisualMapOpts(              type\_="color",              min\_=min(df\_flavor\_region['other'].to\_list()),              max\_=max(df\_flavor\_region['other'].to\_list()),              range\_color = ['#87CEFA','#5793f3'],              # is\_piecewise=True,              is\_show=False,          )      )      return bar1  def barPage2() -> Bar:      bar2 = (          Bar()          .add\_xaxis(x\_data)          .add\_yaxis(              "",              df\_flavor\_region['sweet'].to\_list(),              category\_gap="60%",   #柱子间隔              label\_opts=opts.LabelOpts(is\_show=True),              color='#87CEFA'          )      )      bar2.set\_global\_opts(          title\_opts=opts.TitleOpts(title="sweet",pos\_left='40%',title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)),          xaxis\_opts=opts.AxisOpts(name\_rotate=60,axislabel\_opts={"rotate":45},is\_show=True),          visualmap\_opts=opts.VisualMapOpts(              type\_="color",              min\_=min(df\_flavor\_region['sweet'].to\_list()),              max\_=max(df\_flavor\_region['sweet'].to\_list()),              range\_color = ['#87CEFA','#5793f3'],              # is\_piecewise=True,              is\_show=False,          )      )      return bar2  def barPage3() -> Bar:      bar3 = (          Bar()          .add\_xaxis(x\_data)          .add\_yaxis(              "",              df\_flavor\_region['bitter'].to\_list(),              category\_gap="60%",   #柱子间隔              label\_opts=opts.LabelOpts(is\_show=True),              color='#87CEFA'          )      )      bar3.set\_global\_opts(          title\_opts=opts.TitleOpts(title="bitter",pos\_left='40%',title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)),          xaxis\_opts=opts.AxisOpts(name\_rotate=60,axislabel\_opts={"rotate":45},is\_show=True),          visualmap\_opts=opts.VisualMapOpts(              type\_="color",              min\_=min(df\_flavor\_region['bitter'].to\_list()),              max\_=max(df\_flavor\_region['bitter'].to\_list()),              range\_color = ['#87CEFA','#5793f3'],              # is\_piecewise=True,              is\_show=False,          )      )      return bar3  def barPage4() -> Bar:      bar4 = (          Bar()          .add\_xaxis(x\_data)          .add\_yaxis(              "",              df\_flavor\_region['sour'].to\_list(),              category\_gap="60%",   #柱子间隔              label\_opts=opts.LabelOpts(is\_show=True),              color='#87CEFA'          )      )      bar4.set\_global\_opts(          title\_opts=opts.TitleOpts(title="sour",pos\_left='40%',title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)),          xaxis\_opts=opts.AxisOpts(name\_rotate=60,axislabel\_opts={"rotate":45},is\_show=True),          visualmap\_opts=opts.VisualMapOpts(              type\_="color",              min\_=min(df\_flavor\_region['sour'].to\_list()),              max\_=max(df\_flavor\_region['sour'].to\_list()),              range\_color = ['#87CEFA','#5793f3'],              # is\_piecewise=True,              is\_show=False,          )      )      return bar4  def barPage5() -> Bar:      bar5 = (          Bar()          .add\_xaxis(x\_data)          .add\_yaxis(              "",              df\_flavor\_region['spicy'].to\_list(),              category\_gap="60%",   #柱子间隔              label\_opts=opts.LabelOpts(is\_show=True),              color='#87CEFA'          )          # .reversal\_axis()      )      bar5.set\_global\_opts(          title\_opts=opts.TitleOpts(title="spicy",pos\_left='40%',title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)),          xaxis\_opts=opts.AxisOpts(name\_rotate=60,axislabel\_opts={"rotate":45},is\_show=True),          visualmap\_opts=opts.VisualMapOpts(              type\_="color",              min\_=min(df\_flavor\_region['spicy'].to\_list()),              max\_=max(df\_flavor\_region['spicy'].to\_list()),              # min\_=0,              # max\_=6,              range\_color = ['#87CEFA','#5793f3'],              # is\_piecewise=True,              is\_show=False,          )      )      return bar5  page = (      Page(layout=Page.DraggablePageLayout)      .add(          barPage1(),          barPage2(),          barPage3(),          barPage4(),          barPage5(),      )  )  page.render('各地区美食风味.html')  Page.save\_resize\_html("各地区美食风味.html",   # 上面的HTML文件名称                        cfg\_file="chart\_config\_6.json",  # 保存的json配置文件                        dest="new\_各地区美食风味.html")  # 新HTML文件名称 |

输出结果如下：

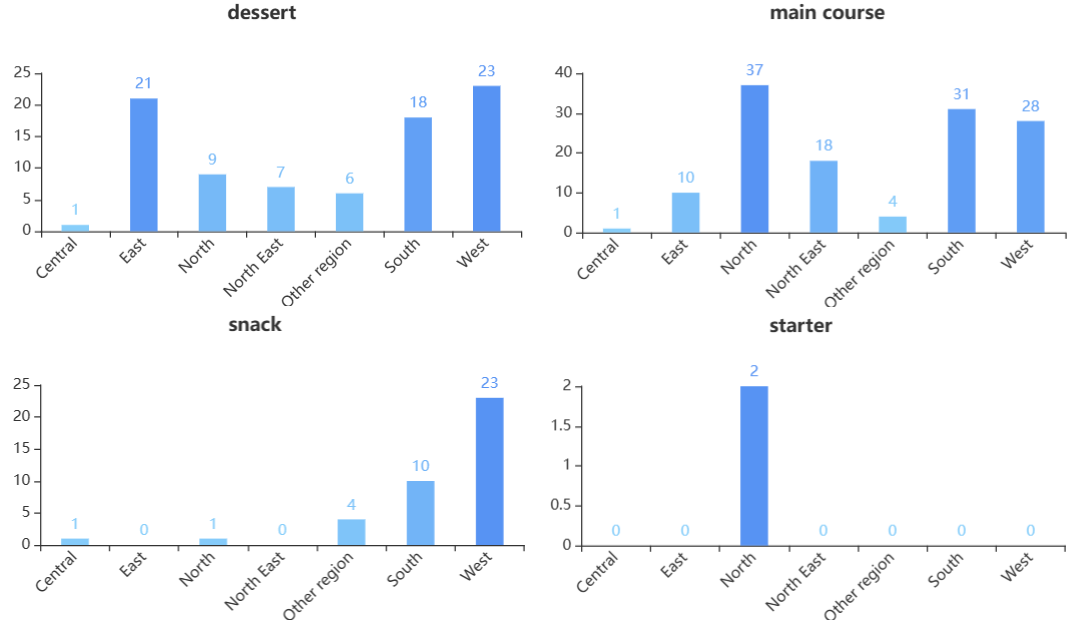


由图可以看出，辛辣风味美食最多的是印度西部地区，北部地区和南部地区也较多；其他口味美食最多的是南部地区，其次是西部地区；甜味口味美食最多的是西部地区，其次是东部地区；苦味口味美食较少，只有北部地区和西部地区才有，且都只有两种美食；酸味口味美食只有在西部地区才有，而且只有一种美食。

分析各地区美食类型美食数量：

|  |
| --- |
| df\_course\_region = pd.pivot\_table(data=df,index='region',columns='course',values='name',aggfunc='count')  df\_course\_region.fillna(0,inplace=True)  x\_data = df\_course\_region.index.to\_list()  def barPage1() -> Bar:      bar1 = (          Bar()          .add\_xaxis(x\_data)          .add\_yaxis(              "",              df\_course\_region['dessert'].to\_list(),              category\_gap="60%",   #柱子间隔              label\_opts=opts.LabelOpts(is\_show=True),              color='#5793f3'          )      )      bar1.set\_global\_opts(          title\_opts=opts.TitleOpts(title="dessert",pos\_left='40%',title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)),          xaxis\_opts=opts.AxisOpts(name\_rotate=60,axislabel\_opts={"rotate":45},is\_show=True),          visualmap\_opts=opts.VisualMapOpts(              type\_="color",              min\_=min(df\_course\_region['dessert'].to\_list()),              max\_=max(df\_course\_region['dessert'].to\_list()),              range\_color = ['#87CEFA','#5793f3'],              # is\_piecewise=True,              is\_show=False,          )      )      return bar1  def barPage2() -> Bar:      bar2 = (          Bar()          .add\_xaxis(x\_data)          .add\_yaxis(              "",              df\_course\_region['main course'].to\_list(),              category\_gap="60%",   #柱子间隔              label\_opts=opts.LabelOpts(is\_show=True),              color='#87CEFA'          )      )      bar2.set\_global\_opts(          title\_opts=opts.TitleOpts(title="main course",pos\_left='40%',title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)),          xaxis\_opts=opts.AxisOpts(name\_rotate=60,axislabel\_opts={"rotate":45},is\_show=True),          visualmap\_opts=opts.VisualMapOpts(              type\_="color",              min\_=min(df\_course\_region['main course'].to\_list()),              max\_=max(df\_course\_region['main course'].to\_list()),              range\_color = ['#87CEFA','#5793f3'],              # is\_piecewise=True,              is\_show=False,          )      )      return bar2  def barPage3() -> Bar:      bar3 = (          Bar()          .add\_xaxis(x\_data)          .add\_yaxis(              "",              df\_course\_region['snack'].to\_list(),              category\_gap="60%",   #柱子间隔              label\_opts=opts.LabelOpts(is\_show=True),              color='#87CEFA'          )      )      bar3.set\_global\_opts(          title\_opts=opts.TitleOpts(title="snack",pos\_left='40%',title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)),          xaxis\_opts=opts.AxisOpts(name\_rotate=60,axislabel\_opts={"rotate":45},is\_show=True),          visualmap\_opts=opts.VisualMapOpts(              type\_="color",              min\_=min(df\_course\_region['snack'].to\_list()),              max\_=max(df\_course\_region['snack'].to\_list()),              range\_color = ['#87CEFA','#5793f3'],              # is\_piecewise=True,              is\_show=False,          )      )      return bar3  def barPage4() -> Bar:      bar4 = (          Bar()          .add\_xaxis(x\_data)          .add\_yaxis(              "",              df\_course\_region['starter'].to\_list(),              category\_gap="60%",   #柱子间隔              label\_opts=opts.LabelOpts(is\_show=True),              color='#87CEFA'          )      )      bar4.set\_global\_opts(          title\_opts=opts.TitleOpts(title="starter",pos\_left='40%',title\_textstyle\_opts=opts.TextStyleOpts(font\_size=15)),          xaxis\_opts=opts.AxisOpts(name\_rotate=60,axislabel\_opts={"rotate":45},is\_show=True),          visualmap\_opts=opts.VisualMapOpts(              type\_="color",              min\_=min(df\_course\_region['starter'].to\_list()),              max\_=max(df\_course\_region['starter'].to\_list()),              range\_color = ['#87CEFA','#5793f3'],              # is\_piecewise=True,              is\_show=False,          )      )      return bar4  page = (      Page(layout=Page.DraggablePageLayout)      .add(          barPage1(),          barPage2(),          barPage3(),          barPage4(),      )  )  page.render('各地区美食类型.html')  Page.save\_resize\_html("各地区美食类型.html",   # 上面的HTML文件名称                        cfg\_file="chart\_config\_5.json",  # 保存的json配置文件                        dest="new\_各地区美食类型.html")  # 新HTML文件名称 |

输出结果如下：



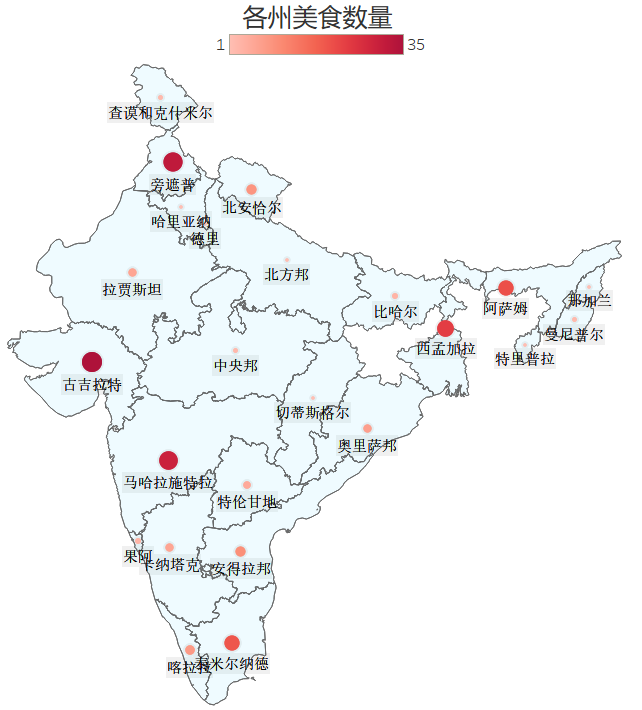
由图可以看出，甜点类型美食最多的是印度西部地区，东部地区和南部地区也较多；主食类型美食最多的是北部地区，其次是南部地区、西部地区；小吃类型美食最多的是西部地区，其次是南部地区；开胃菜美食较少，只有北部地区

### 5.6.6分析印度各州菜肴的特色

为方便分析，我们将各州名称翻译为中文。

分析各州美食数量：

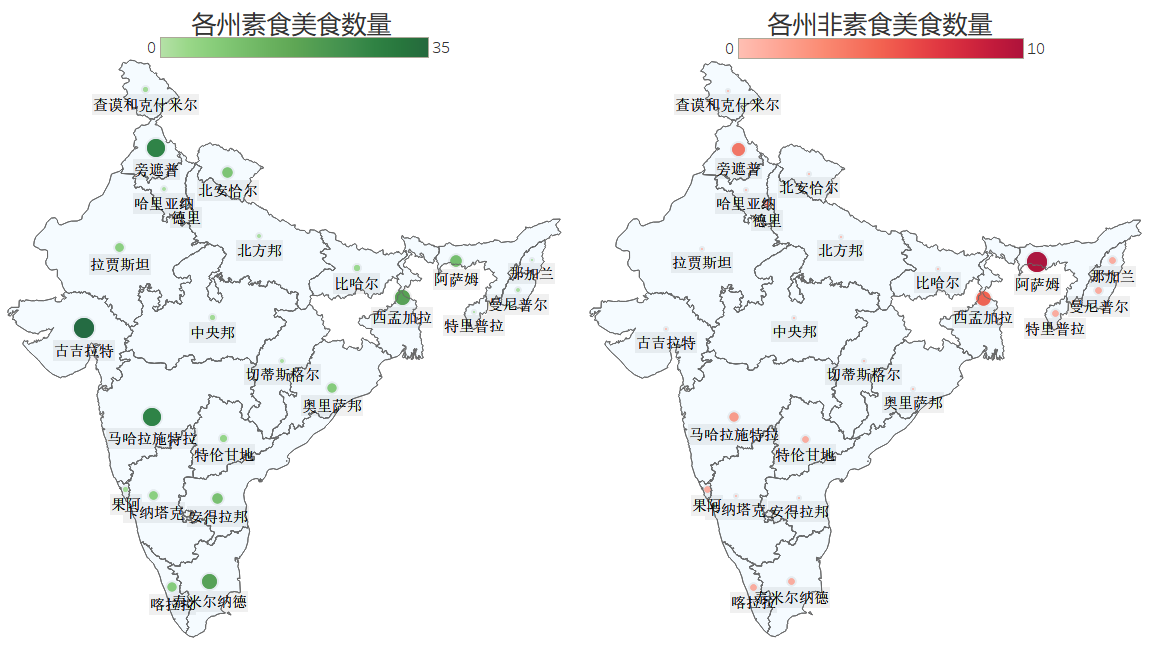
无代码，使用tableau绘制：



可以看到，美食种类较多的有吉吉拉特、旁遮普和马哈拉施特拉，中部的中央邦、切蒂斯格尔等州美食种类较少。

分析各州素食和非素食美食数量：

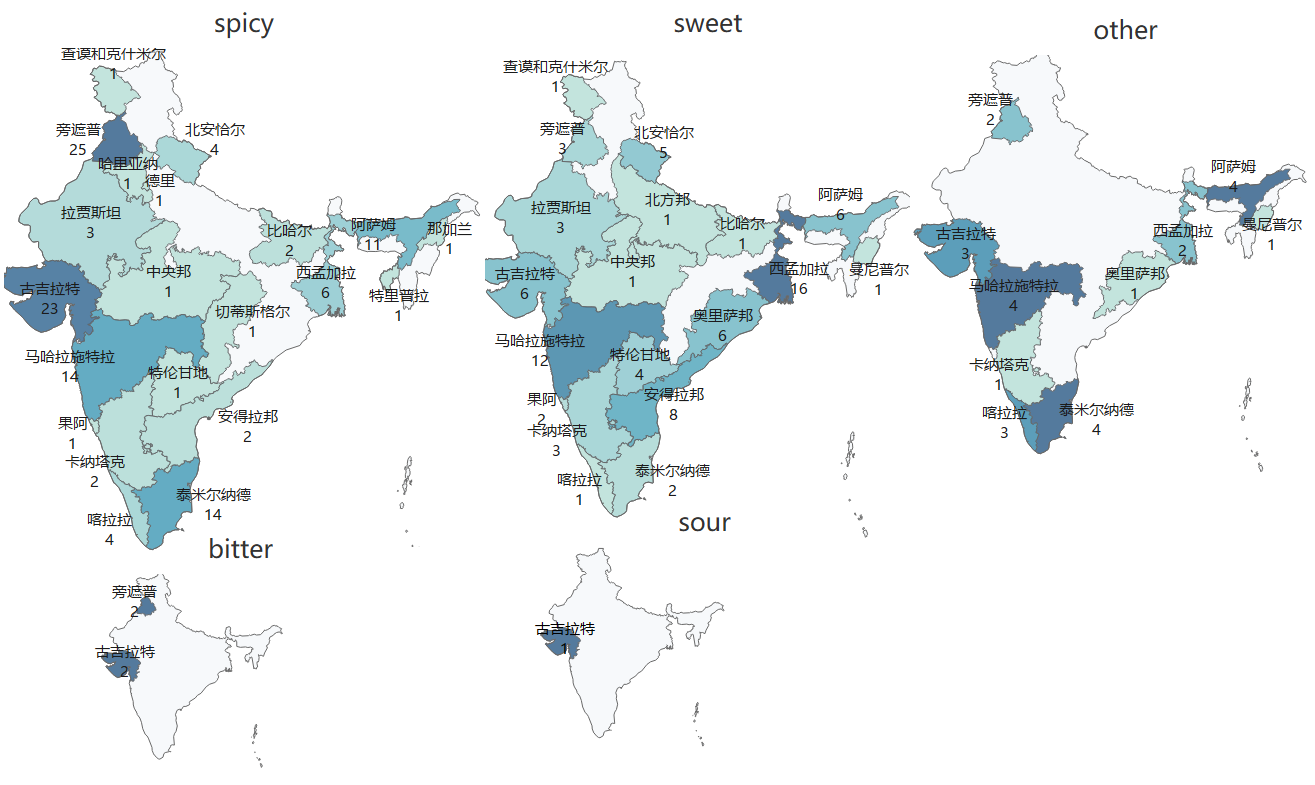
无代码，使用tableau绘制：



素食美食较多的州有吉吉拉特、旁遮普和马哈拉施特拉；非素食美食种类较多的州有阿萨姆、西孟加拉以及旁遮普。

分析各州美食风味：

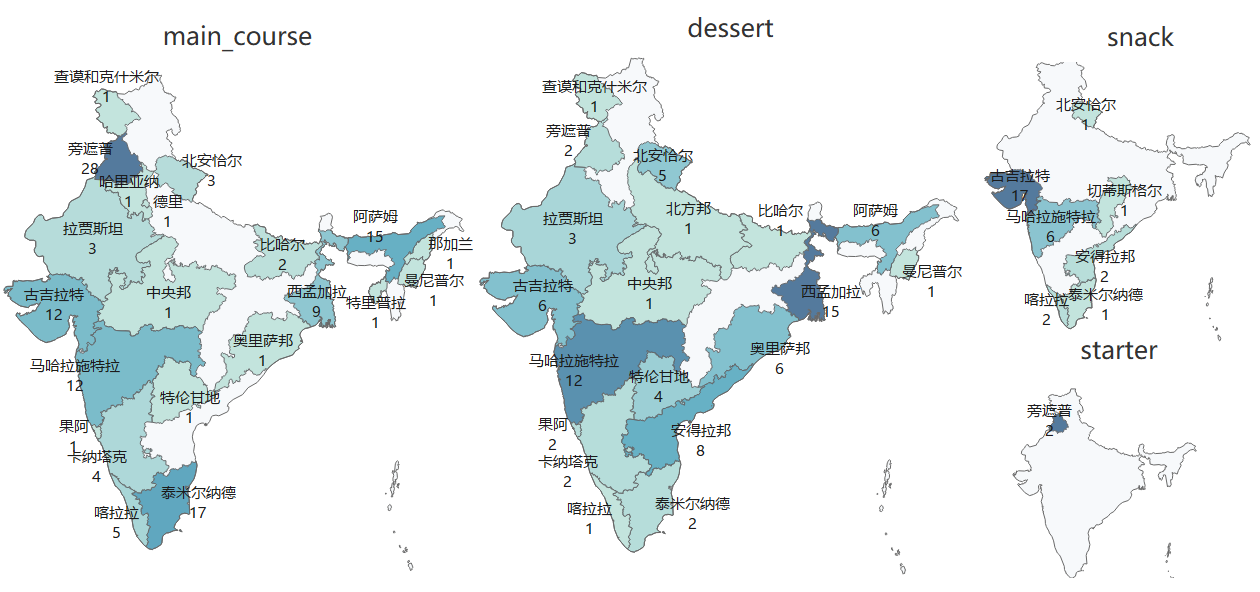
无代码，使用tableau绘制：



由图可以看出，辛辣风味美食最多的州是旁遮普，吉吉拉特的美食也较多；甜味口味美食最多的是西孟加拉，其次是马哈拉施特拉；其他口味美食最多的是阿萨姆、马哈拉施特拉和泰米尔纳德；苦味口味美食较少，只有旁遮普和吉吉拉特才有，且都只有两种美食；酸味口味美食只有在吉吉拉特才有，而且只有一种美食。

分析各地区美食类型美食数量：

无代码，使用tableau绘制：



由图可以看出，主食类型美食最多的州是旁遮普，阿萨姆和泰米尔纳德也较多；甜点类型美食最多的是西孟加拉，其次是马哈拉施特拉；小吃类型美食最多的是吉吉拉特，其次是马哈拉施特拉；开胃菜美食较少，只有旁遮普才有。

## 5.7相似性分析

此项目采用基于余弦相似度计算每个美食之间的相似性。余弦相似度是用向量空间中两个向量夹角的余弦值作为衡量两个个体间差异的大小的度量。余弦值越接近1，就表明夹角越接近0度，也就是两个向量越相似，这就叫"余弦相似性"。

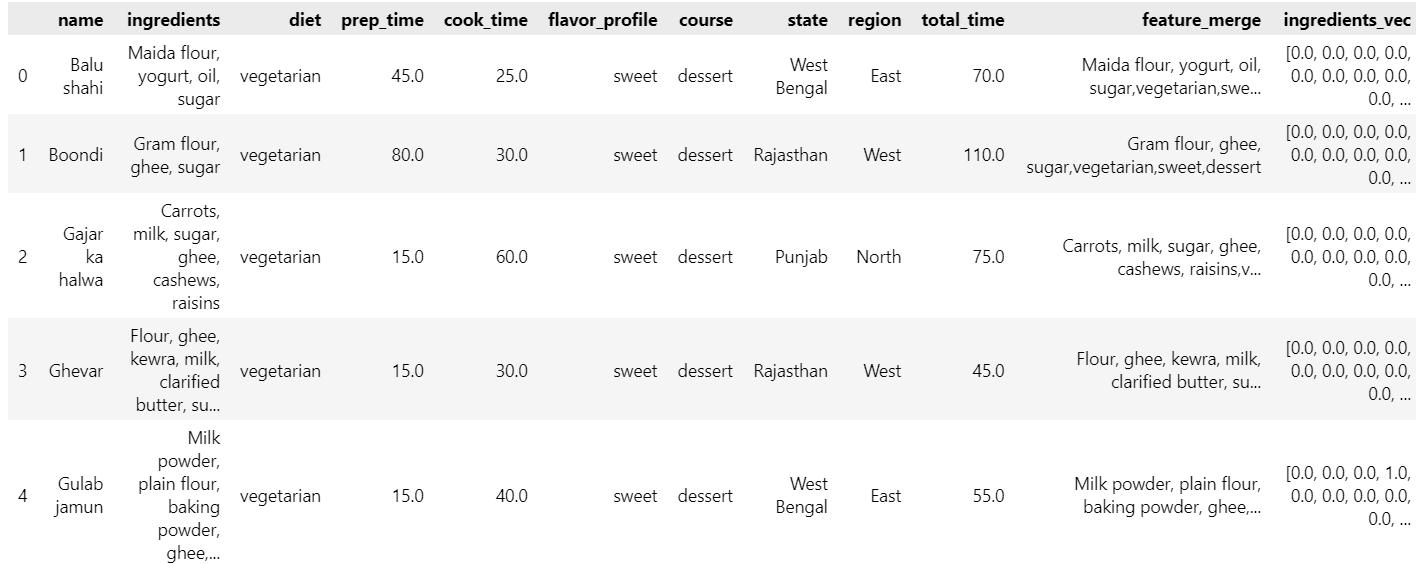
通过使用余弦相似性度量方法，我们可以将各个美食之间的特征进行比较，并获得一个数值化的相似性度量结果。这样的计算目的是为了帮助人们在选择美食时，可以更好地了解各种美食之间的差异和相似之处，从而做出更准确的选择或推荐。

### 5.7.1数据再处理

此项目余弦相似度计算是基于特征ingredients的各个食材、diet特征、flavor\_profile特征以及course特征。为方便特征构造向量，将这四个特征合并为一列。然后对这一特征转换成特征向量。代码如下：

|  |
| --- |
| df['feature\_merge']=df['ingredients']+','+df['diet']+','+df['flavor\_profile']+','+df['course']  ingredientsList=[]  for i in range(0,len(df)):      ingredientsList.append(df['feature\_merge'][i].split(","))  flat\_list = []  for sublist in ingredientsList:      for item in sublist:          str1 = item.lstrip()          str1 = str1.lower()          str2 = str1.rstrip()          flat\_list.append(str2)  words = np.array(list(set(flat\_list)))  def gen\_ingredients\_vector(ingredients):      ingredients\_vec = np.zeros(words.shape)      flat\_list = [sublist.lower() for sublist in ingredients.split(",")]      s = [a.lstrip() for a in flat\_list]      s = set([m.rstrip() for m in s])      for ingredient in s:          idx = np.where(words == ingredient)          ingredients\_vec[idx] = 1      return ingredients\_vec.tolist()  df["ingredients\_vec"] = df["feature\_merge"].map(gen\_ingredients\_vector) |

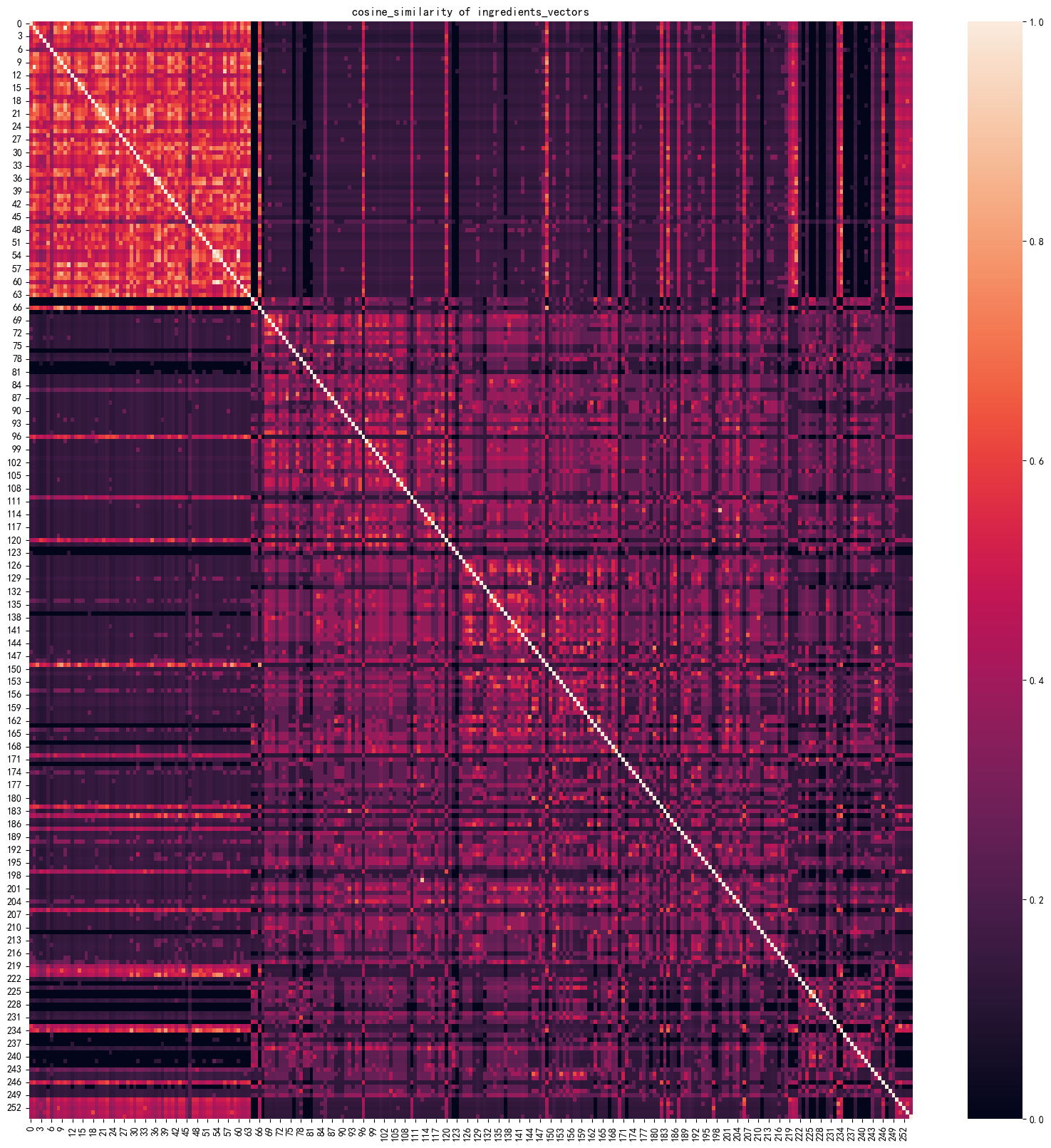
输出结果如下：



### 5.7.2 计算余弦相似度

|  |
| --- |
| ingredients\_vecs = []  for i in range(len(df)):      ingredients\_vecs.append(df["ingredients\_vec"][i])  ingredients\_vecs = np.array(ingredients\_vecs)  cos\_matrix = cosine\_similarity(ingredients\_vecs, ingredients\_vecs)  plt.figure(figsize=(20, 20))  ax = sns.heatmap(cos\_matrix)  ax.set\_title("cosine\_similarity of ingredients\_vectors") |

输出结果如下：

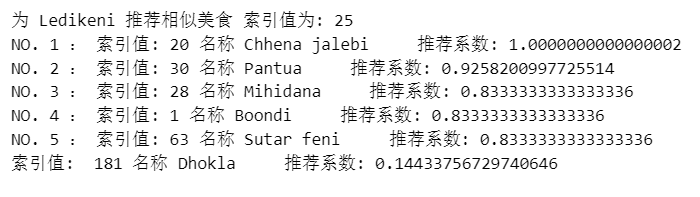


可以看到，这是一个255\*255的相似度热力图，索引值为0~63的相似度较高，说明这些美食之间的相似性较高。索引为0~63的美食与64~255的美食之间有少部分相似性较高，而64~255的美食之间的相似度大部分都比较一般。

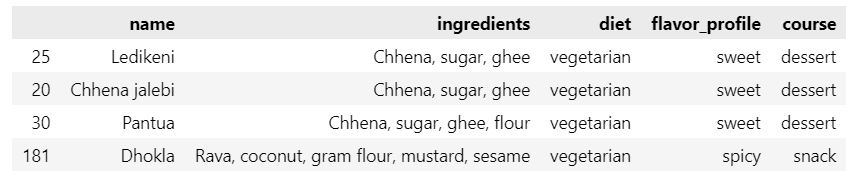
我们选取几个美食，并验证其配料和其它属性。代码如下：

|  |
| --- |
| food = 'Ledikeni'  target\_index = df[df["name"] == food].index[0]  cos\_matrix[target\_index][target\_index] = 0  print('为',food,'推荐相似美食','索引值为:',target\_index)  target\_vector = cos\_matrix[target\_index]  top\_indices = target\_vector.argsort()[-5:][::-1]  top\_indices1 = target\_vector.argsort()[50:51][::-1]  top\_indices  similar\_foods\_with\_similarity = []  for i,index in enumerate(top\_indices):      similarity\_value = cos\_matrix[target\_index][index]      print('NO.',i+1,'：','索引值:',index,'名称',df.loc[index]["name"],"    推荐系数:",similarity\_value)  for i,index in enumerate(top\_indices1):      similarity\_value = cos\_matrix[target\_index][index]      print('索引值: ',index,'名称',df.loc[index]["name"],"    推荐系数:",similarity\_value)  df[['name','ingredients','diet','flavor\_profile','course']].iloc[[25,20,30,181]] |

输出结果如下：



验证数据



从结果上可以看到，与美食'Ledikeni'的相似度的美食中，Chhena jalebi与其相似度为1，即完全一样，其次是Pantua，相似度为0.9258，而Dhokla与其相似度只有0.1443，相似性非常低。输出这些美食的配料以及其它属性，验证我们计算的相似性是否可靠，从数据上看'Ledikeni'与Chhena jalebi的成分及其它类型都完全一致，'Ledikeni'与Pantua也极其相似，Pantua只多了一个配料flour，而'Ledikeni'与Dhokla的比较可以发现，两美食的共同点只有diet属性，都是素食。

## 5.8开发美食推荐系统

基于数据分析和相似性分析的结果，构建一个印度美食推荐系统。近年来，随着互联网和移动设备的普及，人们越来越依赖于推荐系统来发现新的餐厅和美食。然而，在海量的印度美食选择中，如何为用户提供个性化的推荐仍然是一个挑战。因此，本文利用数据分析和相似性分析得到的结论，构建一个准确、高效、面向用户的印度美食推荐系统，以满足用户对于印度美食的需求。

在数据分析方面，我们可以发掘出印度美食所在地区、各地区的美食特色以及对于各各美食特色所推荐的地区。这些分析结果将作为推荐系统的基础，为用户提供个性化的印度美食推荐。

在相似性分析方面，我们可以根据255种美食之间的余弦相似性发掘出根据某个美食来推荐相似的美食，以及比较两美食之间的差异与相似性。并基于数据分析得出的结果进行优化，以提供更准确、个性化的推荐。

本文使用python的tkinter库来开发系统，代码如下：

|  |
| --- |
| import tkinter as tk  import pandas as pd  import numpy as np  from tkinter import ttk  from tkinter import font  from tkinter import messagebox  df = pd.read\_csv('model\_data.csv')  df.drop(columns='Unnamed: 0',inplace=True)  cos\_matrix = np.load("cosine\_similarity\_matrix.npy")  #由美食查找城市  def food\_of\_state():   #相似美食推荐      food = dropdown1.get()  # 获取用户输入的美食名称      if food in df["name"].values:          target\_index = df[df["name"] == food].index[0]          food\_state = df['state'][target\_index]          result\_text.insert(tk.END, f"在{food\_state}能找到这种美食\n")      elif food=="请输入或选择美食" or food=="":          result\_text.insert(tk.END, f"请输入内容!\n")      else:          result\_text.insert(tk.END, f"查找不到 {food} 这种美食,请检查您是否拼写有误,名称首字母需大写!\n")  #由城市推荐风味  def state\_to\_flavor():      state = dropdown2.get()      if state in df["state"].values:          df\_flavor\_state = pd.pivot\_table(data=df,index='state',columns='flavor\_profile',values='name',aggfunc='count').reset\_index()          df\_flavor\_state.fillna(0,inplace=True)          df\_flavor\_state.set\_index('state',inplace=True)          df\_flavo\_sort = df\_flavor\_state.loc[state].reset\_index()          df\_flavo\_sort = df\_flavo\_sort.sort\_values(state,ascending=False)          name1 = df\_flavo\_sort.iloc[0]["flavor\_profile"]          score1 = df\_flavo\_sort.iloc[0][state]          name2 = df\_flavo\_sort.iloc[1]["flavor\_profile"]          score2 = df\_flavo\_sort.iloc[1][state]          if score1 < 5 or score2 == 0:              result\_text.insert(tk.END, f"该州美食风味并不多,推荐以下美食风味：{name1}，美食种数：{score1};\n")          else:              result\_text.insert(tk.END, f"该州中,推荐以下美食风味：{name1}，美食种数：{score1};\n{name2}，美食种数：{score2}\n")      elif state=="请输入或选择州" or state=="":          result\_text.insert(tk.END, f"请输入内容!\n")      else:          result\_text.insert(tk.END, f"查找不到 {state} 这个州,请检查您是否拼写有误,名称首字母需大写!\n")  #由风味推荐城市  def flavor\_to\_state():      flavor = dropdown3.get()      if flavor in df["flavor\_profile"].values:          df\_state\_flavor = pd.pivot\_table(data=df,index='flavor\_profile',columns='state',values='name',aggfunc='count').reset\_index()          df\_state\_flavor.fillna(0,inplace=True)          df\_state\_flavor.set\_index('flavor\_profile',inplace=True)          df\_state\_sort = df\_state\_flavor.loc[flavor].reset\_index()          df\_state\_sort = df\_state\_sort.sort\_values(flavor,ascending=False)          name1 = df\_state\_sort.iloc[0]["state"]          score1 = df\_state\_sort.iloc[0][flavor]          name2 = df\_state\_sort.iloc[1]["state"]          score2 = df\_state\_sort.iloc[1][flavor]          name3 = df\_state\_sort.iloc[2]["state"]          score3 = df\_state\_sort.iloc[2][flavor]          if score1 < 5 or score2 == 0:              result\_text.insert(tk.END, f"该美食风味较少,只推荐去这个州：{name1}，美食种数：{score1};\n")          else:              result\_text.insert(tk.END, f"对于该美食风味,推荐去这三个州：\n{name1}，美食种数：{score1};\n{name2}，美食种数：{score2};\n{name3}，美食种数：{score3}\n")  #由城市推荐类型  def state\_to\_crouse():      state = dropdown2.get()      if state in df["state"].values:          df\_state\_course = pd.pivot\_table(data=df,index='state',columns='course',values='name',aggfunc='count').reset\_index()          df\_state\_course.fillna(0,inplace=True)          df\_state\_course.set\_index('state',inplace=True)          df\_state\_sort = df\_state\_course.loc[state].reset\_index()          df\_state\_sort = df\_state\_sort.sort\_values(state,ascending=False)          name1 = df\_state\_sort.iloc[0]["course"]          score1 = df\_state\_sort.iloc[0][state]          name2 = df\_state\_sort.iloc[1]["course"]          score2 = df\_state\_sort.iloc[1][state]          if score1 < 5 or score2 == 0:              result\_text.insert(tk.END, f"该州美食类型并不多,推荐以下美食类型：{name1}，美食种数：{score1};\n")          else:              result\_text.insert(tk.END, f"对于该城市,推荐两种美食类型：\n{name1}，美食种数：{score1};\n{name2}，美食种数：{score2};\n")      elif state=="请输入或选择州" or state=="":          result\_text.insert(tk.END, f"请输入内容!\n")      else:          result\_text.insert(tk.END, f"查找不到 {state} 这个城市,请检查您是否拼写有误,名称首字母需大写!\n")  #由类型推荐城市  def course\_to\_state():      course = dropdown3.get()      if course in df["course"].values:          df\_course\_state = pd.pivot\_table(data=df,index='course',columns='state',values='name',aggfunc='count').reset\_index()          df\_course\_state.fillna(0,inplace=True)          df\_course\_state.set\_index('course',inplace=True)          df\_course\_sort = df\_course\_state.loc[course].reset\_index()          df\_course\_sort = df\_course\_sort.sort\_values(course,ascending=False)          name1 = df\_course\_sort.iloc[0]["state"]          score1 = df\_course\_sort.iloc[0][course]          name2 = df\_course\_sort.iloc[1]["state"]          score2 = df\_course\_sort.iloc[1][course]          name3 = df\_course\_sort.iloc[2]["state"]          score3 = df\_course\_sort.iloc[2][course]          # result\_text.delete(1.0, tk.END)  # 清空结果文本框          if score1 < 5 or score2 == 0:              result\_text.insert(tk.END, f"该美食类型较少,只推荐去这个州：{name1}，美食种数：{score1};\n")          else:              result\_text.insert(tk.END, f"对于该美食类型,推荐去这三个州：\n{name1}，美食种数：{score1};\n{name2}，美食种数：{score2};\n{name3}，美食种数：{score3}\n")  #由已知美食推荐相似美食  def recommend\_similar\_food():   #相似美食推荐      food = dropdown1.get()  # 获取用户输入的美食名称      if food in df["name"].values:          target\_index = df[df["name"] == food].index[0]          cos\_matrix[target\_index][target\_index] = 0      # 将与自身余弦相似值设为0，避免干扰          # result\_text.delete(1.0, tk.END)  # 清空结果文本框          result\_text.insert(tk.END, f"为 {food} 推荐10种相似美食：\n")          target\_vector = cos\_matrix[target\_index]          top\_indices = target\_vector.argsort()[-10:][::-1]          for i, index in enumerate(top\_indices):              similarity\_value = round(cos\_matrix[target\_index][index],3)              result\_text.insert(                  tk.END, f"NO.{i+1}：索引值:{index}  名称:{df.loc[index]['name']}  \t推荐系数:{similarity\_value}\n")      elif food=="请输入或选择美食" or food=="":          result\_text.insert(tk.END, f"请输入内容!\n")      else:          # result\_text.delete(1.0, tk.END)  # 清空结果文本框          result\_text.insert(tk.END, f"查找不到 {food} 这种美食,请检查您是否拼写有误,名称首字母需大写!\n")  #对比两美食之间的差异和相似性  def food\_diff\_and\_similar():      food1 = dropdown4.get()      food2 = dropdown5.get()      if food1 in df["name"].values and food2 in df["name"].values:          target\_index1 = df[df["name"] == food1].index[0]          target\_index2 = df[df["name"] == food2].index[0]          cos\_similar = round(cos\_matrix[target\_index1][target\_index2],3)          ingred1 = df['ingredients'][target\_index1]          diet1 = df['diet'][target\_index1]          flavor1 = df['flavor\_profile'][target\_index1]          course1 = df['course'][target\_index1]          ingred2 = df['ingredients'][target\_index2]          diet2 = df['diet'][target\_index2]          flavor2 = df['flavor\_profile'][target\_index2]          course2 = df['course'][target\_index2]          if cos\_similar >= 0.8:              result\_text.insert(tk.END, f"这两种美食的相似性 较高,相似度为：{cos\_similar}\n")              result\_text.insert(tk.END, f"两美食的配料及其他属性为:\n美食1 {food1}：\n    配料：{ingred1}\n    类型：{diet1}，{course1}\n    风味：{flavor1}\n\  美食2 {food2}：\n    配料:{ingred2}\n    类型：{diet2}，{course2}\n    风味：{flavor2}\n")          elif cos\_similar >= 0.5:              result\_text.insert(tk.END, f"这两种美食的相似性 一般,相似度为：{cos\_similar}\n")              result\_text.insert(tk.END, f"两美食的配料及其他属性为:\n美食1 {food1}：\n    配料：{ingred1}\n    类型：{diet1}，{course1}\n    风味：{flavor1}\n\  美食2 {food2}：\n    配料:{ingred2}\n    类型：{diet2}，{course2}\n    风味：{flavor2}\n")          else:              result\_text.insert(tk.END, f"这两种美食的相似性 较低,相似度为：{cos\_similar}\n")              result\_text.insert(tk.END, f"两美食的配料及其他属性为:\n美食1 {food1}：\n    配料:{ingred1}\n    类型：{diet1}，{course1}\n    风味：{flavor1}\n\  美食2 {food2}：\n    配料:{ingred2}\n    类型：{diet2}，{course2}\n    风味：{flavor2}\n")      elif food1 not in df["name"].values:          # result\_text.delete(1.0, tk.END)  # 清空结果文本框          result\_text.insert(tk.END, f"查找不到 {food1} 这种美食,请检查您是否拼写有误,名称首字母需大写!\n")      elif food2 not in df["name"].values:          result\_text.insert(tk.END, f"查找不到 {food2} 这种美食,请检查您是否拼写有误,名称首字母需大写!\n")      elif food1=="美食1" or food2=="美食2":          result\_text.insert(tk.END, f"请输入内容!\n")  #清空文本内容函数  def clear\_text():      result\_text.delete(1.0, tk.END)  #退出程序函数  def quit\_app():      if messagebox.askokcancel("退出", "确定要退出应用程序吗？\n\t--GenhaoX"):          window.destroy()  def course\_and\_flavor():      name = dropdown3.get()      if name in df["course"].values:          return course\_to\_state()      elif name in df["flavor\_profile"].values:          return flavor\_to\_state()      elif name=="请输入或选择风味/类型" or name=="":          return result\_text.insert(tk.END, f"请输入内容!\n")      else:          return result\_text.insert(tk.END, f"查找不到 {name} 这种美食风味或类型,请检查您是否拼写有误,名称首字母需大写!\n")  #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ##############################################################################################################################################  # 创建主窗口  window = tk.Tk()  window.title("印度美食推荐系统-------------------------------------------------------------------------------------------------GenhaoX")  window.geometry("800x600")  window.my\_font\_tip=font.Font(family="微软雅黑",size=10,slant=font.ITALIC)#提示词字体  window.my\_font\_input=font.Font(family="微软雅黑",size=10)#提示词字体  style = ttk.Style()  style.configure('TCombobox', foreground='gray', font=('Arial', 10, 'italic'))  style.map('TCombobox',            fieldforeground=[('readonly', 'gray'), ('!disabled', 'black')],            fieldbackground=[('readonly', 'white'), ('!disabled', 'white')])  # # 创建标签和输入框  label0 = tk.Label(window, text="面向旅游用户的美食推荐小程序",font=("华文行楷", 18), fg="green")  label0.place(x=205,y=2)  label1 = tk.Label(window, text="Food   name：",font=("Times New Roman", 15), fg="black")  label1.place(x=100,y=44)  dropdown1 = ttk.Combobox(window, width=18,values=list(set(df["name"])))  dropdown1.set("请输入或选择美食")  dropdown1.place(x=210,y=45)  #第二行\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  label2 = tk.Label(window, text="State   name：",font=("Times New Roman", 15), fg="black")  label2.place(x=100,y=85)  dropdown2 = ttk.Combobox(window, width=18,values=list(set(df["state"])))  dropdown2.set("请输入或选择州")  dropdown2.place(x=210,y=87)  #第三行\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  label3 = tk.Label(window, text="Flavor/Course：",font=("Times New Roman", 13), fg="black")  label3.place(x=100,y=123)  dropdown3 = ttk.Combobox(window, width=18,values=['bitter', 'other', 'sour', 'spicy', 'sweet','dessert', 'main course', 'snack', 'starter'])  dropdown3.set("请输入或选择风味/类型")  dropdown3.place(x=210,y=126)  #第四行  label4 = tk.Label(window, text="Food1&Food2：",font=("Times New Roman", 13), fg="black")  label4.place(x=100,y=171)  dropdown4 = ttk.Combobox(window, width=18,values=list(set(df["name"])))  dropdown4.set("请输入美食1")  dropdown4.place(x=210,y=162)  dropdown5 = ttk.Combobox(window, width=18,values=list(set(df["name"])))  dropdown5.set("请输入美食2")  dropdown5.place(x=210,y=185)  # 添加一个箭头  label4 = tk.Label(window, text="-->",font=("微软雅黑", 15), fg="black")  label4.place(x=365,y=39)  label5 = tk.Label(window, text="-->",font=("微软雅黑", 15), fg="black")  label5.place(x=365,y=78)  label6 = tk.Label(window, text="-->",font=("微软雅黑", 15), fg="black")  label6.place(x=365,y=118)  label7 = tk.Label(window, text="-->",font=("微软雅黑", 15), fg="black")  label7.place(x=365,y=168)  # # # 创建按钮  # 第一行\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  button1 = tk.Button(window, text="美食所在州", command=food\_of\_state,width=18)  button1.place(x=410,y=40)  button2 = tk.Button(window, text="推荐相似美食", command=recommend\_similar\_food,width=18)  button2.grid(row=1, column=1,sticky='w')  button2.place(x=555,y=40)  #第二行\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  button3 = tk.Button(window, text="查找推荐美食风味", command=state\_to\_flavor,width=18)  button3.place(x=410,y=80)  button5 = tk.Button(window, text="查找推荐美食类型", command=state\_to\_crouse,width=18)  button5.place(x=555,y=80)  #第三行\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  button4 = tk.Button(window, text="查找推荐的城市", command=course\_and\_flavor,width=39)  button4.place(x=410,y=120)  #第四行\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  button6 = tk.Button(window, text="比较差异与相似性", command=food\_diff\_and\_similar,width=39)  button6.place(x=410,y=168)  #添加输出文本框  result\_text = tk.Text(window, font=("微软雅黑", 12),height=15,width=75)  result\_text.place(x=58,y=210)#x加40,y加90  clear\_button = tk.Button(window, text="清空文本", command=clear\_text,width=12)  clear\_button.place(x=100,y=535)#x加40,y加90  quit\_button = tk.Button(window, text="退出程序", command=quit\_app,width=12)  quit\_button.place(x=600,y=535)#x加40,y加90  window.mainloop() |

输出结果如下：



通过这个小程序，用户就可以更加便利的寻找喜爱的美食。

## 5.9 分析总结与建议

1. 根据各地区、各州的素食和非素食美食特色数据分析结果，虽然印度当地美食十分丰富,但是绝大多数都是素食美食，印度大部分地区因为宗教信仰，所以非素食美食十分稀少。如果游客想要前往印度旅游,那么建议游客在此之前需要考虑当地的美食类型是否合口，如果是想要品尝印度肉食美食，我们并不建议游客前往印度，若还是想要去的话，我们推荐去印度的阿萨姆邦和西孟加拉邦；若游客十分喜爱素食美食，我们强烈推荐去印度的北部地区旁遮普邦、西部地区吉吉拉特以及马哈拉施特拉邦。
2. 根据印度美食食材及各类型，各风味数据分析结果，印度255种美食总共使用了365种食材，其中素食美食使用了321种食材，非素食美食使用了96种食材。在所有美食使用的食材中，糖使用的最多，而这些使用了糖的美食绝大多数都是素食，它们在许多传统食谱中都有使用，从所有美食的风味分布、类型分布中可以看出，甜味风味美食占比较高，甜点类型美食占比也较高，表明印度当地的甜味、甜点美食众多，喜欢吃甜味素食的游客可以前往印度的东北地区的西孟加拉邦以及西部地区马哈拉施特拉邦。姜的使用量也比较高，姜是一种辣味调料，而无论是素食还是非素食，印度美食风味中，占比最高的都是辛辣风味美食，那么我们推荐喜爱吃辛辣口味美食的游客一定不能错过。
3. 根据相似性分析及开发的美食推荐系统结果，如果游客在印度当地品尝了一款口味不错或者口感糟糕的美食，又想要再找一种与之相似的美食，继续品尝或者避免再次踩坑，那么一定要使用我们开发的美食推荐系统，它会根据两美食的余弦相似度计算出两者的相似性，越接近1，说明越相似。游客可以根据已知一种美食，来寻找其它254种美食中哪种与其最相似，也可以根据已知两种美食计算两者相似性。与此同时还有其它四种功能：根据已知美食寻找该美食所在州、根据想要吃的美食类型或者风味，推荐去哪些州以及根据所在州，推荐美食类型或风味。

## 5.10项目总结

**项目名称**：印度美食分析与相似性研究

**项目背景和目的**：本项目旨在通过分析印度美食通过研究分析印度美食，帮助游客更好地理解不同地区的菜肴特色，为游客提供更好的服务。

**数据分析方法**：本项目采用了数据清洗、数据可视化、统计分析和余弦相似度等方法。首先对数据进行清洗，然后采用数据可视化方法对数据进行探索性分析，通过数据分析发掘出各类型、风味、地区、州的美食特色，最后使用余弦相似度计算各美食之间的相似度，以此分析各美食的差异和相似性。

**数据分析结果**：通过数据分析，得出以下结论：

1. 通过分析素食和非素食的美食特色，发现255种美食，素食占了绝大多数，这显示出了印度素食主义文化的深厚影响。素食是印度文化的一个重要组成部分，因为根据印度教、耆那教和锡克教的教义，禁止食用肉类。
2. 通过分析各菜肴风味特色，这反映了印度菜肴普遍具有浓郁的香料和辛辣调味的特点。辣味在印度饮食中被广泛使用，为菜品带来独特的风味和火辣的口感；不仅如此，印度还有着丰富的甜点文化，这些甜点通常包含各种奶制品、果干、坚果和糖浆等成分；而对于其它风味的美食，这可能是指一些特殊口味或者结合多种味道的菜品，如酸辣、咖喱、咖啡等。这些菜品可能具有独特的风味组合。无论是素食还是非素食，风味最多的都是辛辣口味。素食有许多甜味口味的美食，而非素食口味却没有。
3. 通过分析各菜肴类型特色，可以知道，主菜通常是印度餐中的主要部分，提供了大量的蛋白质、碳水化合物和蔬菜；印度的甜点种类繁多，这85种甜点可能包括了各种饼干、蛋糕、布丁、凝胶、奶昔和传统的印度甜点，如古拉和拉斯马利。这些甜点通常会使用大量的糖、牛奶和香料，口感丰富；对于小吃，通常是较轻的食物，可以在任何时间食用。可能包括炸食、烤食、沙拉和各种小点心；开胃菜通常是在正餐开始前食用的，以激发食欲，所以种类不太多。无论是素食还是非素食，类型最多的都是主食，素食有85种美食是甜点，而非素食却没有这种类型的美食，说明印度非素食美食是不吃甜的。
4. 通过分析印度美食食材使用，糖的使用量最大，48种美食都使用了糖，这可能是因为甜味在印度美食中扮演着重要角色。印度有多种糖类，如jaggery（一种未精炼的棕糖），它们在许多传统食谱中都有使用；姜的使用量也很大，这可能是因为姜不仅为食物增添了风味，还有一定的药用价值。在印度料理中，姜常被用来制作腌料、酱汁和咖喱；加拉姆香料是一种常见的印度混合香料，通常包括丁香、肉桂、孜然、豆蔻等香料。它为印度菜肴提供了复杂的香气和味道；与此同时，我们还发现有各种混合的面粉使用的也比较多，印度的很多美食都含有或多或少的各种面粉。素食食材中，使用量较大的是sugar，ghee，urad dal等，而在48种使用了糖的美食中，有47种都是素食美食，酥油（ghee）也都是用在了素食上；非素食食材中，使用量较大的与素食食材使用大相径庭，使用量较大的是mustard oil（芥末油），ginger（姜），chicken等。
5. 通过分析印度各地区菜肴的特色，西部美食最多，有74种，其次是南部地区，有59种，美食种类最少的是中部地区，只有3种美食；素食美食与上述各地区美食分布类似，而非素食美食种类最多的是东北地区，有13种美食；辛辣风味美食最多的是印度西部地区，北部地区和南部地区也较多；其他口味美食最多的是南部地区，其次是西部地区；甜味口味美食最多的是西部地区，其次是东部地区；苦味口味美食较少，只有北部地区和西部地区才有，且都只有两种美食；酸味口味美食只有在西部地区才有，而且只有一种美食；甜点类型美食最多的是印度西部地区，东部地区和南部地区也较多；主食类型美食最多的是北部地区，其次是南部地区、西部地区；小吃类型美食最多的是西部地区，其次是南部地区；开胃菜美食较少，只有北部地区。
6. 通过分析印度各州菜肴的特色，可以看到，美食种类较多的有吉吉拉特、旁遮普和马哈拉施特拉，中部的中央邦、切蒂斯格尔等州美食种类较少；素食美食较多的州有吉吉拉特、旁遮普和马哈拉施特拉；非素食美食种类较多的州有阿萨姆、西孟加拉以及旁遮普；辛辣风味美食最多的州是旁遮普，吉吉拉特的美食也较多；甜味口味美食最多的是西孟加拉，其次是马哈拉施特拉；其他口味美食最多的是阿萨姆、马哈拉施特拉和泰米尔纳德；苦味口味美食较少，只有旁遮普和吉吉拉特才有，且都只有两种美食；酸味口味美食只有在吉吉拉特才有，而且只有一种美食；主食类型美食最多的州是旁遮普，阿萨姆和泰米尔纳德也较多；甜点类型美食最多的是西孟加拉，其次是马哈拉施特拉；小吃类型美食最多的是吉吉拉特，其次是马哈拉施特拉；开胃菜美食较少，只有旁遮普才有。
7. 通过相似性分析, 可以看到，这是一个255\*255的相似度热力图，索引值为0~63的相似度较高，说明这些美食之间的相似性较高。索引为0~63的美食与64~255的美食之间有少部分相似性较高，而64~255的美食之间的相似度大部分都比较一般。

**结论和建议**：

1. 根据各地区、各州的素食和非素食美食特色数据分析结果，虽然印度当地美食十分丰富,但是绝大多数都是素食美食，印度大部分地区因为宗教信仰，所以非素食美食十分稀少。如果游客想要前往印度旅游,那么建议游客在此之前需要考虑当地的美食类型是否合口，如果是想要品尝印度肉食美食，我们并不建议游客前往印度，若还是想要去的话，我们推荐去印度的阿萨姆邦和西孟加拉邦；若游客十分喜爱素食美食，我们强烈推荐去印度的北部地区旁遮普邦、西部地区吉吉拉特以及马哈拉施特拉邦。
2. 根据印度美食食材及各类型，各风味数据分析结果，印度255种美食总共使用了365种食材，其中素食美食使用了321种食材，非素食美食使用了96种食材。在所有美食使用的食材中，糖使用的最多，而这些使用了糖的美食绝大多数都是素食，它们在许多传统食谱中都有使用，从所有美食的风味分布、类型分布中可以看出，甜味风味美食占比较高，甜点类型美食占比也较高，表明印度当地的甜味、甜点美食众多，喜欢吃甜味素食的游客可以前往印度的东北地区的西孟加拉邦以及西部地区马哈拉施特拉邦。姜的使用量也比较高，姜是一种辣味调料，而无论是素食还是非素食，印度美食风味中，占比最高的都是辛辣风味美食，那么我们推荐喜爱吃辛辣口味美食的游客一定不能错过。
3. 根据相似性分析及开发的美食推荐系统结果，如果游客在印度当地品尝了一款口味不错或者口感糟糕的美食，又想要再找一种与之相似的美食，继续品尝或者避免再次踩坑，那么一定要使用我们开发的美食推荐系统，它会根据两美食的余弦相似度计算出两者的相似性，越接近1，说明越相似。游客可以根据已知一种美食，来寻找其它254种美食中哪种与其最相似，也可以根据已知两种美食计算两者相似性。与此同时还有其它四种功能：根据已知美食寻找该美食所在州、根据想要吃的美食类型或者风味，推荐去哪些州以及根据所在州，推荐美食类型或风味。