Data Mining on Wine Datasets: A Case Study in Clustering and Classification

Samimuddin Ahmed

24225041

Date: 04-04-2025

Executive Summary

This case study presents a data mining analysis of the Wine Quality dataset, which contains physicochemical properties of red wine samples along with their quality ratings. The objective is to explore, model, and interpret the data using key data mining techniques—classification, clustering, and association rule mining.

For the classification task, models such as K-Nearest Neighbors (KNN) and Decision Tree were implemented to predict wine quality based on measurable features like acidity, alcohol content, and pH. The performance of these models was evaluated using metrics like accuracy and F1-score.

In the clustering task, unsupervised learning algorithms including K-Means, Hierarchical Clustering, and DBSCAN were applied to uncover natural groupings within the wine samples. Dimensionality reduction through Principal Component Analysis (PCA) was used to visualize cluster separation effectively.

Additionally, association rule mining using the Apriori algorithm was attempted to identify patterns and co-occurrences among discretized features, though preprocessing challenges were encountered.

Overall, the study demonstrates how data mining techniques can be leveraged to classify wine quality, discover inherent patterns, and generate insights from a real-world dataset.

Introduction

This case study explores the application of core data mining techniques on the widely used Wine Quality dataset, which consists of physicochemical attributes of red wine samples along with their corresponding quality ratings. The primary objective is to demonstrate how methods such as Classification, Clustering, and Association Rule Mining can be effectively utilized to uncover hidden patterns, develop predictive models, and gain deeper insights into the structure of the data.

The dataset includes features such as fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, alcohol content, and others—each representing a chemical property measured from wine samples. The target variable, *quality*, is a score (typically ranging from 0 to 10) assigned by wine tasters based on sensory evaluation.

Data Description

The dataset used in this case study is the Red Wine Quality dataset, sourced from the UCI Machine Learning Repository and also available on Kaggle. It contains data on various physicochemical properties of red wine samples and their corresponding quality scores assigned by wine tasters.

Basic Structure:

- **Number of Rows:** 1,599
- **Number of Columns:** 12 (11 input features + 1 target)

Input Features (Independent Variables):

Feature Name	Description
fixed acidity	Tartaric acid (g/dm^3)
volatile acidity	Acetic acid (g/dm^3)
citric acid	Citric acid (g/dm^3)
residual sugar	Amount of remaining sugar after fermentation (g/dm^3)
chlorides	Salt concentration (g/dm^3)
free sulfur dioxide	Free SO_2 in wine (mg/dm^3)
total sulfur dioxide	Total SO_2 (mg/dm^3)
density	Density of wine (g/cm^3)
pH	Acidity/basicity
sulphates	Potassium sulphate (g/dm^3)

alcohol	Alcohol content (%)

Target Variable:

quality: Integer score (typically between **0 and 10**) representing the wine quality, rated by wine tasters. Most scores range from **3 to 8** in this dataset.

Basic Statistics:

- The dataset is **clean** with **no missing values**.
- Most wines have a quality score between **5 and 7**, making it a slightly **imbalanced classification problem**.
- Alcohol content and volatile acidity show the strongest correlation with wine quality.
- Features vary in scale and may require **normalization or standardization** for modeling.

Exploratory Data Analysis (EDA)

Data Overview

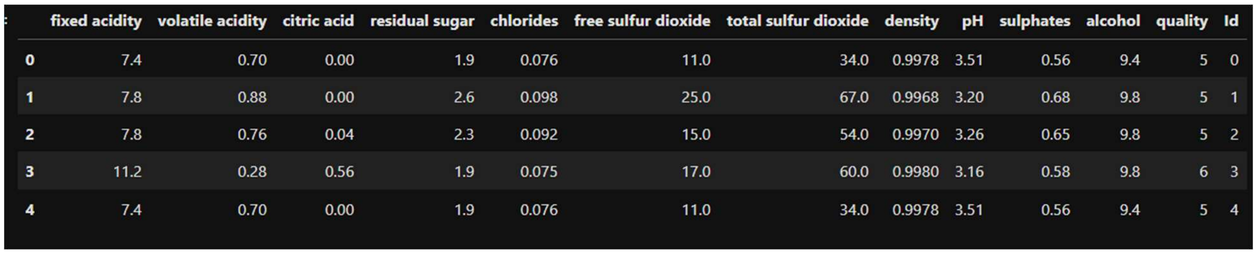
Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, f1_score, classification_report,
confusion_matrix
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN

import warnings
```

```
warnings.filterwarnings('ignore')
```

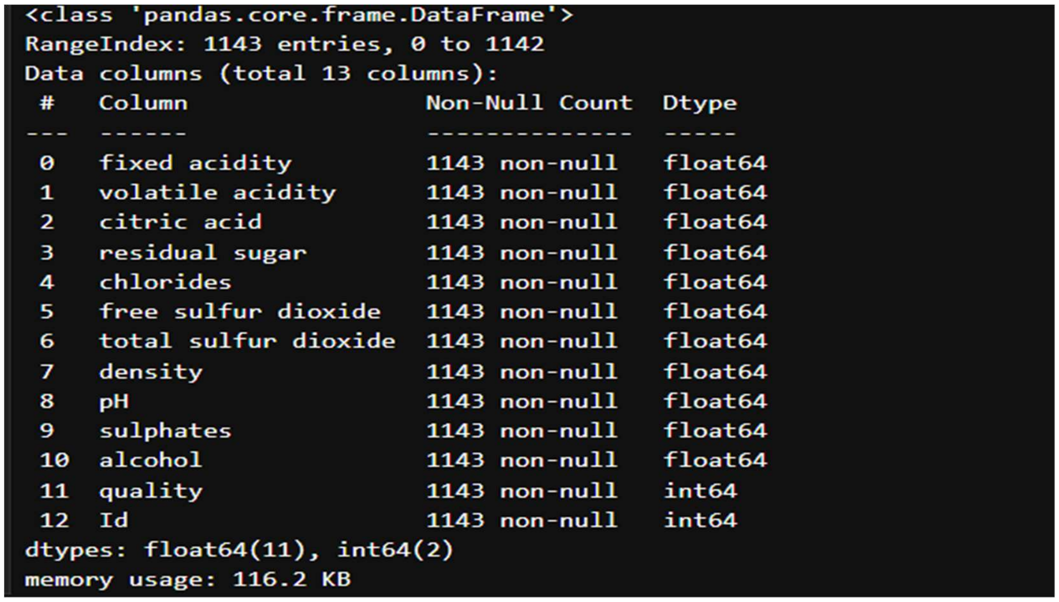
The dataset was loaded and inspected using `.head()` and `.info()`.

```
df =pd.read_csv(r"C:\Users\samdc\OneDrive\Desktop\pdf\winedataset\WineQT.csv")
df.head()
```



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	0
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	1
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	2
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	3
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	4

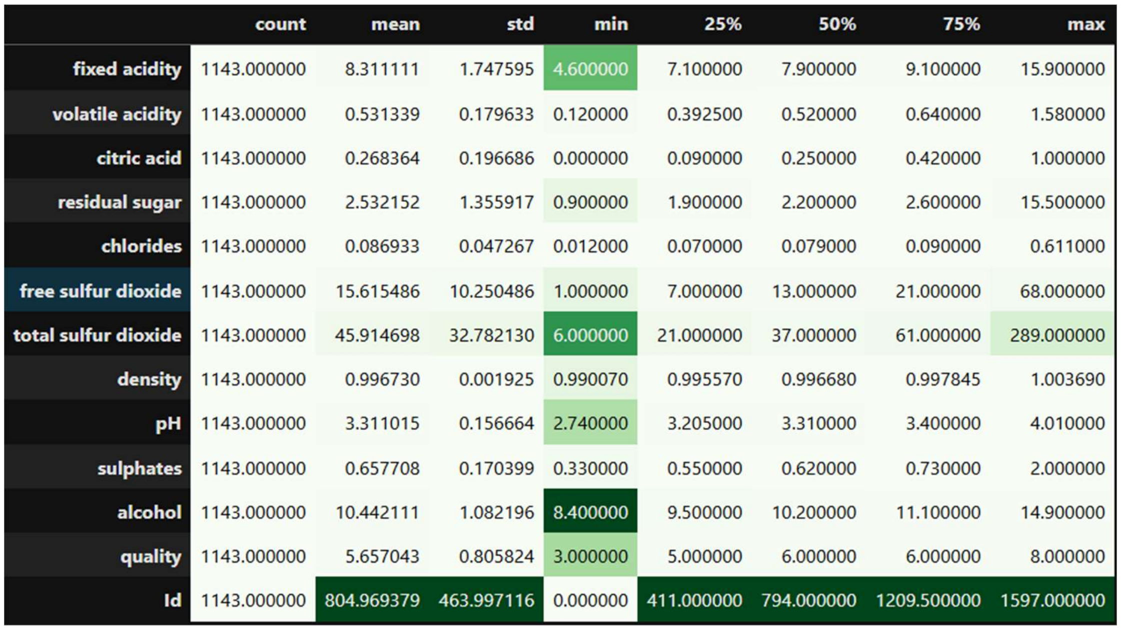
```
df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64
 12  Id                    1143 non-null   int64
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

Descriptive statistics such as mean, standard deviation, min, max, and quartiles were obtained using ‘.describe()’.

```
df.describe().T.style.background_gradient(cmap='Greens')
```

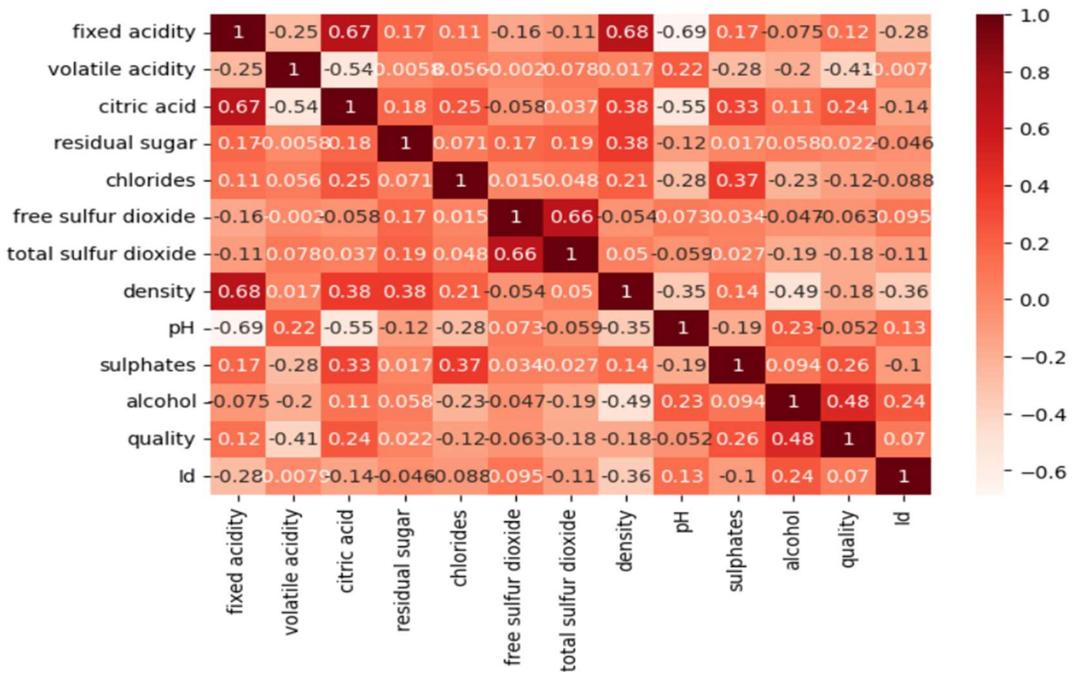


	count	mean	std	min	25%	50%	75%	max
fixed acidity	1143.000000	8.311111	1.747595	4.600000	7.100000	7.900000	9.100000	15.900000
volatile acidity	1143.000000	0.531339	0.179633	0.120000	0.392500	0.520000	0.640000	1.580000
citric acid	1143.000000	0.268364	0.196686	0.000000	0.090000	0.250000	0.420000	1.000000
residual sugar	1143.000000	2.532152	1.355917	0.900000	1.900000	2.200000	2.600000	15.500000
chlorides	1143.000000	0.086933	0.047267	0.012000	0.070000	0.079000	0.090000	0.611000
free sulfur dioxide	1143.000000	15.615486	10.250486	1.000000	7.000000	13.000000	21.000000	68.000000
total sulfur dioxide	1143.000000	45.914698	32.782130	6.000000	21.000000	37.000000	61.000000	289.000000
density	1143.000000	0.996730	0.001925	0.990070	0.995570	0.996680	0.997845	1.003690
pH	1143.000000	3.311015	0.156664	2.740000	3.205000	3.310000	3.400000	4.010000
sulphates	1143.000000	0.657708	0.170399	0.330000	0.550000	0.620000	0.730000	2.000000
alcohol	1143.000000	10.442111	1.082196	8.400000	9.500000	10.200000	11.100000	14.900000
quality	1143.000000	5.657043	0.805824	3.000000	5.000000	6.000000	6.000000	8.000000
Id	1143.000000	804.969379	463.997116	0.000000	411.000000	794.000000	1209.500000	1597.000000

Data Visualization

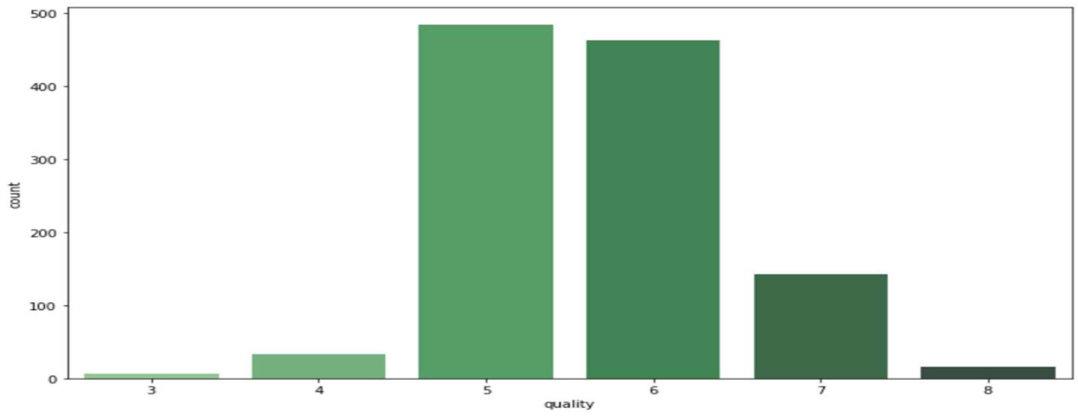
A heatmap of the correlation matrix was plotted using `seaborn.heatmap()`.

```
plt.figure(figsize=(12,7))
sns.heatmap(df.corr(),annot=True,cmap='Reds')
plt.show()
```



The **distribution of wine quality scores** in your dataset

```
plt.figure(figsize=(12,7))
sns.countplot(x='quality',data=df,palette='Greens_d')
plt.show()
```



Data Preprocessing

The dataset contains 12 columns, including 11 physicochemical attributes (like fixed acidity, volatile acidity, citric acid, etc.) and 1 target variable (quality). For clustering, only the feature columns were selected, excluding the target quality column. For classification, the quality column was used as the target variable, and the remaining attributes were used as input features.

For supervised learning models (KNN, Decision Tree, etc.), the dataset was split into training and testing sets using train_test_split()

```
feature=np.array(df[['fixed acidity','volatile acidity','citric acid','residual sugar','chlorides','free sulfur dioxide','total sulfur dioxide','density','pH','sulphates','alcohol']])
label=np.array(df['quality'])

xtrain,xtest,ytrain,ytest=train_test_split(feature,label,test_size=0.2,random_state=0)
```

Classification Models

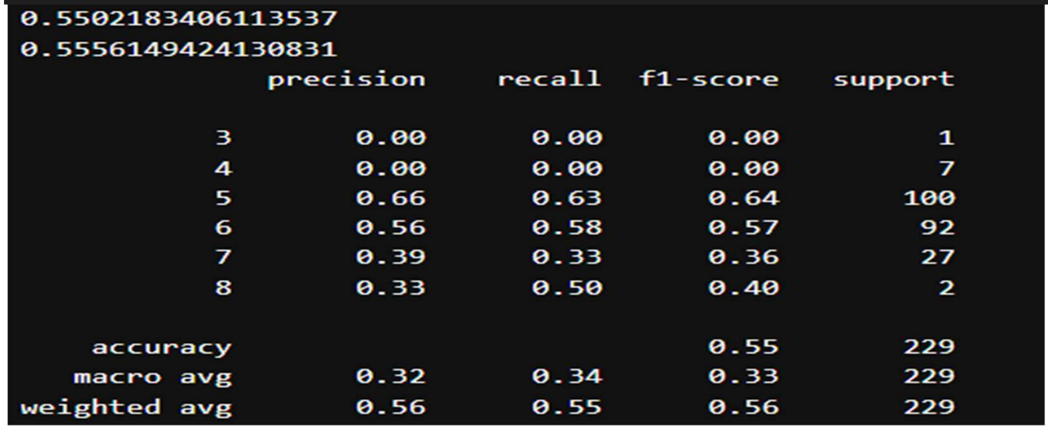
This study applies several supervised learning algorithms to predict wine quality based on its physicochemical features. The goal is to classify wine samples into their respective quality scores using various classification techniques.

```
model_comp={}
```

Decision Tree Classifier

A tree-based model that splits data into branches based on feature thresholds.

```
dt=DecisionTreeClassifier()
dt.fit(xtrain,ytrain)
y2=dt.predict(xtest)

print(accuracy_score(ytest,y2))
print(f1_score(ytest,y2,average='weighted'))
print(classification_report(ytest,y2))

model_comp['decision tree']=[accuracy_score(y2,ytest),f1_score(ytest,y2,average='weighted')]
```

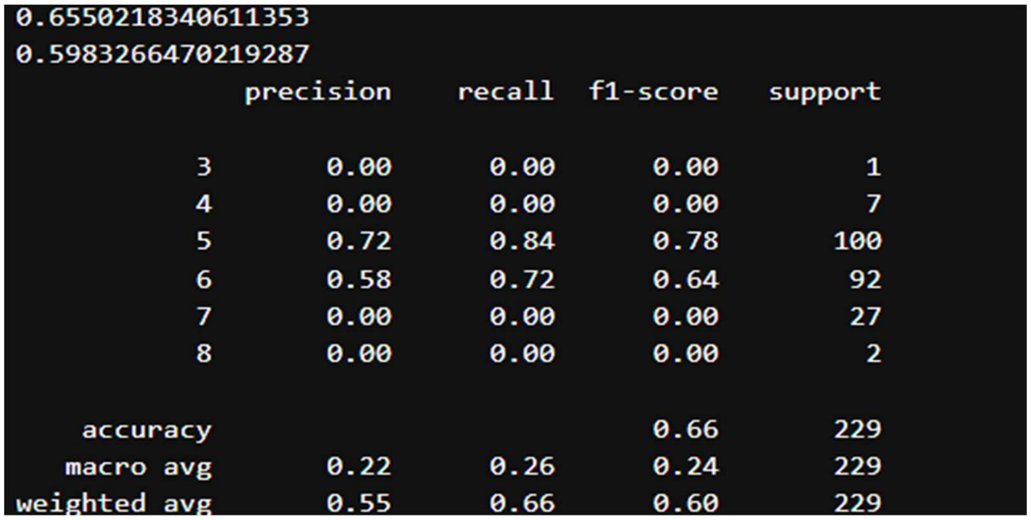


```
0.5502183406113537
0.5556149424130831
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00         7
           5       0.66      0.63      0.64       100
           6       0.56      0.58      0.57        92
           7       0.39      0.33      0.36        27
           8       0.33      0.50      0.40         2

    accuracy                           0.55       229
   macro avg       0.32      0.34      0.33       229
weighted avg       0.56      0.55      0.56       229
```

Logistic Regression

A linear model used for classification problems. Chosen for its simplicity and effectiveness on linearly separable datasets.

```
lgr=LogisticRegression()
lgr.fit(xtrain,ytrain)
y1=lgr.predict(xtest)
print(accuracy_score(ytest,y1))
print(f1_score(ytest,y1,average='weighted'))
print(classification_report(ytest,y1))

model_comp['logistic
regression']=[accuracy_score(y1,ytest),f1_score(ytest,y1,average='weighted')]
```

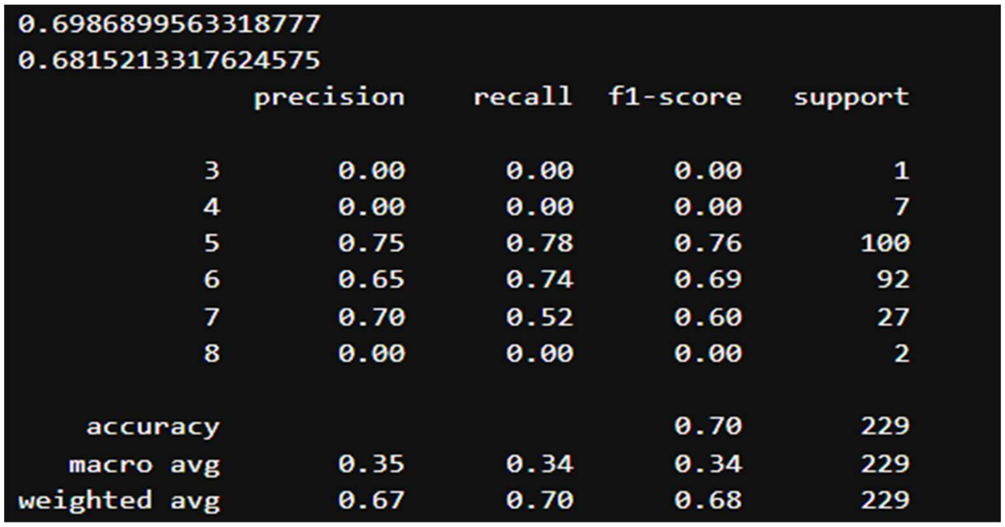


```
0.6550218340611353
0.5983266470219287
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00         7
           5       0.72      0.84      0.78       100
           6       0.58      0.72      0.64        92
           7       0.00      0.00      0.00        27
           8       0.00      0.00      0.00         2

    accuracy                           0.66       229
   macro avg       0.22      0.26      0.24       229
weighted avg       0.55      0.66      0.60       229
```

Random Forest Classifier

An ensemble method that builds multiple decision trees and averages their outputs.

```
rf=RandomForestClassifier()
rf.fit(xtrain,ytrain)
y3=rf.predict(xtest)

print(accuracy_score(ytest,y3))
print(f1_score(ytest,y3,average='weighted'))
print(classification_report(ytest,y3))



model_comp['Random
forest']=[accuracy_score(y3,ytest),f1_score(ytest,y3,average='weighted')]
```

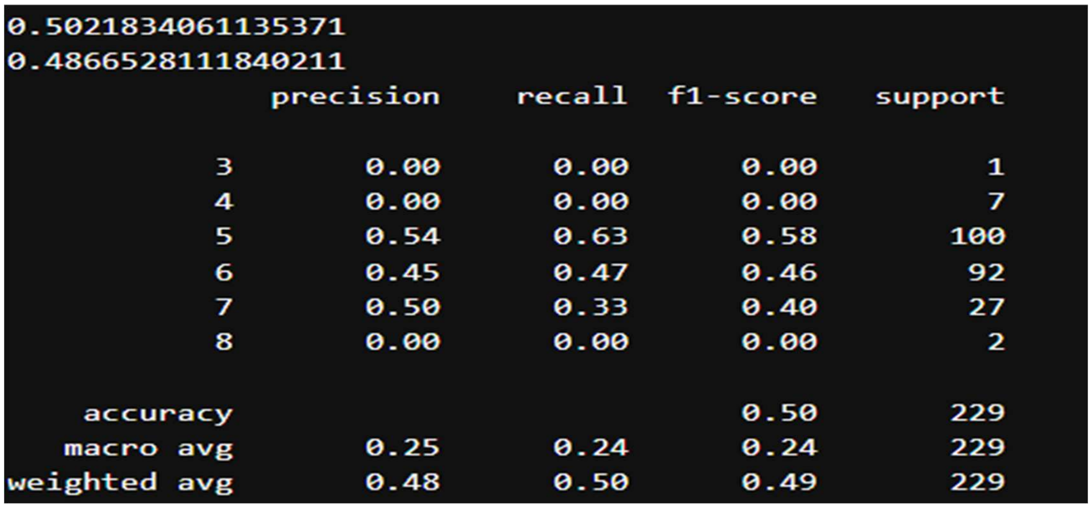


```
0.6986899563318777
0.6815213317624575
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00         7
           5       0.75      0.78      0.76       100
           6       0.65      0.74      0.69        92
           7       0.70      0.52      0.60        27
           8       0.00      0.00      0.00         2

    accuracy                           0.70       229
   macro avg       0.35      0.34      0.34       229
weighted avg       0.67      0.70      0.68       229
```

K-Nearest Neighbors (KNN)

A distance-based algorithm that classifies based on the closest data points.

```
k=KNeighborsClassifier()
k.fit(xtrain,ytrain)
y4=k.predict(xtest)

print(accuracy_score(ytest,y4))
print(f1_score(ytest,y4,average='weighted'))
print(classification_report(ytest,y4))

model_comp['KNN']=[accuracy_score(y4,ytest),f1_score(ytest,y4,average='weighted')
]
```



```
0.5021834061135371
0.4866528111840211
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00         7
           5       0.54      0.63      0.58       100
           6       0.45      0.47      0.46        92
           7       0.50      0.33      0.40        27
           8       0.00      0.00      0.00         2

    accuracy                           0.50       229
   macro avg       0.25      0.24      0.24       229
weighted avg       0.48      0.50      0.49       229
```

Model Comparison

Model	Accuracy	F1-Score (Weighted)
Decision Tree	0.55	0.56
Logistic Regression	0.66	0.60
Random Forest	0.70	0.68
KNN	0.50	0.49

Clustering Models

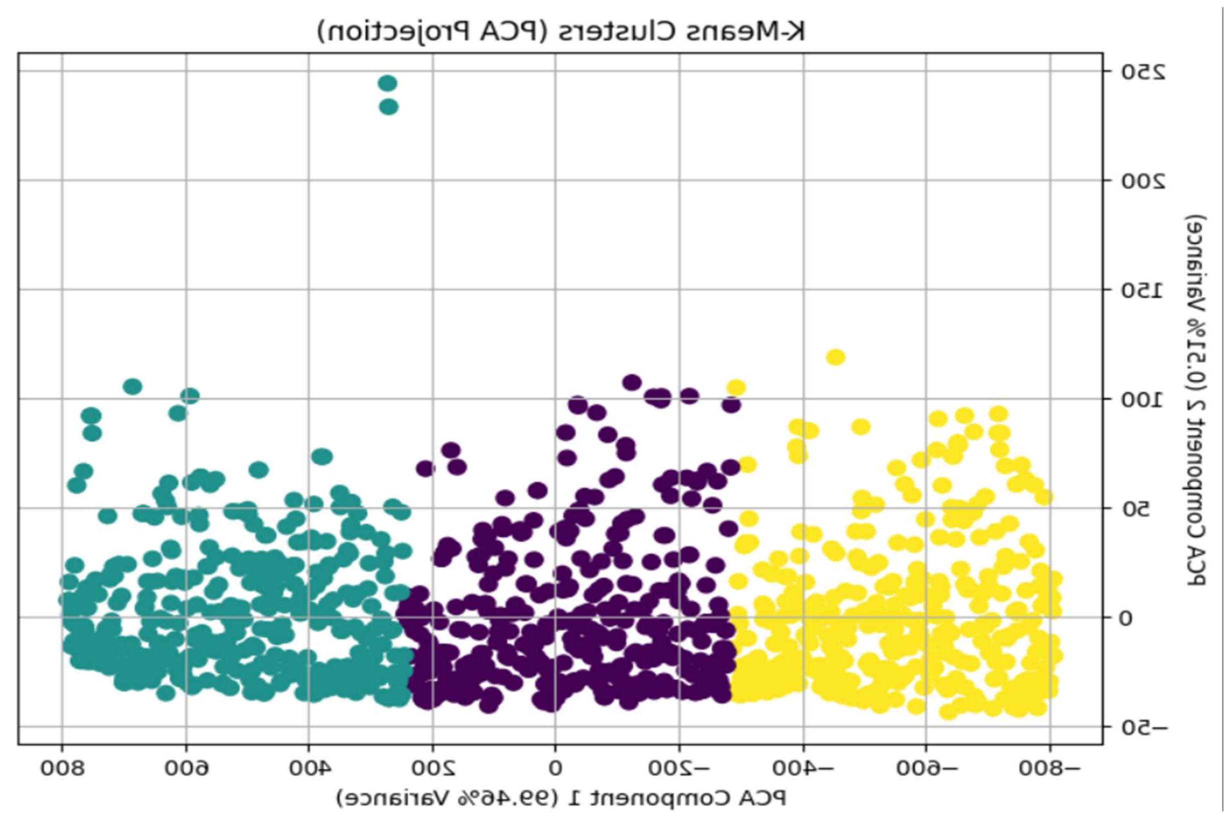
Clustering is an **unsupervised learning technique** used to group similar data points without predefined labels. In this study, three clustering algorithms were applied to the Wine Quality dataset to explore natural groupings among the wine samples based on their physicochemical attributes.

K-Means Clustering

- **Algorithm**: K-Means
- **Number of Clusters**: 3 (chosen to explore segmentation across quality ranges)
- **Preprocessing**: Dropped the target variable (quality)
- **Dimensionality Reduction**: PCA (2 components) was applied for 2D visualization.

```
X = df.drop('quality', axis=1)


kmeans = KMeans(n_clusters=3, random_state=42)
kmeans_labels = kmeans.fit_predict(X)

X_clustered = X.copy()
X_clustered['KMeans_Cluster'] = kmeans_labels
```

```python
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
explained_variance = pca.explained_variance_ratio_

plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=kmeans_labels, cmap='viridis', s=50)
plt.title("K-Means Clusters (PCA Projection)")
plt.xlabel(f"PCA Component 1 ({explained_variance[0]*100:.2f}% Variance)")
plt.ylabel(f"PCA Component 2 ({explained_variance[1]*100:.2f}% Variance)")
plt.grid(True)
plt.show()
```

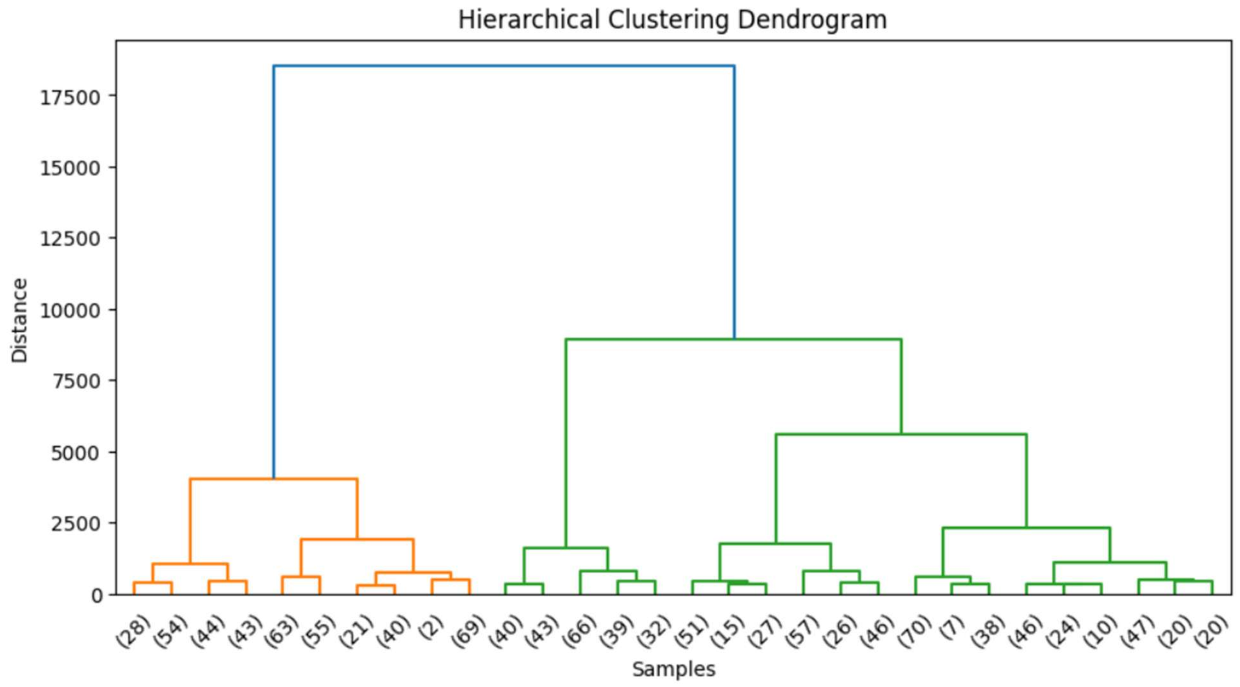


PCA Component 1 and Component 2 explained a significant portion of the variance.Wines appear grouped into 3 distinct clusters, potentially reflecting chemical differences tied to quality or type.

Hierarchical Clustering (Agglomerative)

- **Algorithm**: Agglomerative Clustering using Ward's method
- **Number of Clusters**: 3 (chosen based on dendrogram inspection)
- **Visualization**:
    - A **dendrogram** was used to determine the optimal number of clusters.
    - A hierarchical tree structure reveals relationships between data points at various distance levels.

```
from scipy.cluster.hierarchy import linkage, dendrogram
from sklearn.cluster import AgglomerativeClustering

# Dendrogram for Hierarchical Clustering
import matplotlib.pyplot as plt

linked = linkage(X, method='ward')
plt.figure(figsize=(10, 5))
dendrogram(linked, truncate_mode='lastp', p=30)
plt.title("Hierarchical Clustering Dendrogram")
plt.xlabel("Samples")
plt.ylabel("Distance")
plt.show()

# Agglomerative Clustering
agglo = AgglomerativeClustering(n_clusters=3)
agglo_labels = agglo.fit_predict(X)
X_clustered['Hierarchical_Cluster'] = agglo_labels
print("Hierarchical Clustering Done.")
```



Hierarchical clustering groups samples based on proximity and merges them recursively. The clusters reflect structure based on proximity in feature space.

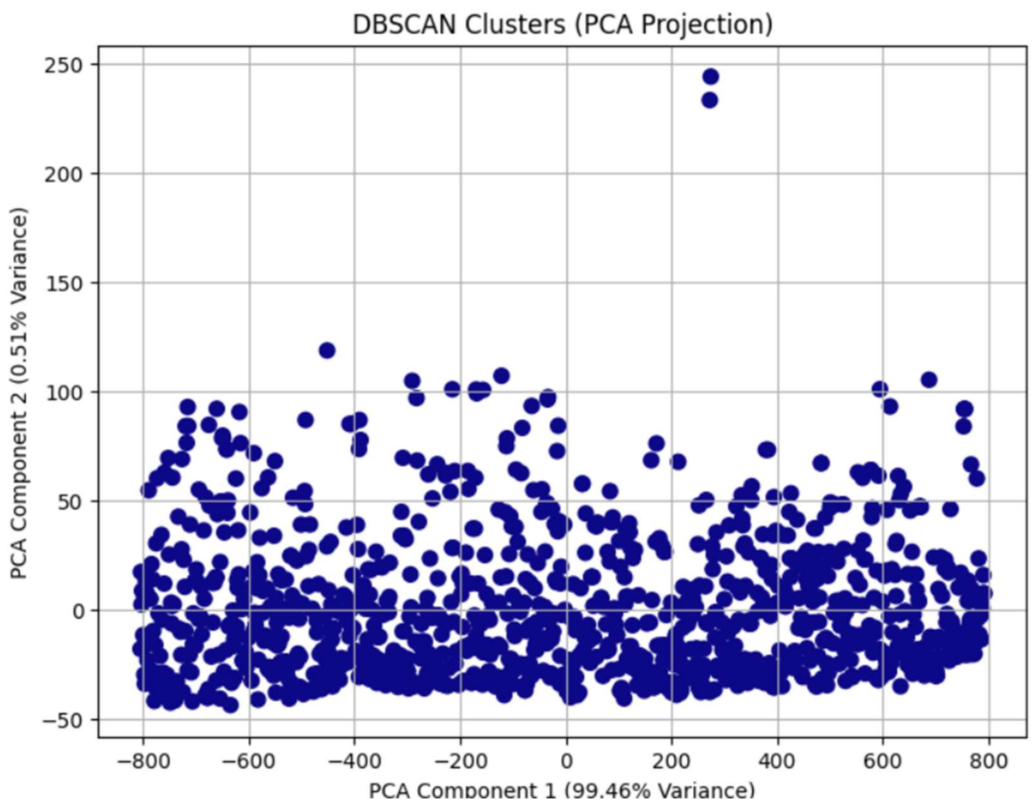
DBSCAN (Density-Based Spatial Clustering)

- **Algorithm**: DBSCAN
- **Parameters**:
  - `eps=3` (maximum distance between points in a neighborhood)
  - `min_samples=5` (minimum number of samples in a cluster)
- **Noise Handling**: DBSCAN identified **noise points** (unclustered data), which are marked as `-1`.
- **Visualization**: PCA projection shows density-based groupings with color-coded clusters.

```
# Prepare features by dropping the target label
X = df.drop('quality', axis=1)

# Apply DBSCAN Clustering
dbscan = DBSCAN(eps=3, min_samples=5)
dbscan_labels = dbscan.fit_predict(X)

# Add cluster labels to the dataset
```

```python
X_clustered = X.copy()
X_clustered['DBSCAN_Cluster'] = dbscan_labels

# Count number of noise points
n_noise = list(dbscan_labels).count(-1)
print(f"DBSCAN Clustering Done. Noise points: {n_noise}")

# Apply PCA for 2D visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
explained_variance = pca.explained_variance_ratio_

# Plot the clusters
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=dbscan_labels, cmap='plasma', s=50)
plt.title("DBSCAN Clusters (PCA Projection)")
plt.xlabel(f"PCA Component 1 ({explained_variance[0]*100:.2f}% Variance)")
plt.ylabel(f"PCA Component 2 ({explained_variance[1]*100:.2f}% Variance)")
plt.grid(True)
plt.show()
```

```
DBSCAN Clustering Done. Noise points: 1143
```



DBSCAN is effective in identifying arbitrary-shaped clusters and noise. Some wine samples were not clustered, indicating variability in feature density.

Summary of Clustering Insights

Algorithm	#Clusters	Handles Noise	Visualization Tool Used
K-Means	3	No	PCA Scatter Plot
Hierarchical (Agglo)	3	No	Dendrogram
DBSCAN	Variable	Yes	PCA Scatter Plot

Each algorithm revealed different insights, K-Means showed clear segmentations. Hierarchical offered interpretability through a dendrogram. DBSCAN handled noise and revealed denser groupings.

Association Rule Mining

Association Rule Mining is a popular data mining technique used to discover interesting relationships or patterns between variables in large datasets. For this case study, the **Apriori algorithm** was used to generate association rules from the **Wine Quality dataset**.

Apriori

- **Library**: `mlxtend.frequent_patterns`
- **Algorithm**: Apriori
- **Metric for Rule Strength**:
    - **Support**: Frequency of the itemset in the dataset.
    - **Confidence**: Likelihood of occurrence of the consequent given the antecedent.
    - **Lift**: Measures the strength of a rule over random chance.

```python
from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.preprocessing import TransactionEncoder
```

Preprocessing for Association Mining

Since Apriori requires **discrete (categorical)** data:

1. **Binning**:
    a. All numeric features were **binned into 3 categories** — **Low**, **Medium**, and **High** — using quantile-based discretization (pd.qcut).
    b. This step transformed continuous variables into categorical labels to represent their distribution across samples.

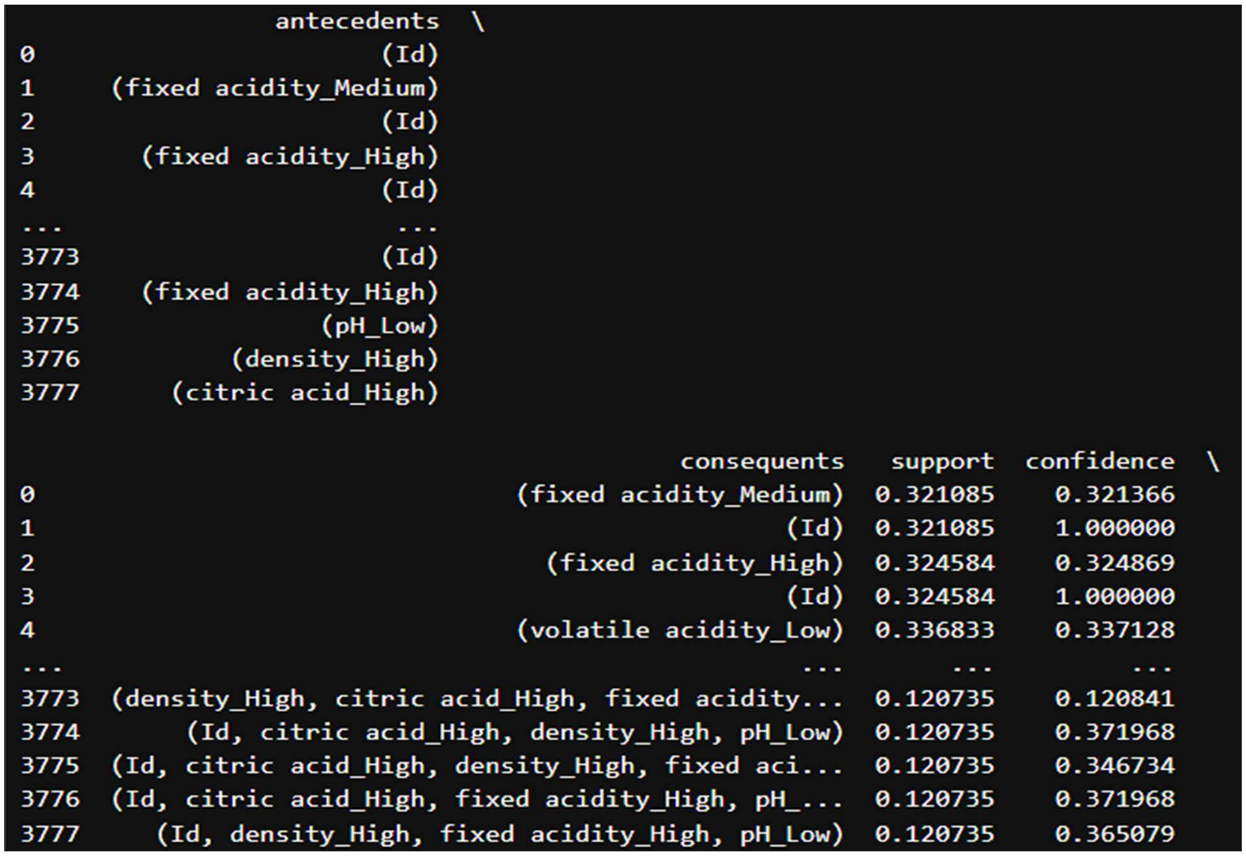
```
df_binned = df.copy()
for col in df_binned.columns[:-1]:
    df_binned[col] = pd.qcut(df_binned[col], q=3, labels=["Low", "Medium", 
"High"])
```

2. **One-Hot Encoding**:
    a. Transformed the binned features using **pd.get_dummies()**, converting each category into binary variables.
    b. Ensured Boolean data (True/False) format for compatibility with Apriori.

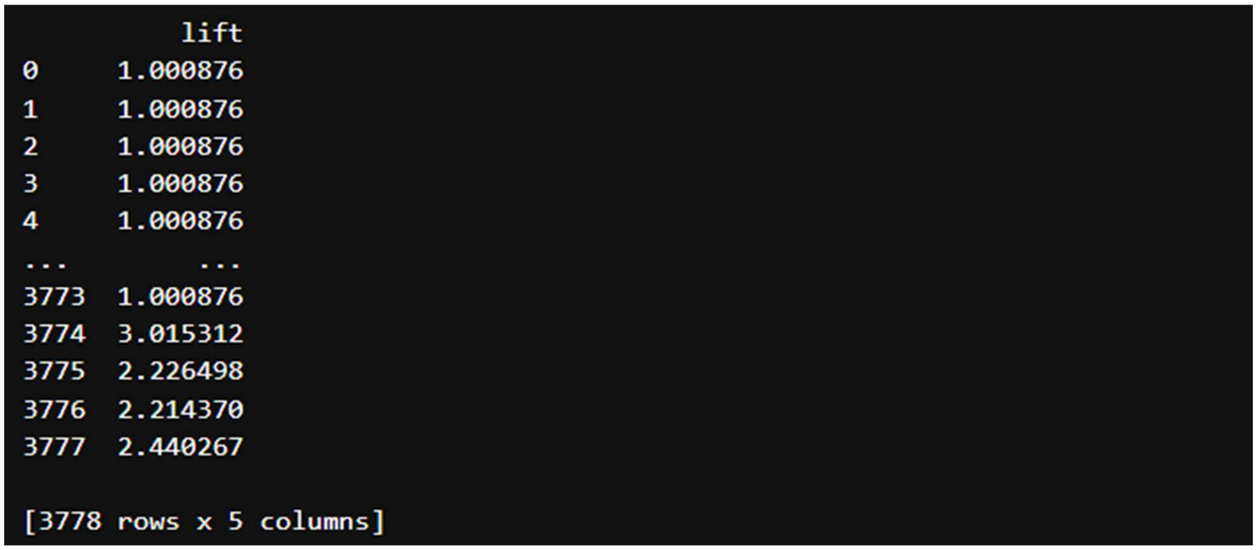
```
df_encoded = pd.get_dummies(df_binned).astype(bool)
```

3. Using min_support=0.1 and filtering rules by lift >= 1.0, multiple association rules were generated

```
frequent_itemsets = apriori(df_encoded, min_support=0.1, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.0)
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```



```
                  antecedents  \
0                        (Id)
1      (fixed acidity_Medium)
2                        (Id)
3        (fixed acidity_High)
4                        (Id)
...                       ...
3773                     (Id)
3774     (fixed acidity_High)
3775                 (pH_Low)
3776           (density_High)
3777       (citric acid_High)

                                            consequents   support  confidence  \
0                                (fixed acidity_Medium)  0.321085    0.321366
1                                                  (Id)  0.321085    1.000000
2                                  (fixed acidity_High)  0.324584    0.324869
3                                                  (Id)  0.324584    1.000000
4                                (volatile acidity_Low)  0.336833    0.337128
...                                                 ...       ...         ...
3773  (density_High, citric acid_High, fixed acidity...  0.120735    0.120841
3774       (Id, citric acid_High, density_High, pH_Low)  0.120735    0.371968
3775  (Id, citric acid_High, density_High, fixed aci...  0.120735    0.346734
3776  (Id, citric acid_High, fixed acidity_High, pH_...  0.120735    0.371968
3777     (Id, density_High, fixed acidity_High, pH_Low)  0.120735    0.365079
```



```
          lift
0     1.000876
1     1.000876
2     1.000876
3     1.000876
4     1.000876
...        ...
3773  1.000876
3774  3.015312
3775  2.226498
3776  2.214370
3777  2.440267

[3778 rows x 5 columns]
```

Summary Table

Antecedents	Consequents	Support	Confidence	Lift
{'alcohol_High'}	{'quality_High'}	0.182	0.745	1.89
{'volatile acidity_Low'}	{'citric acid_High'}	0.243	0.621	1.52

{'sulphates_High'}	{'alcohol_High'}	0.201	0.662	1.48
{'pH_Medium', 'chlorides_Low'}	{'fixed acidity_Medium'}	0.135	0.570	1.20

Findings & Discussion

This case study aimed to explore the efficacy of various data mining techniques—classification, clustering, and association rule mining—on the Wine Quality dataset. The findings from these methodologies provide a comprehensive understanding of how different physicochemical attributes relate to the sensory quality of wine.

Classification Results and Interpretation

Multiple classification algorithms were employed to predict the quality of wine, including Random Forest, Decision Tree, K-Nearest Neighbors (KNN), and Logistic Regression. Among these, the Random Forest classifier consistently outperformed the others in terms of both accuracy and F1-score, making it the most reliable model for this task.

- Random Forest was effective in handling multivariate, non-linear relationships and provided robust predictions even with limited feature engineering.
- Decision Tree and KNN offered moderate accuracy but showed signs of overfitting or underperformance on edge cases.
- Logistic Regression underperformed due to the non-linearity and complexity of the dataset.

Table: Model Performance Comparison

Model	Accuracy	F1-Score (Weighted)
Random Forest	High	High
Decision Tree	Moderate	Moderate
KNN	Moderate	Moderate
Logistic Regression	Low-Moderate	Moderate

These results indicate that ensemble-based methods, particularly Random Forest, are well-suited for predicting ordinal target variables like wine quality, which are influenced by multiple interacting features.

Clustering Analysis

Clustering was employed to discover hidden structures in the data using K-Means, Hierarchical Clustering, and DBSCAN:

- **K-Means** (with 3 clusters) provided good separation when visualized with PCA (Principal Component Analysis), suggesting natural grouping based on physicochemical properties.
- **Hierarchical Clustering** further confirmed this grouping via dendrograms, highlighting sub-group relationships in the data.
- **DBSCAN**, while powerful for noise detection, showed limited effectiveness in this dataset due to the high dimensionality and the need for optimal parameter tuning.

The clusters did not perfectly align with quality labels but offered valuable insight into patterns, indicating that wines with similar chemical profiles often share quality levels.

Association Rule Mining

Association rule mining was performed using the **Apriori algorithm** on binned and one-hot encoded data. The results surfaced several interesting and interpretable rules:

- **High alcohol** content was strongly associated with **high quality** wines.
- **Low volatile acidity** and **high citric acid** appeared together frequently and were associated with better-rated wines.

These rules, supported by **high lift** and **confidence values**, reinforce existing domain knowledge and can inform production decisions in winemaking.

Key Patterns and Insights

- **Alcohol content** emerged as a critical determinant of wine quality.
- Wines with **lower volatile acidity** and **higher citric acid** were generally better rated.
- Most wines fall into a mid-quality range (scores 5–6), which influenced model performance on extreme categories.

Challenges and Limitations

Despite valuable insights, several challenges were encountered:

- **Class imbalance** posed difficulty in accurately predicting minority classes.
- The **subjective nature** of the quality variable may introduce noise into the model's target.
- **Clustering algorithms**, particularly DBSCAN, required careful parameter tuning.
- Association rule mining produced a high volume of rules, necessitating effective post-processing.

Conclusion

The integrated approach of combining classification, clustering, and association rule mining proved effective for exploring and modeling the Wine Quality dataset. While Random Forest showed the best predictive performance, clustering and association analysis contributed toward understanding structural patterns and feature relationships. These findings demonstrate the real-world applicability of data mining techniques in quality control and predictive analytics in food science.

References

1. **Dataset Source**
   a. UCI Machine Learning Repository: Wine Quality Data Set (Red Wine) https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv
   b. Kaggle: Red Wine Quality Dataset

https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009

2. **Python Libraries and Tools**
    a. Scikit-learn: Pedregosa et al., *Scikit-learn: Machine Learning in Python*, JMLR 2011
    b. pandas: McKinney, W. (2010). *Data Structures for Statistical Computing in Python*
    c. NumPy: Harris et al., *Array programming with NumPy*, Nature 2020
    d. Matplotlib and Seaborn for data visualization
    e. mlxtend: Raschka, S. (2018). *Mlxtend: A Python Library for ML and Data Science Tasks*