第十二章

1. 实践案例

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from matplotlib.patches import Ellipse

from sklearn.datasets import load\_iris

from sklearn.decomposition import PCA

from sklearn.neighbors import KernelDensity

from scipy import stats

iris = load\_iris()

df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

df['species'] = pd.Categorical.from\_codes(iris.target, iris.target\_names)

print(df.head())

# 2. PCA 降维

pca = PCA(n\_components=2)

principal\_components = pca.fit\_transform(df.iloc[:, :-1])

dat\_pca = pd.DataFrame(data=principal\_components, columns=['Dim.1', 'Dim.2'])

dat\_pca['sample2'] = df['species']

print(dat\_pca.head())

# 3. 创建绘图

gridspec\_kw = {'height\_ratios': [1, 2],'width\_ratios':[3,2]}

fig, axs = plt.subplots(2, 2,figsize=(12,8), gridspec\_kw=gridspec\_kw)

palette = {"setosa": "#FFA07A", "versicolor": "#FFB6C1", "virginica": "#41b6c4"}

# 4. 绘制概率密度函数图

axs[0, 0].clear()

sns.kdeplot(data=dat\_pca, x='Dim.1', hue='sample2', fill=True, alpha=0.6, palette=palette, linewidth=0, ax=axs[0, 0],common\_norm=True)

axs[0,0].set\_xlabel("")

axs[0,0].set\_title("Probability Density Function of PCA Dim.1", fontsize=10)

axs[0,0].set\_xlim(dat\_pca['Dim.1'].min(), dat\_pca['Dim.1'].max())

axs[0,0].set\_ylim(0, 0.7)

axs[0,0].axhline(0, color='black', lw=0.5)

#5.绘制概率密度等高线

X\_pca = pca.fit\_transform(iris.data)

y = iris.target

kde\_models = []

for i in range(len(np.unique(y))):

kde = KernelDensity(bandwidth=0.5).fit(X\_pca[y == i])

kde\_models.append(kde)

x\_min, x\_max = X\_pca[:, 0].min() - 1, X\_pca[:, 0].max() + 1

y\_min, y\_max = X\_pca[:, 1].min() - 1, X\_pca[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, 0.1),

np.arange(y\_min, y\_max, 0.1))

grid\_points = np.vstack([xx.ravel(), yy.ravel()]).T

for i, (label,color) in enumerate(palette.items()):

kde = kde\_models[i]

log\_density = kde.score\_samples(grid\_points)

density = np.exp(log\_density).reshape(xx.shape)

contour = axs[0,1].contour(xx, yy, density, levels=[0.01, 0.05, 0.1],

colors=color, alpha=0.5)

axs[0,1].clabel(contour, inline=True, fontsize=8, fmt='%.2f', colors=color)

axs[0,1].axhline(0, color='grey', lw=0.8, ls='--')

axs[0,1].axvline(0, color='grey', lw=0.8, ls='--')

axs[0,1].set\_xlabel('PC1', fontsize=8)

axs[0,1].set\_ylabel('PC2', fontsize=8)

axs[0,1].set\_title('Iris Dataset PCA Contour Plot', fontsize=10)

axs[0,1].grid(True)

#6.绘制散点图

sns.scatterplot(data=dat\_pca, x='Dim.1', y='Dim.2', hue='sample2', palette=palette,

style="sample2", s=100, edgecolor="#000000",ax=axs[1,0])

axs[1,0].axvline(x=0, color="#708090", linestyle='dashed')

axs[1,0].axhline(y=0, color="#708090", linestyle='dashed')

def draw\_ellipse(ax, mean, cov, color):

eigenvalues, eigenvectors = np.linalg.eig(cov)

width, height = 4\* np.sqrt(eigenvalues)

angle = np.degrees(np.arctan2(\*eigenvectors[:, 0][::-1]))

ellipse = Ellipse(xy=mean, width=width, height=height, angle=angle, color=color, alpha=0.2)

axs[1,0].add\_patch(ellipse)

for species in dat\_pca['sample2'].unique():

subset = dat\_pca[dat\_pca['sample2'] == species]

mean = subset[['Dim.1', 'Dim.2']].mean().values

cov = np.cov(subset[['Dim.1', 'Dim.2']], rowvar=False)

draw\_ellipse(plt.gca(), mean, cov, color=palette[species])

axs[1,0].set\_title("PCA of Iris Dataset", fontsize=10, color="#000000")

axs[1,0].set\_xlabel(f"PC1 ({round(pca.explained\_variance\_ratio\_[0] \* 100, 1)}%)", fontsize=8)

axs[1,0].set\_ylabel(f"PC2 ({round(pca.explained\_variance\_ratio\_[1] \* 100, 1)}%)", fontsize=8)

axs[1,0].legend(fontsize=8,loc='lower right')

axs[1,0].grid(True)

#7.绘制箱线图

boxplot = sns.boxplot(x='sample2', y='Dim.1', data=dat\_pca, hue='sample2', palette=palette,ax=axs[1,1])

sns.stripplot(x='sample2', y='Dim.1', data=dat\_pca,

hue='sample2', palette=palette, marker='o', alpha=0.6, size=5)

def add\_stat\_annotation(ax, comparisons):

categories = dat\_pca['sample2'].cat.categories.tolist()

for cat1, cat2 in comparisons:

x1 = categories.index(cat1)

x2 = categories.index(cat2)

y = dat\_pca['Dim.1'].max() + 0.2

group1 = dat\_pca[dat\_pca['sample2'] == cat1]['Dim.1']

group2 = dat\_pca[dat\_pca['sample2'] == cat2]['Dim.1']

t\_stat, p\_val = stats.ttest\_ind(group1, group2, equal\_var=False)

axs[1,1].text((x1 + x2) / 2, y, f"p = {p\_val:.3f}", ha='center',fontsize=8)

my\_comparisons = [('setosa', 'versicolor'), ('versicolor', 'virginica'), ('setosa', 'virginica')]

add\_stat\_annotation(boxplot, my\_comparisons)

axs[1,1].set\_title("Boxplot of PCA Dim.1 by Species", fontsize=10)

axs[1,1].set\_xlabel("")

axs[1,1].set\_ylabel("")

axs[1,1].set\_ylim(-3.5, 4.5)

plt.tight\_layout()

plt.show()

2. 实验一

import matplotlib.pyplot as plt

import geopandas as gpd

import pandas as pd

from matplotlib.ticker import FuncFormatter

# 合并地图数据和气温数据

data = pd.read\_excel(r"C:\xxx\2020四川省平均气温.xlsx")

print(data.head())

data['quarter'] = (data['month'] - 1) // 3 + 1

quarter\_temp= data.groupby(['city', 'quarter'])['temp'].mean().reset\_index()

print(quarter\_temp.head())

map\_data = gpd.read\_file(r"C:\Users\丁月甜\Downloads\四川省 (1).json")

map\_data = map\_data.merge(quarter\_temp, left\_on='name', right\_on='city')

print(map\_data.head())

print(map\_data.describe())

# 创建图形和子图

fig, axs = plt.subplots(2, 2, figsize=(10, 8))

axs = axs.flatten()

vmin = map\_data['temp'].min()

vmax = map\_data['temp'].max()

quarters = ['第一季度', '第二季度', '第三季度', '第四季度']

plt.rcParams['font.family'] = 'SimHei'

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

plt.rcParams['axes.unicode\_minus'] = False

def format\_longitude(x, pos):

return f'{int(x)}°E'

def format\_latitude(y, pos):

return f'{int(y)}°N'

for i, quarter in enumerate(range(1, 5)):

temp\_data = map\_data[map\_data['quarter'] == quarter]

temp\_data.plot(column='temp', ax=axs[i], legend=False,

cmap='coolwarm', vmin=vmin, vmax=vmax)

axs[i].xaxis.set\_major\_formatter(FuncFormatter(format\_longitude))

axs[i].yaxis.set\_major\_formatter(FuncFormatter(format\_latitude))

for x, y, city in zip(temp\_data.geometry.centroid.x,

temp\_data.geometry.centroid.y,

temp\_data['city']):

axs[i].text(x, y, city, fontsize=7, ha='center', color='black')

axs[i].set\_title(f'{quarters[i]}平均气温', fontsize=10)

sm = plt.cm.ScalarMappable(cmap='coolwarm', norm=plt.Normalize(vmin=vmin, vmax=vmax))

sm.set\_array([])

cbar\_ax = fig.add\_axes([0.92, 0.15, 0.02, 0.7])

cbar = fig.colorbar(sm, ax=axs, cax=cbar\_ax)

cbar.set\_label("平均气温(°C)")

plt.tight\_layout(rect=[0, 0, 0.9, 1])

plt.show()

3. 实验二

import re

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfVectorizer

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

from gensim import corpora

from gensim.models import LdaModel

from textblob import TextBlob

from wordcloud import WordCloud

from bs4 import BeautifulSoup

#一、数据预处理

#1.文本预处理函数

def preprocess\_text(text):

text = BeautifulSoup(text, "html.parser").get\_text() # 去除HTML标签

text = re.sub(r'[^a-zA-Z\s]', '', text) # 去除特殊字符

text = text.lower() # 将所有字母转换为小写

stop\_words = set(stopwords.words('english')) # 删除停用词

text = ' '.join(word for word in text.split() if word not in stop\_words)

lemmatizer = WordNetLemmatizer() # 词形还原

text = ' '.join(lemmatizer.lemmatize(word) for word in text.split())

return text

#2.应用CSV文本文件

def load\_data(file\_path):

return pd.read\_csv(file\_path)

def preprocess\_csv(file\_path):

df = load\_data(file\_path)

df['processed\_text'] = df['text'].apply(preprocess\_text)

return df

if \_\_name\_\_ == "\_\_main\_\_":

file\_path = r"C:\Users\丁月甜\Desktop\wikipedia\_articles.csv"

output\_file\_path=r"C:\Users\丁月甜\Desktop\1processed\_wikipedia\_articles.csv"

processed\_df = preprocess\_csv(file\_path)

processed\_df.to\_csv(output\_file\_path, index=False, encoding='utf-8-sig')

df=pd.read\_csv(r"C:\Users\丁月甜\Desktop\1processed\_wikipedia\_articles.csv")

print(df.head())

#二、LDA主题分类并绘制词云图

#1.使用TF-IDF向量化

vectorizer = TfidfVectorizer(stop\_words='english')

tfidf\_matrix = vectorizer.fit\_transform(df['processed\_text'])

#2.将TF-IDF矩阵转换为稀疏矩阵

sparse\_matrix = tfidf\_matrix.toarray()

#3.创建词典和语料库

dictionary = corpora.Dictionary([vectorizer.get\_feature\_names\_out()])

corpus = [dictionary.doc2bow(text) for text in df['processed\_text'].apply(lambda x: x.split())]

#4.应用LDA模型

num\_topics = 5

lda\_model = LdaModel(corpus, num\_topics=num\_topics, id2word=dictionary, passes=15,random\_state=10)

#5.获取每个文档的主题分布

topic\_distribution = lda\_model.get\_document\_topics(corpus)

df['Topic'] = [max(dist, key=lambda x: x[1])[0] for dist in topic\_distribution]

#6.从 LDA 模型中提取主题关键词并自定义主题名称

topics = lda\_model.print\_topics(num\_words=5)

print(topics)

topic\_names = {i: f"Topic {i}" for i in range(num\_topics)}

for i, topic in enumerate(topics):

print(f"主题 {i}: {topic[1]}")

# 修改条件判断

if "state" in topic[1] and "university" in topic[1]:

topic\_names[i] = "Politics and Global Affairs"

elif "game" in topic[1] and "century" in topic[1]:

topic\_names[i] = "Sports and Entertainment"

elif "displaystyle" in topic[1] and "study" in topic[1]:

topic\_names[i] = "Science and Technology"

elif "philosophy" in topic[1] and "theory" in topic[1]:

topic\_names[i] = "Philosophy and Theory"

elif "water" in topic[1] and "energy" in topic[1]:

topic\_names[i] = "Health and Environmental Studies"

df['Topic\_name'] = df['Topic'].map(topic\_names)

#6.绘制词云图

def plot\_wordcloud(lda\_model, topic\_names):

for i in range(len(topic\_names)):

plt.figure(figsize=(10, 5))

wordcloud = WordCloud(width=800, height=400, background\_color='white').generate\_from\_frequencies(dict(lda\_model.show\_topic(i, topn=50)))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis('off')

plt.title(topic\_names[i],fontsize=20)

plt.show()

plot\_wordcloud(lda\_model, topic\_names)

#三、情感分析并绘制柱状堆积图、雷达图

#1.基于主题分类结果进行情感分析

def analyze\_sentiment(text):

analysis = TextBlob(text)

sentiment\_value = analysis.sentiment.polarity

if sentiment\_value > 0:

sentiment\_label = 'positive'

elif sentiment\_value < 0:

sentiment\_label = 'negative'

else:

sentiment\_label = 'neutral'

return sentiment\_value, sentiment\_label

df[['Sentiment\_Value', 'Sentiment']] = df['processed\_text'].apply(lambda x: analyze\_sentiment(x)).apply(pd.Series)

#2.绘制柱状堆积图

custom\_colors = ['#ADD8E6','#F7DC6F','#FFB6C1']

topic\_sentiment\_counts = df.groupby(['Topic', 'Sentiment']).size().unstack(fill\_value=0)

topic\_sentiment\_counts.index = [topic\_names[i] for i in topic\_sentiment\_counts.index]

ax = topic\_sentiment\_counts.plot(kind='bar', stacked=True,

color=[custom\_colors[i % len(custom\_colors)] for i in range(len(topic\_sentiment\_counts))])

plt.title('Sentiment Distribution by Topic')

plt.xlabel('Topic')

plt.ylabel('Number of Articles')

plt.legend(title='Sentiment')

plt.xticks(rotation=0,fontsize=8)

plt.tight\_layout()

plt.show()

#3.绘制雷达图

#3.1准备雷达图数据

sentiment\_means = df.groupby(['Topic\_name', 'Sentiment'])['Sentiment\_Value'].mean().unstack(fill\_value=0)

sentiments = sentiment\_means.columns

num\_vars = sentiment\_means.index.size

colors = {

'positive': 'green',

'negative': 'orange',

'neutral': 'red'

}

#3.2设置雷达图角度

angles = np.linspace(0, 2 \* np.pi, num\_vars, endpoint=False).tolist()

angles += angles[:1]

#3.3绘制雷达图

fig, ax = plt.subplots(figsize=(10, 6), subplot\_kw=dict(polar=True))

for sentiment in sentiments:

values = sentiment\_means[sentiment].values.flatten().tolist()

values += values[:1]

ax.fill(angles, values, color=colors[sentiment], alpha=0.1, label=sentiment)

ax.plot(angles, values, color=colors[sentiment], linewidth=1, alpha=0.3)

for v in values:

if v > 0:

for i in range(num\_vars):

size = values[i]\* 200

ax.scatter(angles[i], values[i], s=size, color=colors[sentiment], alpha=0.6)

elif v==0:

for i in range(num\_vars):

size = 20

ax.scatter(angles[i], values[i], s=size, color=colors[sentiment], alpha=0.6,marker='\*')

elif v<0:

for i in range(num\_vars):

size = abs(values[i])\* 200

ax.scatter(angles[i], values[i], s=size, color=colors[sentiment], alpha=0.6,marker='^')

#3.4设置雷达图的标签

y\_tick\_values = np.linspace(-0.2, 0.1, num=4)

ax.set\_yticks(y\_tick\_values)

ax.tick\_params(axis='y', colors='#4A235A', labelsize=12)

ax.set\_yticklabels([])

for y\_tick in y\_tick\_values:

angle = 0

ax.text(angle, y\_tick , f"{round(y\_tick,2)}", color='black', ha='center', va='center', fontsize=10)

ax.axvline(x=0, color='#4A235A', linewidth=3, alpha=0.5)

ax.set\_xticks(angles[:-1])

ax.set\_xticklabels(sentiment\_means.index)

#3.5添加图例

handles = [plt.Line2D([0], [0], color=colors[t], lw=8, alpha=0.1) for t in sentiments]

labels = list(sentiments)

for y\_val in y\_tick\_values:

scatter\_size = abs(y\_val) \* 200

if round(y\_val, 2) > 0:

handles.append(plt.Line2D([0], [0], marker='o', markersize=scatter\_size/3, color='w', markerfacecolor='gray', alpha=0.8))

labels.append(f'Value {round(y\_val, 2)}')

elif round(y\_val, 2)==0:

handles.append(plt.Line2D([0], [0], marker='\*', markersize=20/3, color='w', markerfacecolor='gray', alpha=0.8))

labels.append('Value 0')

elif round(y\_val, 2)<0:

handles.append(plt.Line2D([0], [0], marker='^', markersize=scatter\_size/3, color='w', markerfacecolor='gray', alpha=0.8))

labels.append(f'Value {round(y\_val, 2)}')

ax.legend(handles=handles, labels=labels, loc='upper right', bbox\_to\_anchor=(1.4, 1), fontsize=9, title='Topic and Sentiment')

plt.title('Sentiment Analysis Radar Chart',fontsize=15)

plt.show()