橄榄油质量特性关系分析

——基于多元统计方法的多维研究

陈子熠　2022010503

摘要

本研究采用多种多元统计分析方法，系统性探讨了橄榄油的物理与化学特性之间的关系。通过主成分分析和聚类分析，识别出不同特性的橄榄油类别，并通过典型相关分析建立了橄榄油质量评估模型。结果表明，橄榄油的视觉特征如颜色和光泽度与其物理特性如浓稠度具有显著相关性；氧化指标与酸度等化学特性之间也存在明显的线性关联；能够根据物理特性对总体化学特性进行预测。这为消费者提供了科学的选购依据，同时为生产者提供了全面的质量控制策略。

1. 研究背景

橄榄油因其丰富的营养价值和健康益处，在全球范围内广受欢迎。橄榄油的质量受到多种物理和化学特性的影响，这些特性不仅决定了橄榄油的感官品质和营养价值，还对其保质期和市场竞争力有重要影响。传统上，橄榄油质量评价主要依赖于酸度、过氧化值等，但这些化学特性在测量仪器受限的情况下难以直接得到，消费者在选购橄榄油时通常仅依赖于外观、色泽等可视和感官特性。本研究旨在通过多维数据分析方法，系统性地探讨橄榄油质量特性之间的关系，从而为消费者提供更科学的选购依据，同时为生产者提供更全面的质量控制策略。

2. 研究问题

本研究旨在通过多维数据分析方法，系统性地探讨橄榄油质量特性之间的关系，以期为橄榄油的选购与质量控制提供科学依据。具体研究问题如下：

1. 视觉特征 vs 浓稠度：如何按橄榄油的颜色、透光度等视觉特征对橄榄油进行分类？视觉特征如何影响浓稠度等其它物理特征？
2. 氧化程度 vs 酸度：如何按氧化程度对橄榄油进行分类？橄榄油的氧化程度与酸度等化学性质之间有何种关联？
3. 物理性质 vs 化学性质：如何按总体物理性质或化学性质对橄榄油进行分类？物理性质与化学性质之间存在何种内在联系？

3. 数据介绍

3.1 变量说明

表 1　oliveoil 数据集变量说明

特征变量	名称	标签	释义
ID	样品编号	/	表示每个橄榄油样本的唯一标识符，用于区分不同的样本。
yellow	黄色度	physics, visual, color	表示橄榄油的黄色色调强度，通常通过色度计测量。黄色度是橄榄油外观的重要指标，反映其成熟度和品质。
green	绿色度	physics, visual, color	表示橄榄油的绿色色调强度，同样通过色度计测量。绿色度主要由橄榄中的叶绿素含量决定，通常与新鲜度和加工方式相关。
brown	棕色度	physics, visual, color	表示橄榄油的棕色色调强度，通过色度计测量。棕色度可能由氧化和其他化学反应引起，反映了橄榄油的储存和加工条件。
glossy	光泽度	physics, visual, light	表示橄榄油的光泽度，可能通过视觉评价或光度计测量。光泽度是橄榄油感官质量的重要指标之一，影响消费者的购买决策。
transp	透明度	physics, visual, light	表示橄榄油的透明度。透明度是橄榄油纯净度的表现，受悬浮颗粒和沉淀物的影响。
syrup	浓稠度	physics, viscosity	表示橄榄油的浓稠度。浓稠度影响橄榄油的流动性和使用体验，通常由脂肪酸组成和温度决定。
Acidity	酸度	chemistry, acid	表示橄榄油的游离脂肪酸含量，通常以百分比表示。酸度是橄榄油质量的重要化学指标，反映其新鲜度和加工处理的良好程度。
Peroxide	过氧化值	chemistry, oxidation	表示橄榄油中的过氧化物含量，衡量初级氧化程度。过氧化值是检测橄榄油氧化状态和保质期的关键指标。
K232	K232 值	chemistry, oxidation	表示橄榄油在 232 纳米波长下的吸光度，反映初期氧化产物的含量。K232 值用于评估橄榄油的新鲜度和氧化程度。
K270	K270 值	chemistry, oxidation	表示橄榄油在 270 纳米波长下的吸光度，反映次级氧化产物的含量。K270 值通常用于检测橄榄油的老化程度和次级氧化产物。
DK	ΔK 值	chemistry, oxidation	表示 K232 和 K270 之间的差值，用于评估橄榄油的氧化状态。ΔK 值可帮助检测橄榄油的掺假情况和综合氧化水平。

3.2 探索型数据分析

3.2.1 数据集基本信息及缺失值检查

该数据集包含 16 个橄榄油样本和 12 个特征变量（包括样本编号 ID 和 11 个数值特征）。所有特征都没有缺失值，且数据类型为浮点数（float64）。

3.2.2 数据描述性统计

	yellow	green	brown	glossy	transp	syrup	Acidity	Peroxide	K232	K270	DK
count	16.000000	16.000000	16.000000	16.00000	16.000000	16.000000	16.000000	16.000000	16.000000	16.000000	16.000000
mean	50.875000	33.512500	12.331250	80.81250	78.193750	47.975000	0.311875	13.252500	1.708250	0.118144	-0.001750
std	19.458623	23.486986	5.128706	6.18804	8.307384	3.065398	0.176568	3.345141	0.248731	0.023707	0.002236
min	21.400000	9.700000	8.000000	67.70000	63.500000	42.300000	0.150000	8.140000	1.331000	0.085000	-0.005000
25%	32.075000	12.075000	10.025000	77.80000	74.175000	46.150000	0.190000	10.950000	1.536000	0.101500	-0.003250
50%	52.800000	31.150000	10.800000	80.40000	77.200000	47.500000	0.260000	12.400000	1.653500	0.116000	-0.002000
75%	68.800000	54.700000	11.975000	85.37500	84.875000	50.650000	0.312500	15.375000	1.893250	0.128500	0.000000
max	73.500000	73.400000	28.400000	89.90000	89.700000	52.800000	0.730000	19.400000	2.222000	0.168000	0.003000

图 1　oliveoil 数据集基本特征

如图 1，属于物理性质的特征变量数值明显大于属于化学性质的特征变量。即使对于相同标签的特征变量，如 yellow, green 与 brown，其均值与标准差也存在明显的区别，因此需要对数据进行标准化处理。

由于特征变量均为连续数值，且数值的大小具备明显的实际含义，因此无需对原始变量采用 one-hot 编码或周期性特征编码。

3.2.3 数据分布

如图 2，从数据分布图可以看出，各个变量的分布存在一定的偏态，并非正态分布。这可能源于橄榄油的生产和处理过程中存在的自然变异，以及不同橄榄油样本在颜色、酸度、氧化程度等方面的天然差异。在后续的数据分析和建模过程中，可能需要使用鲁棒的统计方法，避免采用过于依赖正态性假设的统计方法，以减少偏态对分析结果的影响。

3.2.4 数据关联

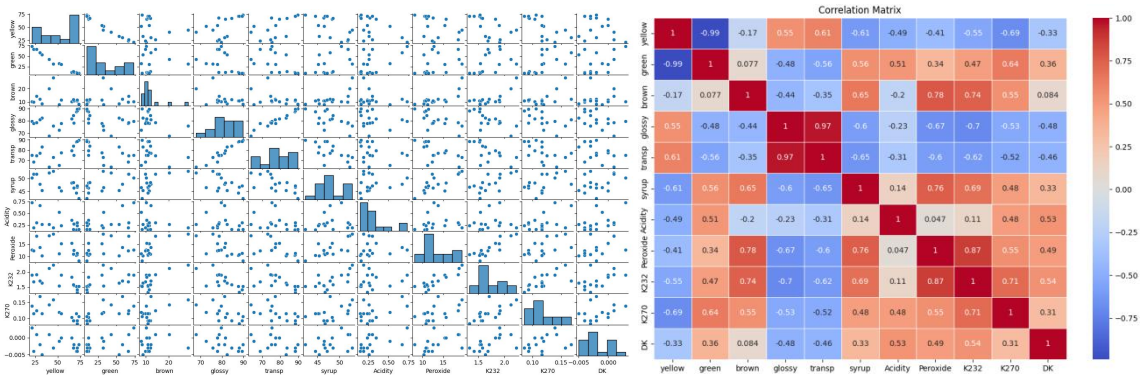

图 2/3　oliveoil 数据集数据集 pair-plot

如图 2/3，部分变量之间存在较强的线性相关性，如 yellow 与 green、glossy 与 transp、Peroxide 与 K232。这为对数据进行降维提供了依据。此外，可以发现橄榄油的某些物理特性（如棕色度、浓稠度）与其化学特性（如氧化程度）密切相关。这为之后研究橄榄油性质之间的关联性提供了初步的参考。同时，在后续的分析和建模过程中，需要特别关注这些强相关性，避免多重共线性问题，同时可以利用这些相关性进行模型优化。

4. 研究结果

4.1 氧化程度 vs 酸度

4.1.1 主成分分析

氧化程度的衡量指标包括：过氧化值、K232、K270 与 ΔK；酸度衡量指标包括酸度。在 EDA 中观察到过氧化值、K232、K270 等氧化程度衡量指标间存在较强的线性相关性，故可以尝试 PCA 方法对其进行降维，提取关键成分。对于酸度，由于仅存在一个衡量指标，故无需进行降维。

4.1.1.1 主成分个数的选择

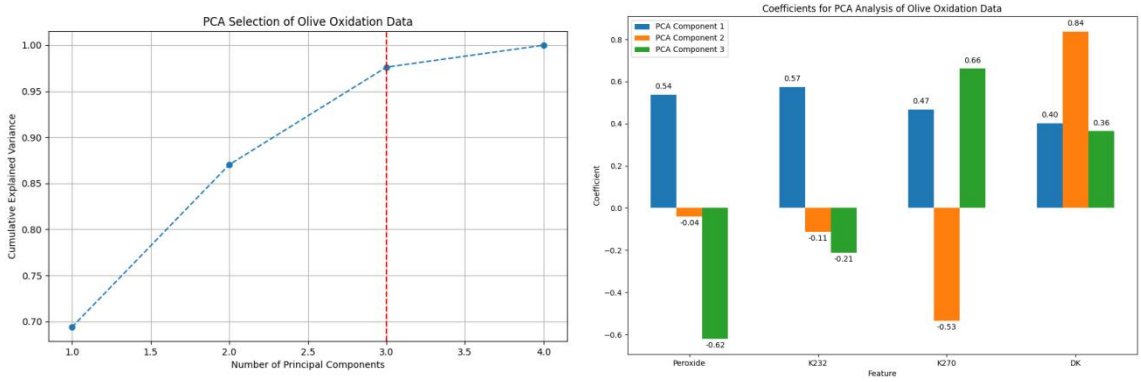

图 3/4　橄榄油氧化程度指标主成分分析

如图 3，前三个主成分能够解释绝大部分的方差，故选取主成分个数为 3。

4.1.1.2 主成分分析结果

如图 4，观察各个主成分对应特征的系数，可以发现：

1. 第一主成分主要代表橄榄油**总体的氧化特性**。该主成分的高值意味着橄榄油的过氧化值、K232、K270 和ΔK 值较高，反映了橄榄油氧化程度较高。
2. 第二主成分主要区分了 K270 和ΔK 的差异。负系数大的 K270 和正系数大的ΔK 表示这个主成分综合考虑了 K270 与 K232 之间的差异和 K270 的绝对值的影响，衡量**次级氧化物相对初级氧化物的含量**。
3. 第三主成分进一步区分了初级氧化物与次级氧化物的差异。负系数大的 Peroxide 和正系数大的 K270 衡量了**次级氧化物与初级氧化物含量的差值**，其中初级氧化物的指标以 Peroxide 为主.，次级氧化物指标以 K270 为主。

据此，为了便于理解主成分的实际意义，可以写作：

$$\mathrm{PCA1} = \mathrm{Peroxide} + \mathrm{K232} + \mathrm{K270} + \mathrm{DK}$$

$$\mathrm{PCA2} = \frac{DK}{K270} \times 100\%$$

$$\mathrm{PCA3} = \mathrm{K270} - \mathrm{Peroxide}$$

4.1.2 聚类分析

根据橄榄油的氧化状态将样本分成不同的组。采用 K-means 方法对氧化程度指标进行聚类，可以识别出橄榄油在不同氧化程度下的分布情况。

对于酸度，由于仅存在一个衡量指标，根据与中心点的距离可直接聚类，故不详细讨论。

4.1.2.1 聚类个数的选择

采用肘部法则选择合适的 K 值。如图 5，可见肘部点位于 K=4。

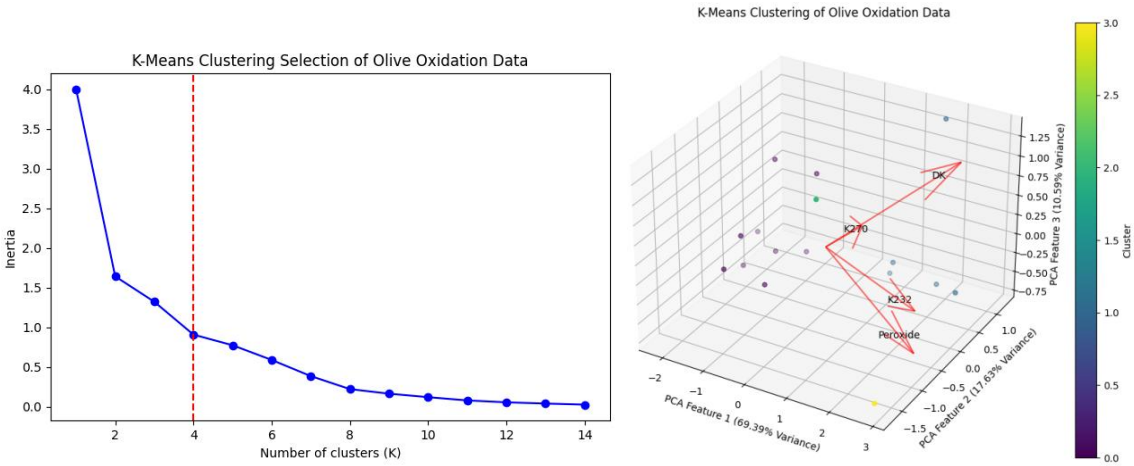

图 5/6　橄榄油按氧化程度指标聚类

4.1.2.2 K-means 聚类分析结果

利用 PCA 指标与 K-means 聚类结果对样本按氧化程度进行分类，并可视化，如图 6。据此，可以将样本按氧化程度分为 4 类：

1. 紫色，低 PCA1，代表总体氧化程度较低的一类橄榄油。
2. 灰色，高 PCA1，高 PCA2，可能代表总体氧化程度较高，且以次级氧化物为主的橄榄油。
3. 黄色，高 PCA1，低 PCA2，低 PCA3，可能代表总体氧化程度较高，且以初级氧化物为主的一类橄榄油，此类样本较少。
4. 绿色，低 PCA2，高 PCA3，可能代表用 K232 衡量的初级氧化物产量和次级氧化物相当，但用 Peroxide 衡量的初级氧化物产量较低的一类橄榄油。这可能是由于两种不同测量初级氧化物的方法存在误差所致。

4.1.3 典型相关分析

利用提取到的衡量氧化程度的主成分，与衡量酸度的指标进行典型相关分析，得到如下载荷与典型相关系数：

```
Canonical component 1 X loadings: [0.39425505 0.25682668 0.88238484]
Canonical component 1 Y loadings: [1.]
First canonical correlation: 0.7974425660817311
```

根据典型相关载荷可知，氧化物含量与酸度呈正相关，且次级氧化物产量在相关性中起主导作用。根据典型相关系数和典型相关分析效果图（图 7），两个典型相关变量之间存在很好的线性关系。据此，可以根据得到的典型相关关系可以建立过原点的线性回归模型，进而通过氧化物含量对酸度进行预测。

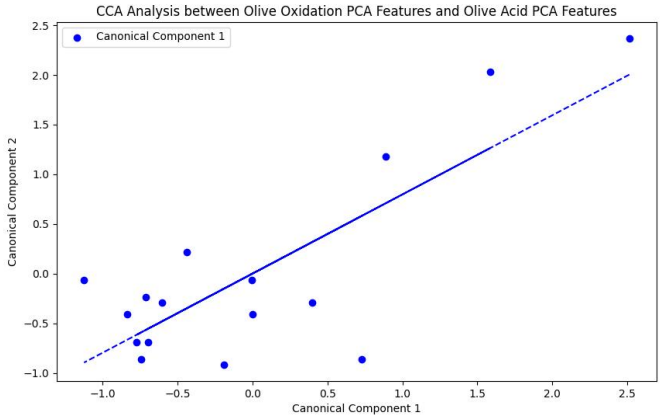

图 7　橄榄油氧化程度-酸度典型相关分析

4.2 视觉特征 vs 浓稠度

4.2.1 主成分分析

视觉特征的衡量指标包括黄色度、绿色度、棕色度、光泽度与透明度；浓稠度衡量指标包括浓稠度。在 EDA 中观察到黄色度与绿色度、光泽度与透明度等视觉衡量指标间存在较强的线性相关性，故可以尝试 PCA 方法对其进行降维，提取关键成分。对于浓稠度，由于仅存在一个衡量指标，故无需进行降维。

4.2.1.1 主成分个数的选择

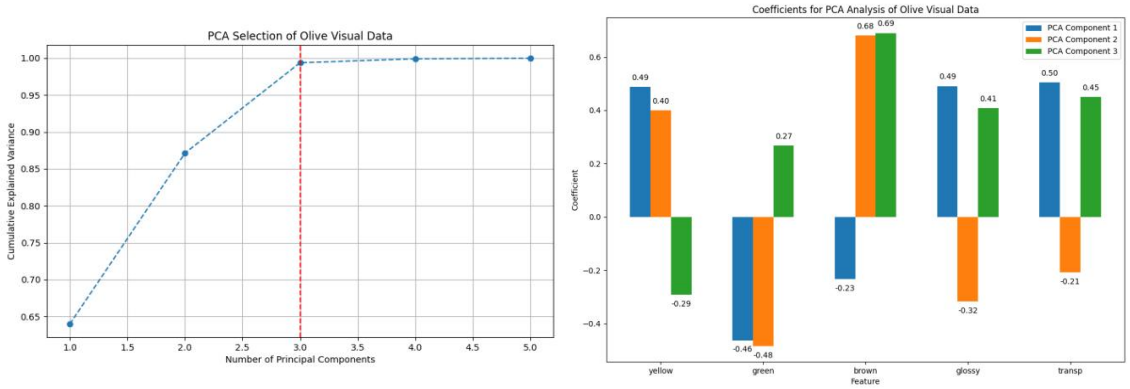

图 8/9　橄榄油视觉特征指标主成分分析

如图 8，前三个主成分能够解释绝大部分的方差，故选取主成分个数为 3。

4.2.1.2 主成分分析结果

如图 9，观察各个主成分对应特征的系数，可以发现：

1. 第一主成分主要代表橄榄油的黄色度、光泽度和透明度。该主成分的高值通常意味着较好的感官品质。
2. 第二主成分主要代表黄色度和棕色度，并与绿色度呈负相关，可能与橄榄油成熟度有关。
3. 第三主成分主要与棕色度、光泽度、透明度正相关，并与黄色度负相关，可能反映出一种特殊的橄榄油特性或处理工艺。

4.2.2 聚类分析

根据橄榄油的视觉特征将样本分成不同的组。采用 K-means 方法对氧化程度指标进行聚类，可以识别出橄榄油在不同视觉特征下的分布情况。

对于浓稠度，由于仅存在一个衡量指标，根据与中心点的距离可直接聚类，故不详细讨论。

4.2.2.1 聚类个数的选择

如图 10，采用肘部法则选择合适的 K 值。如图，可见肘部点位于 K=3。

4.2.2.2 K-means 聚类分析结果

利用 PCA 指标与 K-means 聚类结果对样本按视觉特征进行分类，并可视化，如图 11。据此，可以将样本按视觉特征分为 3 类：

1. 紫色，具有高绿色度，代表成熟度较低的一类橄榄油。
2. 绿色，具有较高的黄色度、透明度与光泽度，代表具有较好视觉品质的一类橄榄球。
3. 黄色，具有高棕色度，可能反映出一种特殊的橄榄油特性或处理工艺。

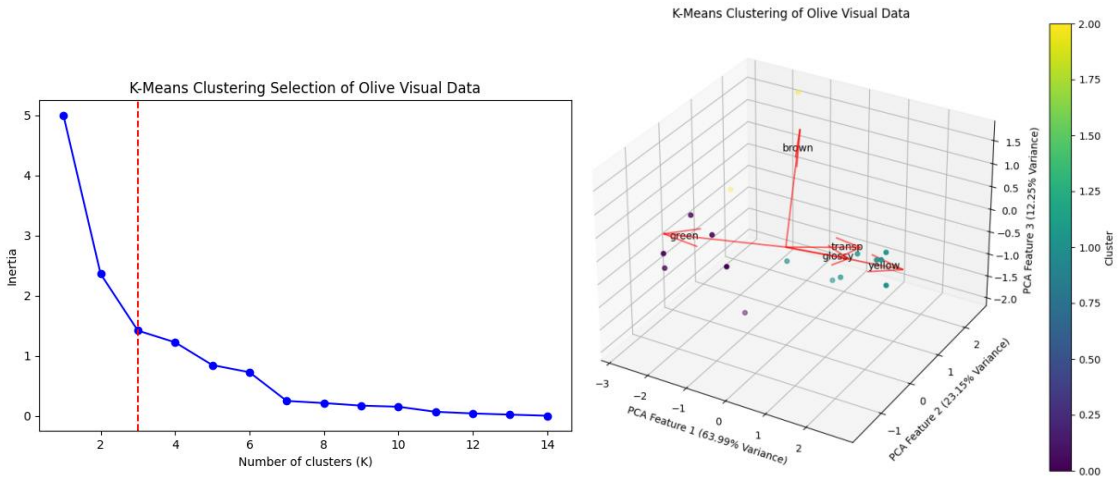

图 10/11　橄榄油按视觉特征指标聚类

4.2.3 典型相关分析

利用提取到的衡量氧化程度的主成分，与衡量酸度的指标进行典型相关分析，得到如下载荷与典型相关系数：

```
Canonical component 1 X loadings: [0.88705416 -0.28058296 -0.36661712]
Canonical component 1 Y loadings: [-1.]
First canonical correlation: 0.8368318567565435
```

根据典型相关载荷可知，绿色度与粘稠度呈高度负相关，而其它因素与粘稠度存在正相关。根据典型相关系数和典型相关分析效果图（图 12），两个典型相关变量之间存在很好的线性关系。据此，可以根据得到的典型相关关系可以建立线性回归模型，进而可以直接通过视觉特征对粘稠度进行预测。

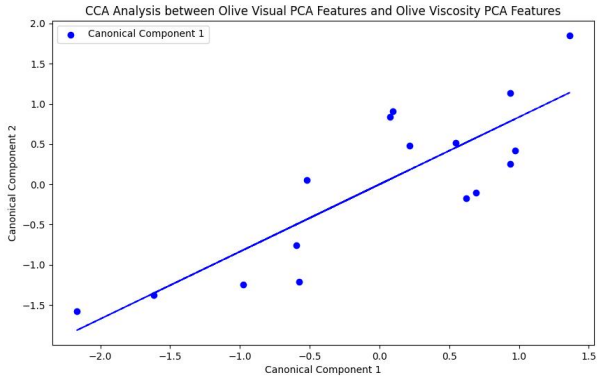

图 12　橄榄油视觉特征-浓稠度典型相关分析

4.3 物理性质 vs 化学性质

4.3.1 主成分分析

橄榄油的物理性质包括黄色度、绿色度、棕色度、光泽度、透明度和浓稠度；化学性质包括过氧化值、K232、K270、ΔK 和酸度。在 EDA 中观察到黄色度与绿色度、透明度与光泽度、过氧化值与 K232、K270 等指标间存在较强的线性相关性，故可以尝试 PCA 方法对其进行降维，提取关键成分。

4.3.1.1 主成分个数的选择

如图 13/14，前三个主成分均能够解释绝大部分的方差，故选取主成分个数均为 3。

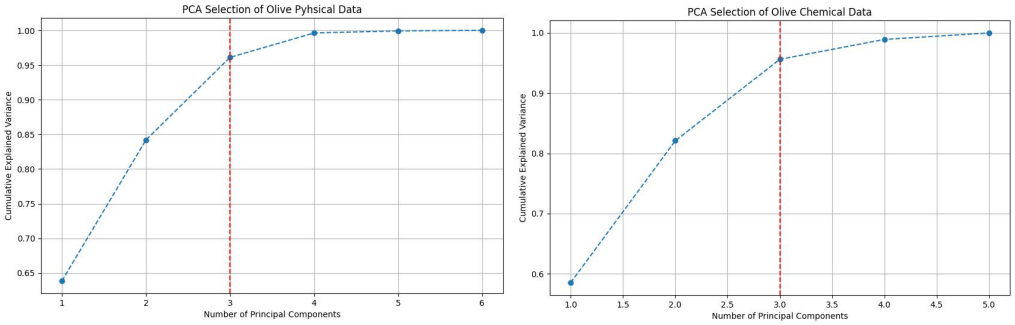

图 13/14　橄榄油物理与化学性质指标主成分个数选择

4.3.1.2 主成分分析结果

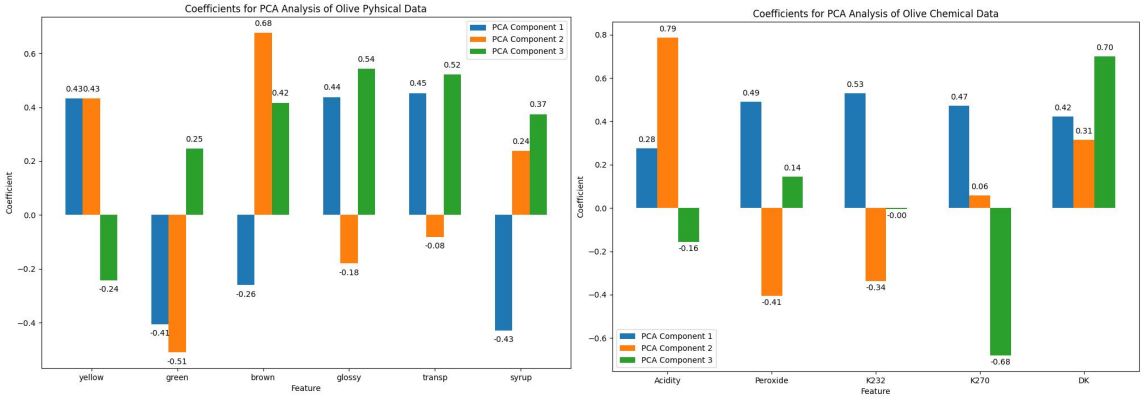

图 13/14　橄榄油物理与化学性质指标主成分载荷

对比图 9、13，对于物理性质，主成分分析结果与对去除浓稠度后的视觉特征进行主成分分析所得基本一致，浓稠度与三个主成分的关系与典型相关分析所得结果相吻合。这说明视觉特征能够保持绝大部分的物理信息，基于视觉特征对浓稠度的预测是合理且可靠的。

对比图 4、14，对于化学性质，主成分分析结果相比对去除酸度后的氧化度特征进行主成分分析所得出现了变异。具体地，第一主成分保持一致，加入酸度后第二主成分成为第三主成分，并新增由酸度和 ΔK 描述的第二主成分。该主成分与酸度和 ΔK 值呈正相关，并与初级氧化物指标呈负相关，体现了酸度与次级氧化物指标在橄榄油化学性质中起到的联合作用。比较关于氧化程度和酸度的 CCA 结果，可见该主成分与 CCA 结果相一致。基于此，氧化程度指标可能无法完整地描述全部感兴趣的化学性质，这一点与视觉效果在物理性质中起到的作用并不完全相同。

4.3.2 聚类分析

分别根据橄榄油的物理性质和化学性质将样本分成不同的组。采用 K-means 方法对其进行聚类，可以识别出橄榄油在不同物理性质和化学性质下的分布情况。

4.3.2.1 聚类个数的选择

采用肘部法则选择合适的 K 值。如图 15/16，可见肘部点分别位于 K=4 与 K=3。

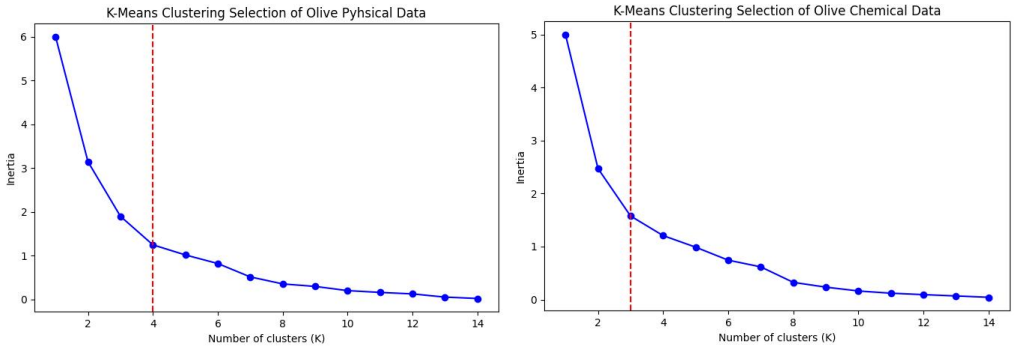

图 15/16　橄榄油按物理/化学指标聚类 K 值选择

4.3.2.2 K-means 聚类分析结果

利用 PCA 指标与 K-means 聚类结果对样本按视觉特征进行分类，并可视化，如图 17/18。

相比仅依赖视觉特征的聚类结果，综合考虑了全部物理特征的聚类倾向于将全部橄榄油分为 4 类。新增的一类产生的原因在于将具有较好视觉品质的一类橄榄球进一步细分为高透明度（或高光泽度）与低透明度（或低光泽度）两类。

对于化学特征，由于采用的 PCA 指标与之前不一致，因此对于聚类结果可以存在新的解读。

根据聚类结果，可以将样本按化学性质分为 3 类：

1. 紫色，低 PCA1，代表总体氧化程度较低的一类橄榄油。
2. 绿色，高 PCA1，低 PCA2，可能代表总体氧化程度较高，且以初级氧化物为主、酸度较低的一类橄榄油。
3. 黄色，高 PCA1，高 PCA2，高 PCA3，可能代表总体氧化程度较高，且以次级氧化物为主、酸度较高的一类橄榄油。

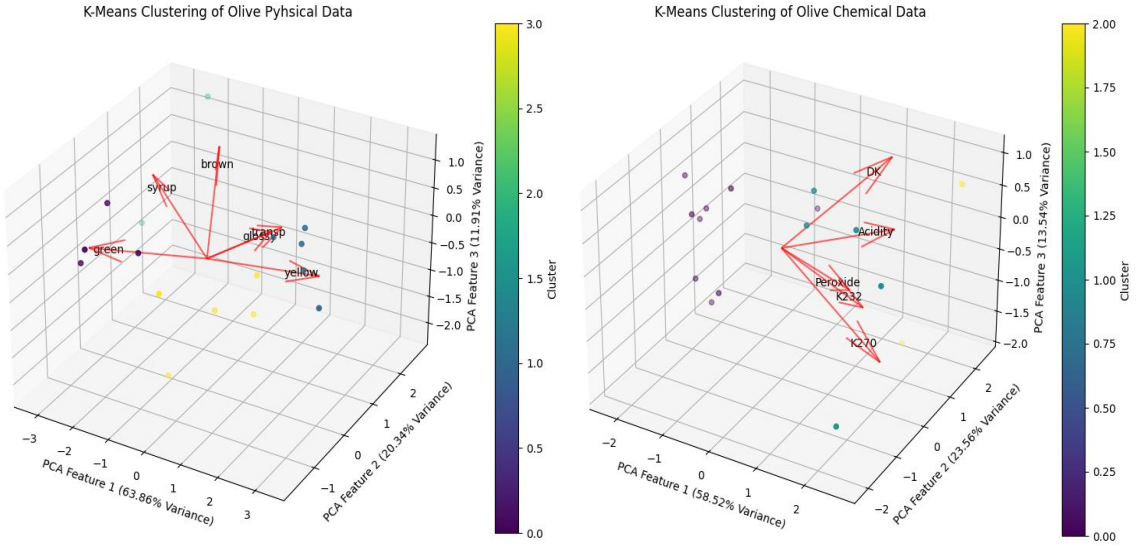

图 17/18　橄榄油按物理/化学指标聚类结果可视化

4.3.3 典型相关分析

利用提取到的衡量物理性质的主成分，与衡量化学性质的主成分进行典型相关分析，得到如下载荷与典型相关系数（典型相关效果图如图 19）：

```
Canonical component 1 X loadings: [ 0.86059859 -0.45246551 -0.23376277]
Canonical component 1 Y loadings: [-0.87260049  0.46235229  0.15747614]
First canonical correlation: 0.888898924341721
Canonical component 2 X loadings: [0.49060323 0.85971678 0.14211095]
Canonical component 2 Y loadings: [-0.42754533 -0.87892135  0.21142907]
First canonical correlation: 0.5969567316050456
Canonical component 3 X loadings: [ 0.13666947 -0.23698526  0.96185209]
Canonical component 3 Y loadings: [-0.23616386 -0.11716492 -0.96462377]
First canonical correlation: 0.31063154764042106
```

可见，第一典型相关系数主要由描述物理性质的第一主成分与描述化学性质的第一主成分决定，这两个主成分也是我们最关心的部分。前者与黄色度、光泽度与透明度呈正相关，与其它物理性质呈负相关；后者描述了总计氧化水平与酸度。据此，我们可以利用第一典型相关系数建立线性回归模型，从而根据物理性质评估橄榄油质量。

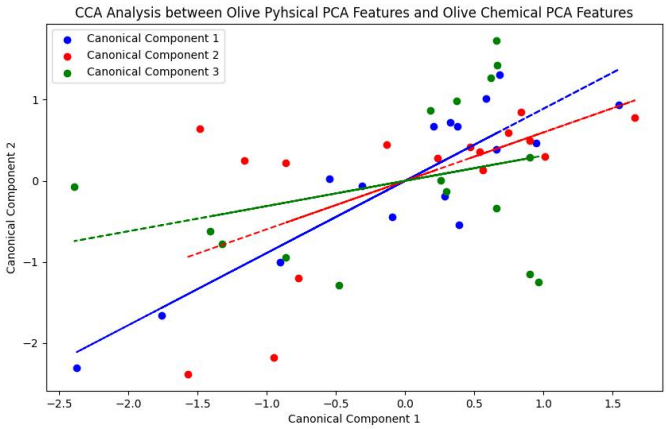

图 19　橄榄油物理-化学性质典型相关分析、

5. 总结

本研究通过对橄榄油质量特性进行多维数据分析，揭示了这些特性之间的复杂关系，提供了科学的依据来提升橄榄油的质量控制和选购决策。首先，研究发现橄榄油的视觉特征（如黄色度、绿色度、棕色度、光泽度和透明度）与其物理特性（如浓稠度）之间存在显著关联。通过主成分分析，能够提取出关键成分，对这些特性进行有效分类。这为消费者在选购橄榄油时提供了更加直观的参考依据。

同时，对于橄榄油的氧化程度和酸度的研究表明，氧化程度的多个衡量指标（过氧化值、K232、K270 和ΔK）之间存在较强的线性相关性，可以通过主成分分析提取出前三个主成分来解释大部分的方差，从而有效地对橄榄油的氧化状态进行分类。聚类分析进一步将橄榄油样本按氧化程度和酸度分为不同类别，揭示了橄榄油在不同氧化程度下的分布情况。

研究还通过对橄榄油的物理和化学性质进行典型相关分析，验证了它们之间存在较好的线性关系。物理特性的第一主成分（主要与黄色度、光泽度和透明度相关）与化学特性的第一主成分（主要描述总氧化水平和酸度）之间的关联，为通过物理特性预测橄榄油质量提供了可能性。这一结果不仅为生产者提供了全面的质量控制策略，还为消费者在选购橄榄油时提升了决策能力。

综上所述，本研究通过多维数据分析，系统探讨了橄榄油质量特性之间的关系，为橄榄油的质量控制和选购提供了科学依据。这不仅有助于提高消费者的选购体验，也为生产者提供了重要的质量管理工具。

6. 讨论

本研究在探索橄榄油质量特性之间的关系时存在一些限制。首先，数据集的样本量较小，仅包含 16 个样本，可能无法全面代表不同产地和加工方式的橄榄油特性，限制了结论的普适性。其次，本研究主要依赖于现有的物理和化学特性指标，而未考虑可能影响橄榄油质量的其他因素，如生产环境、储存条件和包装方式。此外，数据分析方法如主成分分析和聚类分析虽然能够揭示特性之间的关系，但对非线性关系和复杂交互作用的处理能力有限。最后，消费者感官评价在橄榄油选购中的作用未能充分纳入本研究的分析范围，这可能影响到实际应用的效果。因此，未来研究应扩展样本量、引入更多影响因素，并结合消费者感官评价进行更加全面的分析。

参考文献

1. 橄榄油的质量分级检测技术研究进展，张英姿等，食品安全质量检测学报，第 8 卷，第 11 期。
2. http://www.good-import.com/industry/170.html
3. https://www.xbyolive.com/mei-ri-gan-lan-you/suan-du/

附录

A. 文件清单

```
.
|-- dataset
|-- description
|-- example
|-- report
|-- result
|   |-- analysis
|   |-- eda
|   `-- selection
`-- src
    |-- analysis.py
    |-- datatype.py
    |-- eda.py
    |-- main.py
    |-- preprocess.py
    `-- selection.py
```

B. Python 代码

B1 datatype.py

```python
physics_columns = [
    'yellow',
    'green',
    'brown',
    'glossy',
    'transp',
    'syrup',
]

chemistry_columns = [
    'Acidity',
    'Peroxide',
    'K232',
    'K270',
    'DK'
]

visual_columns = [
    'yellow',
    'green',
    'brown',
    'glossy',
    'transp',
]

color_columns = [
    'yellow',
    'green',
    'brown',
]

light_columns = [
    'glossy',
    'transp',
]

viscosity_columns = [
    'syrup',
]

acid_columns = [
    'Acidity',
]

oxidation_columns = [
    'Peroxide',
    'K232',
    'K270',
    'DK'
]

```

B2 eda.py

```python
import seaborn as sns
import matplotlib.pyplot as plt


def eda(data, show=True):

    # 数据信息
    print("Data Info:")
    print(data.info())

    print("\nData Description:")
    print(data.describe())

    print("\nMissing Values:")
    print(data.isnull().sum())

    with open('result/eda/data_info.txt', 'w') as f:
        f.write("Data Info:\n")
        data.info(buf=f)

        f.write("\nData Description:\n")
        f.write(data.describe().to_string())

        f.write("\n\nMissing Values:\n")
        f.write(data.isnull().sum().to_string())

    # 分布
    plt.figure(figsize=(12, 8))
    for i, column in enumerate(data.columns[1:], 1):
        plt.subplot(3, 4, i)
        sns.histplot(data[column], kde=True)
        plt.title(f'Distribution of {column}')
    plt.tight_layout()
    if show:
        plt.show()
    else:
        plt.savefig('result/eda/distribution.png')

    # pairplot
    plt.figure(figsize=(24, 16))
    sns.pairplot(data.drop(columns=['ID']))
    if show:
        plt.show()
    else:
        plt.savefig('result/eda/pairplot.png')

    # 相关矩阵
    corr_matrix = data.drop(columns=['ID']).corr()

    plt.figure(figsize=(12, 8))
    sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
    plt.title('Correlation Matrix')
    if show:
        plt.show()
    else:
        plt.savefig('result/eda/corr_matrix.png')
```

B3 preprocess.py

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler


def preprocess_data(data):
    # 标准化
    data = data.drop(columns=['ID'])
    scaler = StandardScaler()
    data_scaled = scaler.fit_transform(data)
    data_scaled = pd.DataFrame(data_scaled, columns=data.columns)

    return data_scaled

```

B4 selection.py

```python
import numpy as np
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt
from kneed import KneeLocator
import logging

logger = logging.getLogger(__name__)
logger.setLevel(logging.INFO)

def Select_PCA(features, critical_value=0.9, plot=True, title=None, show=True):
    if plot and title is None:
        logger.error("Missing parameters for plotting PCA analysis, skipping...")
        plot = False

    pca = PCA()
    pca.fit(features)

    # 计算解释方差比率
    explained_variance_ratio = pca.explained_variance_ratio_
    cumulative_explained_variance = np.cumsum(explained_variance_ratio)
    n_components = np.argmax(cumulative_explained_variance >= critical_value) + 1

    if plot:
        # 可视化累积方差解释比例
        plt.figure(figsize=(10, 6))
        plt.plot(range(1, len(cumulative_explained_variance) + 1),
                 cumulative_explained_variance, marker='o', linestyle='--')
        plt.axvline(x=n_components, color='r', linestyle='--')
        plt.xlabel('Number of Principal Components')
        plt.ylabel('Cumulative Explained Variance')
        plt.title(title)
        plt.grid()
        if show:
            plt.show()
        else:
            plt.savefig(f'result/selection/{title.replace(" ", "_")}.png')

    # 打印解释方差比率和累积解释方差比率
    print("Explained variance ratio per principal component:")
    print(explained_variance_ratio)
    print("Cumulative explained variance ratio:")
    print(cumulative_explained_variance)

    return n_components


```

```python
def Select_CA(features, method='linkage', plot=True, title=None, show=True):
    if plot and title is None:
        logger.error("Missing parameters for plotting CA analysis, skipping...")
        plot = False

    if method == 'linkage':
        # 使用层次聚类确定最佳簇数量
        Z = linkage(features, method='ward')
        plt.figure(figsize=(12, 8))
        dendrogram(Z)
        plt.title(title)
        plt.xlabel('Sample ID')
        plt.ylabel('Distance')
        plt.show()


    elif method == 'kmeans':
        # 使用肘部法则确定最佳K值
        inertia = []
        K_range = range(1, 15)
        for k in K_range:
            kmeans = KMeans(n_clusters=k, random_state=72)
            kmeans.fit(features)
            inertia.append(kmeans.inertia_)

        normalized_inertia = [i / len(features) for i in inertia]

        kneedle = KneeLocator(K_range, normalized_inertia, curve='convex', direction='decreasing')
        optimal_k = kneedle.elbow

        if plot:
            plt.figure(figsize=(8, 5))
            plt.plot(K_range, normalized_inertia, 'bo-')
            plt.xlabel('Number of clusters (K)')
            plt.ylabel('Inertia')
            plt.title(title)
            plt.axvline(x=optimal_k, color='r', linestyle='--')
            if show:
                plt.show()
            else:
                plt.savefig(f'result/selection/{title.replace(" ", "_")}.png')

        print(f'The optimal number of clusters is {optimal_k}')

        return optimal_k

```

B5 analysis.py

```python
import numpy as np
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.cross_decomposition import CCA
from scipy.cluster.hierarchy import linkage, fcluster
import seaborn as sns
import matplotlib.pyplot as plt
import logging

logger = logging.getLogger(__name__)
logger.setLevel(logging.INFO)
```

```python
def Analysis_Corr(X, Y, plot=False, x_columns=None, y_columns=None, title=None, xlabel=None, ylabel=None, show=True)
    if plot and (x_columns is None or y_columns is None or title is None or xlabel is None or ylabel is None):
        logger.error("Missing parameters for plotting correlation analysis, skipping...")
        plot = False

    correlation_matrix = np.corrcoef(X.T, Y.T)
    correlation_matrix = correlation_matrix[X.shape[1]:, :X.shape[1]]

    if plot:
        # 绘制热力图
        plt.figure(figsize=(10, 6))
        sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', xticklabels=x_columns, yticklabels=y_columns)
        plt.title(title)
        plt.xlabel(xlabel)
        plt.ylabel(ylabel)
        if show:
            plt.show()
        else:
            plt.savefig(f'result/analysis/{title.replace(" ", "_")}.png')

    return correlation_matrix


def Analysis_PCA(features, n_components, plot=False, feature_columns=None, title=None, show=True):
    if plot and (feature_columns is None or title is None):
        logger.error("Missing parameters for plotting PCA analysis, skipping...")
        plot = False

    pca = PCA(n_components=n_components)
    pca_features = pca.fit_transform(features)

    if plot:
        if n_components == 2:
            # 可视化前两个主成分
            plt.figure(figsize=(12, 8))
            plt.scatter(pca_features[:, 0], pca_features[:, 1], marker='o')
            plt.xlabel('PCA Feature 1 (%.2f%% Variance)' % (pca.explained_variance_ratio_[0] * 100))
            plt.ylabel('PCA Feature 2 (%.2f%% Variance)' % (pca.explained_variance_ratio_[1] * 100))
            plt.title(title)

            # 可视化每个特征在 PCA 空间中的权重
            loading_vectors = pca.components_.T * np.sqrt(pca.explained_variance_)
            for i, feature in enumerate(feature_columns):
                plt.arrow(0, 0, loading_vectors[i, 0] * 8, loading_vectors[i, 1] * 8,
                          color='red', alpha=0.5, head_width=0.05, head_length=0.1)
                plt.text(loading_vectors[i, 0] * 8.5 + 3, loading_vectors[i, 1] * 8.5,
                         feature, color='black', ha='center', va='center', fontsize=12)
            if show:
                plt.show()
            else:
                plt.savefig(f'result/analysis/{title.replace(" ", "_")}.png')

        elif n_components > 2:
            fig = plt.figure(figsize=(12, 8))
            ax = fig.add_subplot(111, projection='3d')

            ax.scatter(pca_features[:, 0], pca_features[:, 1], pca_features[:, 2], c='b', marker='o')
            ax.set_xlabel('PCA Feature 1 (%.2f%% Variance)' % (pca.explained_variance_ratio_[0] * 100))
            ax.set_ylabel('PCA Feature 2 (%.2f%% Variance)' % (pca.explained_variance_ratio_[1] * 100))
            ax.set_zlabel('PCA Feature 3 (%.2f%% Variance)' % (pca.explained_variance_ratio_[2] * 100))
            ax.set_title(title)

            k_arrow = 0.8
            k_text = 2.5
            loading_vectors = pca.components_.T * np.sqrt(pca.explained_variance_)
            for i, feature in enumerate(feature_columns):
                ax.quiver(0, 0, 0, loading_vectors[i, 0] * k_arrow, loading_vectors[i, 1] * k_arrow, loading_vectors[i, 2] * k_arrow,
                          color='r', alpha=0.6, length=3)
                ax.text(loading_vectors[i, 0] * k_text, loading_vectors[i, 1] * k_text, loading_vectors[i, 2] * k_text,
                         feature, color='black', ha='center', va='center', fontsize=10)
            if show:
                plt.show()
            else:
                plt.savefig(f'result/analysis/{title.replace(" ", "_")}.png')
```

```python

        # 可视化所有主成分的系数
        plt.figure(figsize=(12, 8))
        bar_width = 0.2
        x = np.arange(len(feature_columns))

        for i in range(n_components):
            plt.bar(x + i * bar_width, pca.components_[i], bar_width, label=f'PCA Component {i + 1}')

        plt.xlabel('Feature')
        plt.ylabel('Coefficient')
        plt.title(f'Coefficients for {title}')
        plt.xticks(x + bar_width * (n_components - 1) / 2, feature_columns)

        for i in range(n_components):
            for j in range(len(feature_columns)):
                plt.text(x[j] + i * bar_width, pca.components_[i, j] + 0.02 if pca.components_[i, j] > 0 else pca.components_[i, j] - 0.05,
                         f'{pca.components_[i, j]:.2f}', ha='center', va='bottom')
        plt.legend()
        if show:
            plt.show()
        else:
            plt.savefig(f'result/analysis/{title.replace(" ", "_")}_coefficients.png')

    return pca, pca_features


def Analysis_CA(features, optimal_k, method='linkage', plot=False, pca=None, feature_columns=None, title=None, show=True):
    logger = logging.getLogger(__name__)
    if plot and (pca is None or feature_columns is None or title is None):
        logger.error("Missing parameters for plotting CA analysis, skipping...")
        plot = False

    if method == 'linkage':
        Z = linkage(features, method='ward')
        clusters = fcluster(Z, optimal_k, criterion='maxclust')
        labels = clusters - 1
    elif method == 'kmeans':
        kmeans = KMeans(n_clusters=optimal_k, random_state=72)
        kmeans.fit(features)
        labels = kmeans.labels_

    if plot:
        pca_features = pca.fit_transform(features)
        if pca_features.shape[1] == 2:
            # 可视化聚类结果
            plt.figure(figsize=(10, 6))
            scatter = plt.scatter(pca_features[:, 0], pca_features[:, 1], c=labels, cmap='viridis', marker='o')
            plt.xlabel('PCA Feature 1 (%.2f%% Variance)' % (pca.explained_variance_ratio_[0] * 100))
            plt.ylabel('PCA Feature 2 (%.2f%% Variance)' % (pca.explained_variance_ratio_[1] * 100))
            plt.title(title)
            plt.colorbar(scatter, label='Cluster')

            # 可视化每个特征在 PCA 空间中的权重
            loading_vectors = pca.components_.T * np.sqrt(pca.explained_variance_)
            for i, feature in enumerate(feature_columns):
                plt.arrow(0, 0, loading_vectors[i, 0] * 12, loading_vectors[i, 1] * 12,
                          color='red', alpha=0.5, head_width=0.05, head_length=0.1)
                plt.text(loading_vectors[i, 0] * 12.5 + 4, loading_vectors[i, 1] * 12.5,
                         feature, color='black', ha='center', va='center', fontsize=12)
            if show:
                plt.show()
            else:
                plt.savefig(f'result/analysis/{title.replace(" ", "_")}.png')

        elif pca_features.shape[1] > 2:
            fig = plt.figure(figsize=(12, 8))
            ax = fig.add_subplot(111, projection='3d')

            scatter = ax.scatter(pca_features[:, 0], pca_features[:, 1], pca_features[:, 2], c=labels, cmap='viridis', marker='o')
            ax.set_xlabel('PCA Feature 1 (%.2f%% Variance)' % (pca.explained_variance_ratio_[0] * 100))
            ax.set_ylabel('PCA Feature 2 (%.2f%% Variance)' % (pca.explained_variance_ratio_[1] * 100))
            ax.set_zlabel('PCA Feature 3 (%.2f%% Variance)' % (pca.explained_variance_ratio_[2] * 100))
            ax.set_title(title)
            fig.colorbar(scatter, label='Cluster')

            k_arrow = 0.8
            k_text = 2
            loading_vectors = pca.components_.T * np.sqrt(pca.explained_variance_)
            for i, feature in enumerate(feature_columns):
                ax.quiver(0, 0, 0, loading_vectors[i, 0] * k_arrow, loading_vectors[i, 1] * k_arrow, loading_vectors[i, 2] * k_arrow,
                          color='r', alpha=0.6, length=3)
                ax.text(loading_vectors[i, 0] * k_text, loading_vectors[i, 1] * k_text, loading_vectors[i, 2] * k_text,
                        feature, color='black', ha='center', va='center', fontsize=10)

            if show:
                plt.show()
            else:
                plt.savefig(f'result/analysis/{title.replace(" ", "_")}.png')

    return kmeans


```

```python
def Analysis_CCA(X, Y, n_components, plot=False, x_columns=None, y_columns=None, title=None, show=True):
    if plot and (x_columns is None or y_columns is None or title is None):
        logger.error("Missing parameters for plotting CCA analysis, skipping...")
        plot = False

    cca = CCA(n_components=n_components)
    cca.fit(X, Y)
    X_c, Y_c = cca.transform(X, Y)

    # 打印所有典型载荷
    for i in range(n_components):
        print(f"Canonical component {i + 1} X loadings: {cca.x_loadings_[:, i]}")
        print(f"Canonical component {i + 1} Y loadings: {cca.y_loadings_[:, i]}")
        print("First canonical correlation:", np.corrcoef(X_c[:, i], Y_c[:, i])[0, 1])

    if plot:
        # # 提取典型载荷
        # U_coefficients = cca.x_loadings_[:, 0]
        # V_coefficients = cca.y_loadings_[:, 0]

        # # 可视化典型载荷
        # plt.figure(figsize=(12, 6))

        # plt.subplot(1, 2, 1)
        # plt.barh(x_columns, U_coefficients, color='b')
        # plt.xlabel('Coefficient Value')
        # plt.title(f'{title} loadings for X')

        # plt.subplot(1, 2, 2)
        # plt.barh(y_columns, V_coefficients, color='r')
        # plt.xlabel('Coefficient Value')
        # plt.title(f'{title} loadings for Y')

        # plt.tight_layout()
        # if show:
        #     plt.show()
        # else:
        #     plt.savefig(f'result/analysis/{title.replace(" ", "_")}_loadings.png')

        # 可视化典型相关变量关系
        plt.figure(figsize=(10, 6))
        color = ['b', 'r', 'g', 'y', 'c', 'm', 'k', 'w']
        for i in range(n_components):
            plt.scatter(X_c[:, i], Y_c[:, i], color=color[i], label=f'Canonical Component {i + 1}')
        plt.xlabel('Canonical Component 1')
        plt.ylabel('Canonical Component 2')
        plt.title(title)
        plt.legend()

        # 过原点线性拟合
        for i in range(n_components):
            a, b = np.polyfit(X_c[:, i], Y_c[:, i], 1)
            plt.plot(X_c[:, i], a * X_c[:, i] + b, color=color[i], linestyle='--', label=f'Linear Fit {i + 1}')

        if show:
            plt.show()
        else:
            plt.savefig(f'result/analysis/{title.replace(" ", "_")}.png')


    return cca, X_c, Y_c
```

B5 main.py

```python
import pandas as pd
from selection import Select_PCA, Select_CA
from analysis import Analysis_PCA, Analysis_CA, Analysis_CCA, Analysis_Corr
from preprocess import preprocess_data
from eda import eda
import datatype as dt
import os


```

```python
data = pd.read_csv('dataset/oliveoil.csv')

os.makedirs('result/eda',exist_ok=True)
os.makedirs('result/selection',exist_ok=True)
os.makedirs('result/analysis',exist_ok=True)


def main(x_columns, y_columns, x_name, y_name, critical_value=0.9, plot=True, show=True):
    global data
    x_features = data[x_columns]
    y_features = data[y_columns]
    xy_features = data[x_columns + y_columns]


    correlation_matrix = Analysis_Corr(
        x_features,
        y_features,
        plot=plot,
        x_columns=x_columns,
        y_columns=y_columns,
        title=f'Correlation Matrix between X ({x_name} Features) and Y ({y_name} Features)',
        xlabel=f'{x_name} Features',
        ylabel=f'{y_name} Features',
        show=show
    )


    x_n_components = Select_PCA(
        x_features,
        critical_value=critical_value,
        plot=plot,
        title=f'PCA Selection of Olive {x_name} Data',
        show=show
    )

    x_pca, x_pca_features = Analysis_PCA(
        x_features,
        x_n_components,
        plot=plot,
        feature_columns=x_columns,
        title=f'PCA Analysis of Olive {x_name} Data',
        show=show
    )

    # 将 PCA 结果添加回原数据集
    for i in range(x_n_components):
        data[f'{x_name}_PCA_{i + 1}'] = x_pca_features[:, i]
    x_pca_columns = [f'{x_name}_PCA_{i}' for i in range(1, x_n_components + 1)]
    x_pca_features = data[x_pca_columns]


    y_n_components = Select_PCA(
        y_features,
        critical_value=critical_value,
        plot=plot,
        title=f'PCA Selection of Olive {y_name} Data',
        show=show
    )

    y_pca, y_pca_features = Analysis_PCA(
        y_features,
        y_n_components,
        plot=plot,
        feature_columns=y_columns,
        title=f'PCA Analysis of Olive {y_name} Data',
        show=show
    )


```

```python
# 将 PCA 结果添加回原数据集
for i in range(y_n_components):
    data[f'{y_name}_PCA_{i + 1}'] = y_pca_features[:, i]
y_pca_columns = [f'{y_name}_PCA_{i}' for i in range(1, y_n_components + 1)]
y_pca_features = data[y_pca_columns]


x_optimal_k = Select_CA(
    x_features,
    method="kmeans",
    plot=plot,
    title=f'K-Means Clustering Selection of Olive {x_name} Data',
    show=show
)

x_kmeans = Analysis_CA(
    x_features,
    x_optimal_k,
    method="kmeans",
    plot=plot,
    pca=x_pca,
    feature_columns=x_columns,
    title=f'K-Means Clustering of Olive {x_name} Data',
    show=show
)


y_optimal_k = Select_CA(
    y_features,
    method="kmeans",
    plot=plot,
    title=f'K-Means Clustering Selection of Olive {y_name} Data',
    show=show
)

y_kmeans = Analysis_CA(
    y_features,
    y_optimal_k,
    method="kmeans",
    plot=plot,
    pca=y_pca,
    feature_columns=y_columns,
    title=f'K-Means Clustering of Olive {y_name} Data',
    show=show
)


correlation_matrix_pca = Analysis_Corr(
    x_pca_features,
    y_pca_features,
    plot=plot,
    x_columns=x_pca_columns,
    y_columns=y_pca_columns,
    title=f'Correlation Matrix between {x_name} PCA Features and {y_name} PCA Features',
    xlabel=f'{x_name} PCA Features',
    ylabel=f'{y_name} PCA Features',
    show=show
)


cca, X_c, Y_c = Analysis_CCA(
    x_pca_features,
    y_pca_features,
    n_components=min(x_n_components, y_n_components),
    plot=plot,
    x_columns=x_pca_columns,
    y_columns=y_pca_columns,
    title=f'CCA Analysis between Olive {x_name} PCA Features and Olive {y_name} PCA Features',
    show=show
)
```

```
eda(data, show=True)
data = preprocess_data(data)
# 氧化程度与酸度的关系
main(dt.oxidation_columns, dt.acid_columns, 'Oxidation', 'Acid', critical_value=0.9, plot=True, show=False)
# 视觉与浓稠度的关系
main(dt.visual_columns, dt.viscosity_columns, 'Visual', 'Viscosity', critical_value=0.9, plot=True, show=False)
# 物理性质与化学性质的关系
main(dt.physics_columns, dt.chemistry_columns, 'Pyhsical', 'Chemical', critical_value=0.9, plot=True, show=True)
```