Exploring Wine Classification

with K-Nearest Neighbors: A Comprehensive Analysis

Presented by TEAM UNITY

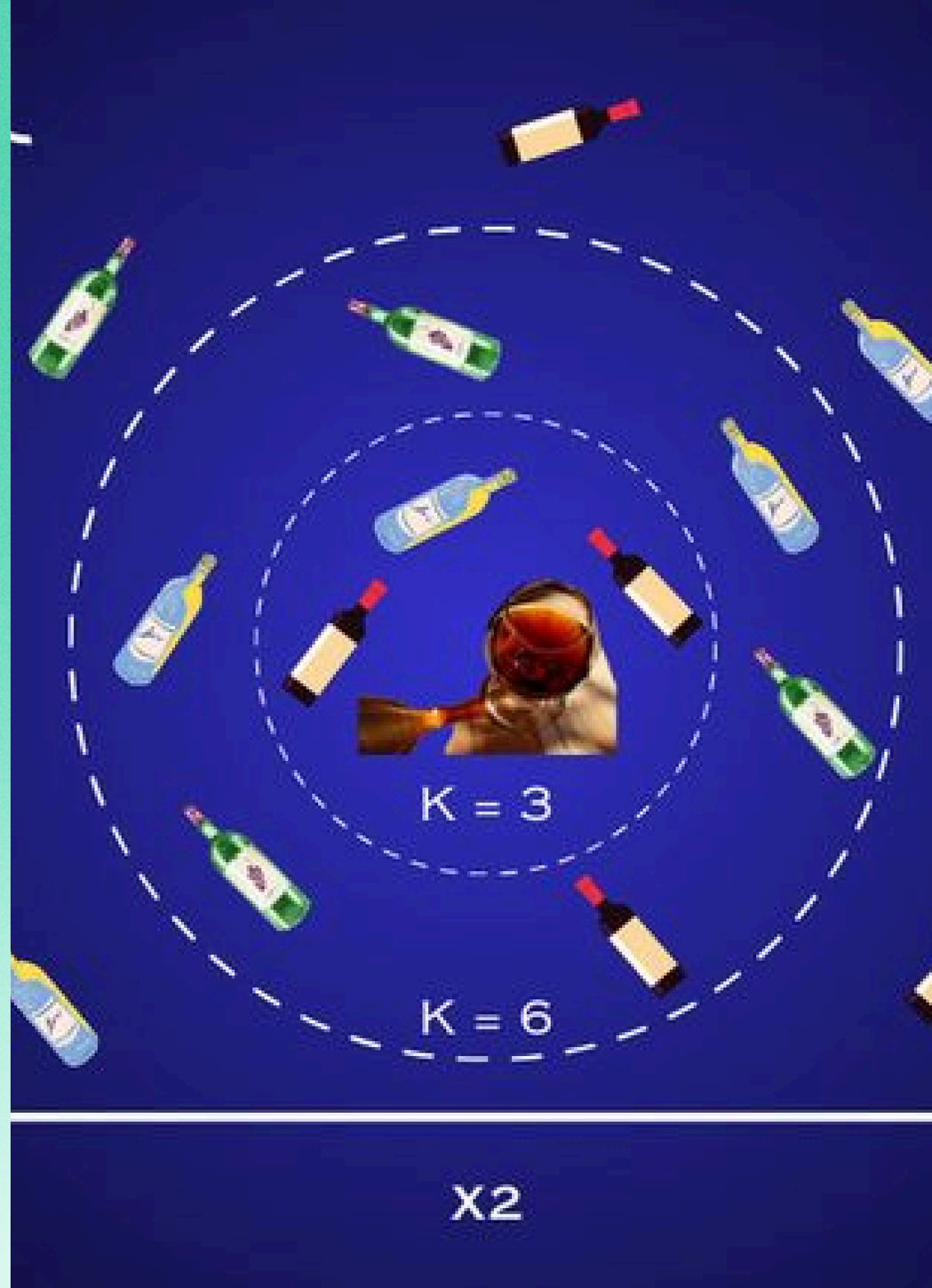
AP21110010200 - Uday Sai Chaganti

AP21110010219 - Sai Rohith

AP21110010228 - Jithin Atluri

AP21110010253 - Chaitanya Sai

AP21110010260 - Ritesh Jadhav



Contents

Abstract

- This project focuses on exploring the Wine dataset, a classic dataset widely used in the field of machine learning.

- The primary objective is to conduct a thorough exploratory analysis to gain insights into the dataset's characteristics and distribution.

- The dataset contains various attributes related to wine samples, such as chemical composition and origin.

- Although the dataset doesn't inherently pose a specific problem, the goal is to leverage data analysis techniques to extract meaningful insights and knowledge.

- Subsequently, the K-Nearest Neighbors (KNN) algorithm will be implemented for classification tasks based on the dataset's features.

- By applying KNN, the aim is to classify wine samples into predefined classes based on their similarity to neighboring samples.

- This project serves as an educational exercise to understand data exploration, feature analysis, and the application of the KNN algorithm in a simple yet effective manner.

Introduction

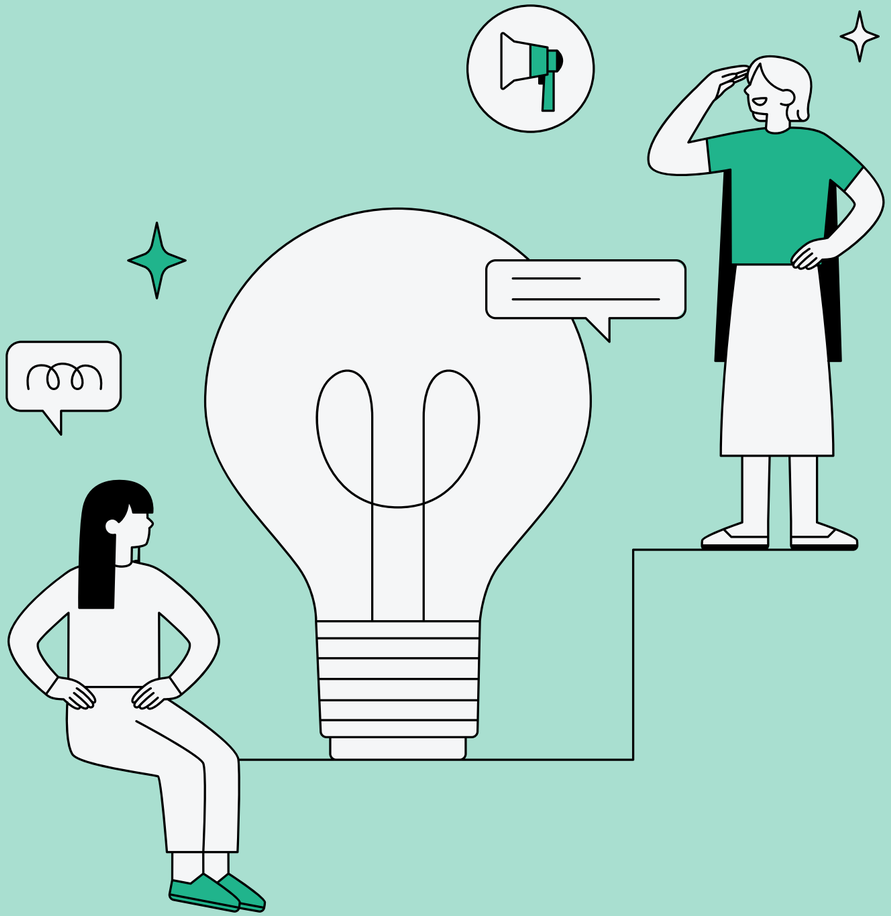
- In this research, we delve into the Wine dataset, a renowned source of viticultural data widely used in machine learning.

- Our objective is twofold: conducting a comprehensive exploratory analysis of the dataset and implementing the K-Nearest Neighbors (KNN) algorithm for wine classification.

- Through rigorous data analysis techniques, we aim to extract meaningful insights relevant to classifying wine samples based on their chemical attributes.

- The central challenge addressed is the accurate identification of wine origins, crucial for industry practitioners and consumers alike.

- By leveraging the KNN algorithm's pattern recognition capabilities, we seek to develop a robust classification model capable of pinpointing the geographical provenance of wine samples with high accuracy.



Dataset Overview

The Wine dataset consists of 14 features, one of which is the target, and 178 labels. All the data is numerical, with no missing values or duplicates. The nomenclature dictionary includes:

Attribute	Definition	Data Type
alcohol	Alcohol content	float64
malic_acid	Malic acid	float64
ash	Ash	float64
alcalinity_of_ash	Alkalinity of ash	float64
magnesium	Magnesium	float64
total_phenols	Total phenols	float64
flavanoids	Flavanoids	float64
nonflavanoid_phenols	Non-flavanoid phenols	float64
proanthocyanins	Proanthocyanins	float64
color_intensity	Color intensity	float64
hue	Hue	float64
od280/od315_of_diluted_wines	OD280/OD315 of diluted wines	float64
proline	Proline	float64
class	Wine type (target)	int32

Methodology

01. Exploratory Data Analysis

02. Pre-Processing

Exploratory Data Analysis

```
In [19]: df_wine.describe(include="all")
```

Out[19]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthoc
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.0
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	0.361854	1.5
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453	0.5
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000	0.4
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000	1.2
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000	1.5
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500	1.9
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000	3.5

```
df_wine.head()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82

```
In [21]: df_wine.duplicated().any()

Out[21]: False
```

```
In [22]: df_wine.apply(lambda x: len(x.unique()))

Out[22]: alcohol                        126
         malic_acid                     133
         ash                             79
         alcalinity_of_ash               63
         magnesium                       53
         total_phenols                   97
         flavanoids                     132
         nonflavanoid_phenols            39
         proanthocyanins                101
         color_intensity                132
         hue                             78
         od280/od315_of_diluted_wines   122
         proline                        121
         class                            3
         dtype: int64
```

Features values count

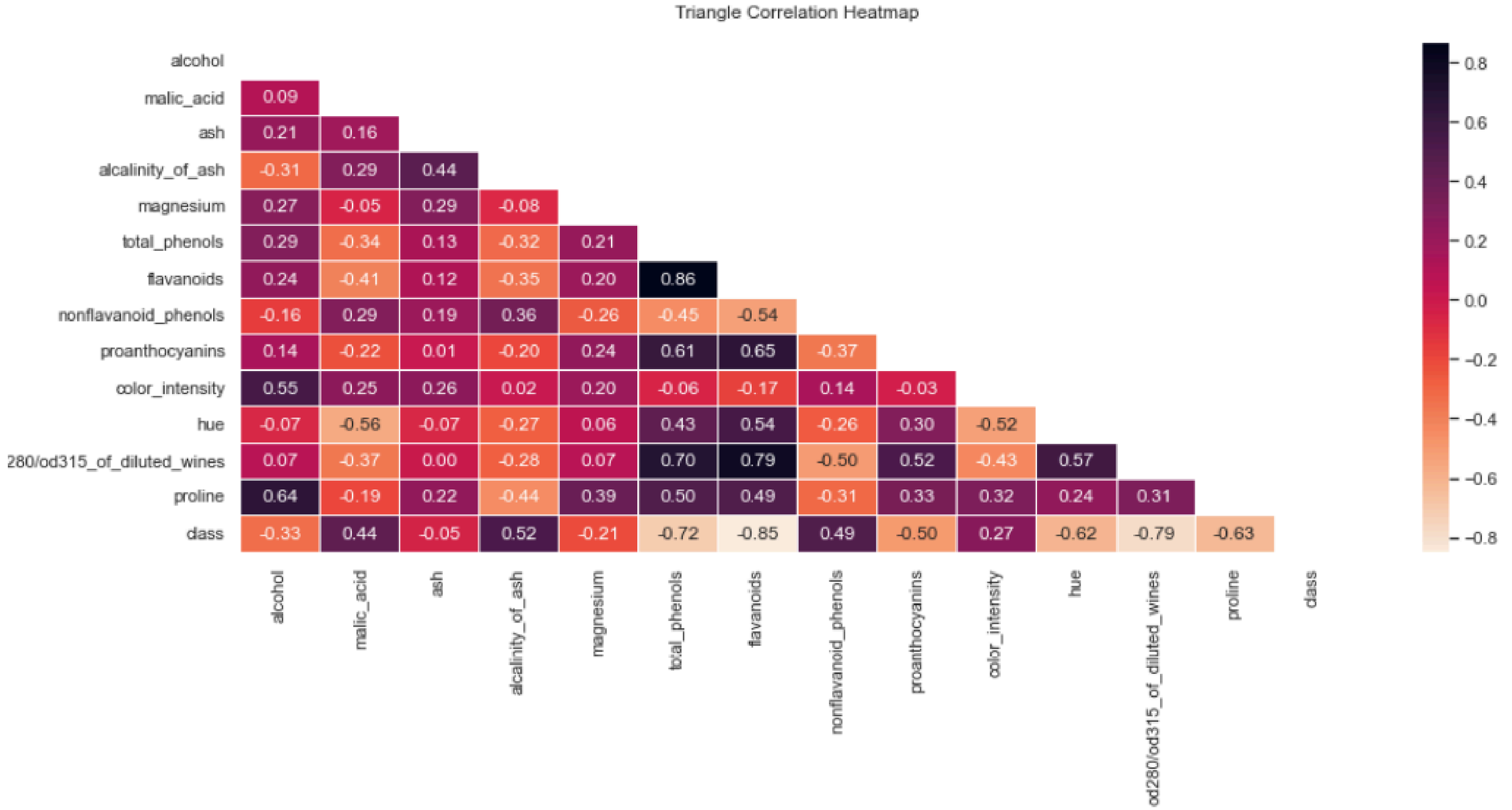
In [24]:

```
for feature in df_wine.columns:
    print('-------------------------')
    print(f'Series: {feature}')
    print('-------------------------')
    print(f'{df_wine[feature].value_counts()}\n')
```

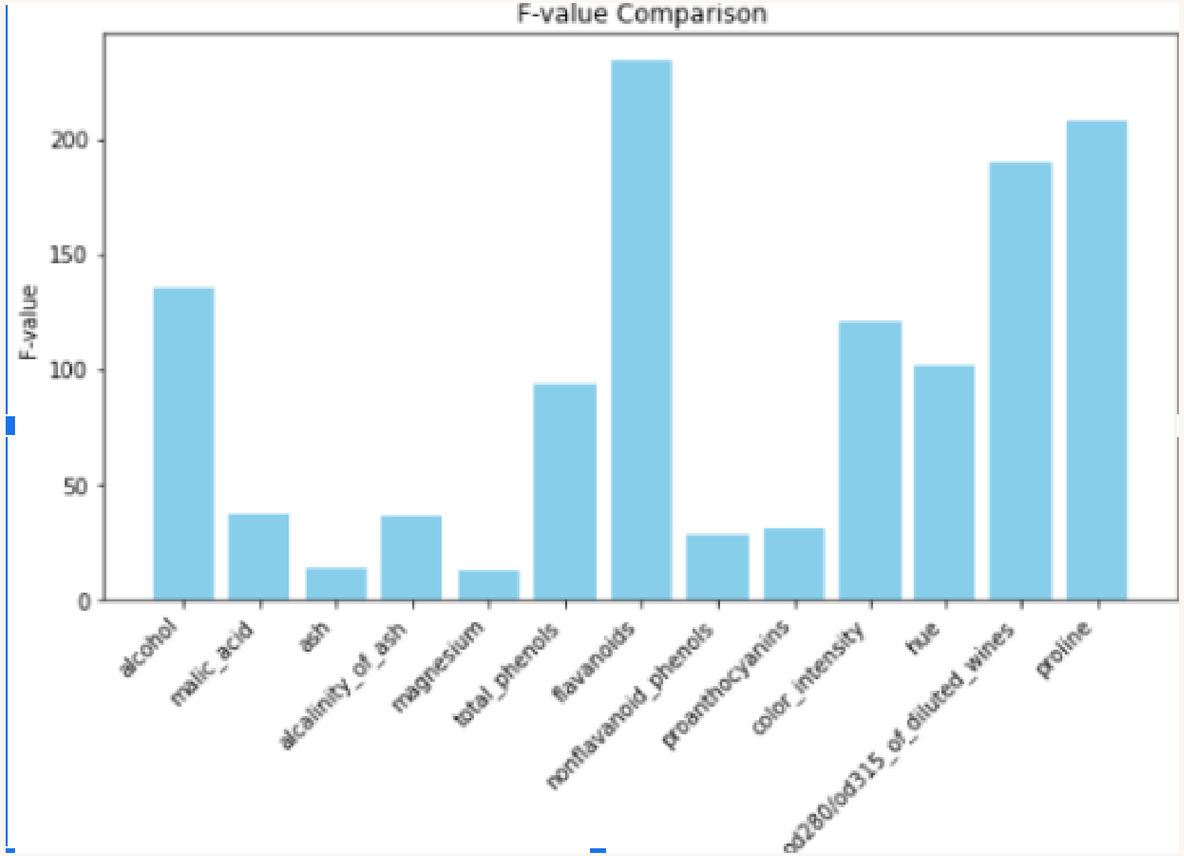
```
-------------------------
Series: alcohol
-------------------------
alcohol
13.05    6
12.37    6
12.08    5
12.29    4
12.42    3
        ..
13.72    1
13.29    1
13.74    1
13.77    1
14.13    1
Name: count, Length: 126, dtype: int64
```

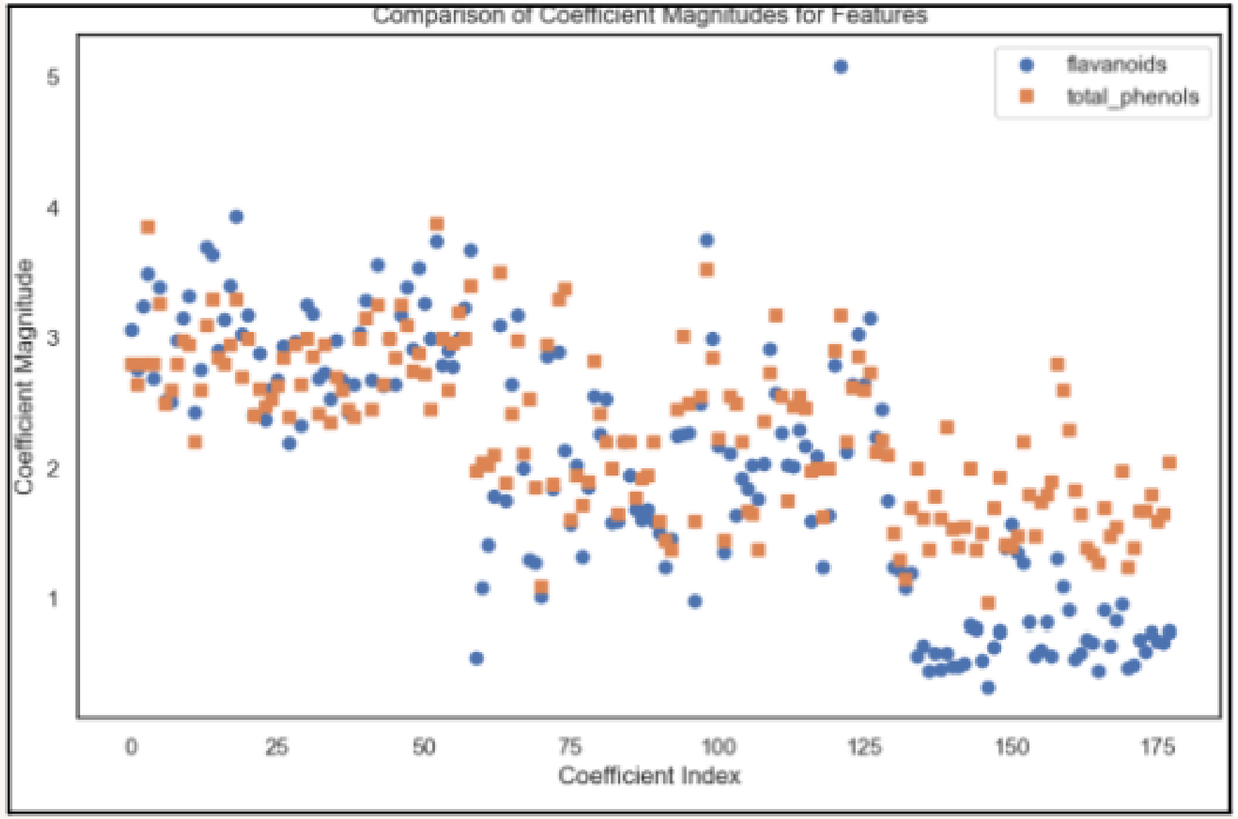
Check Correlation

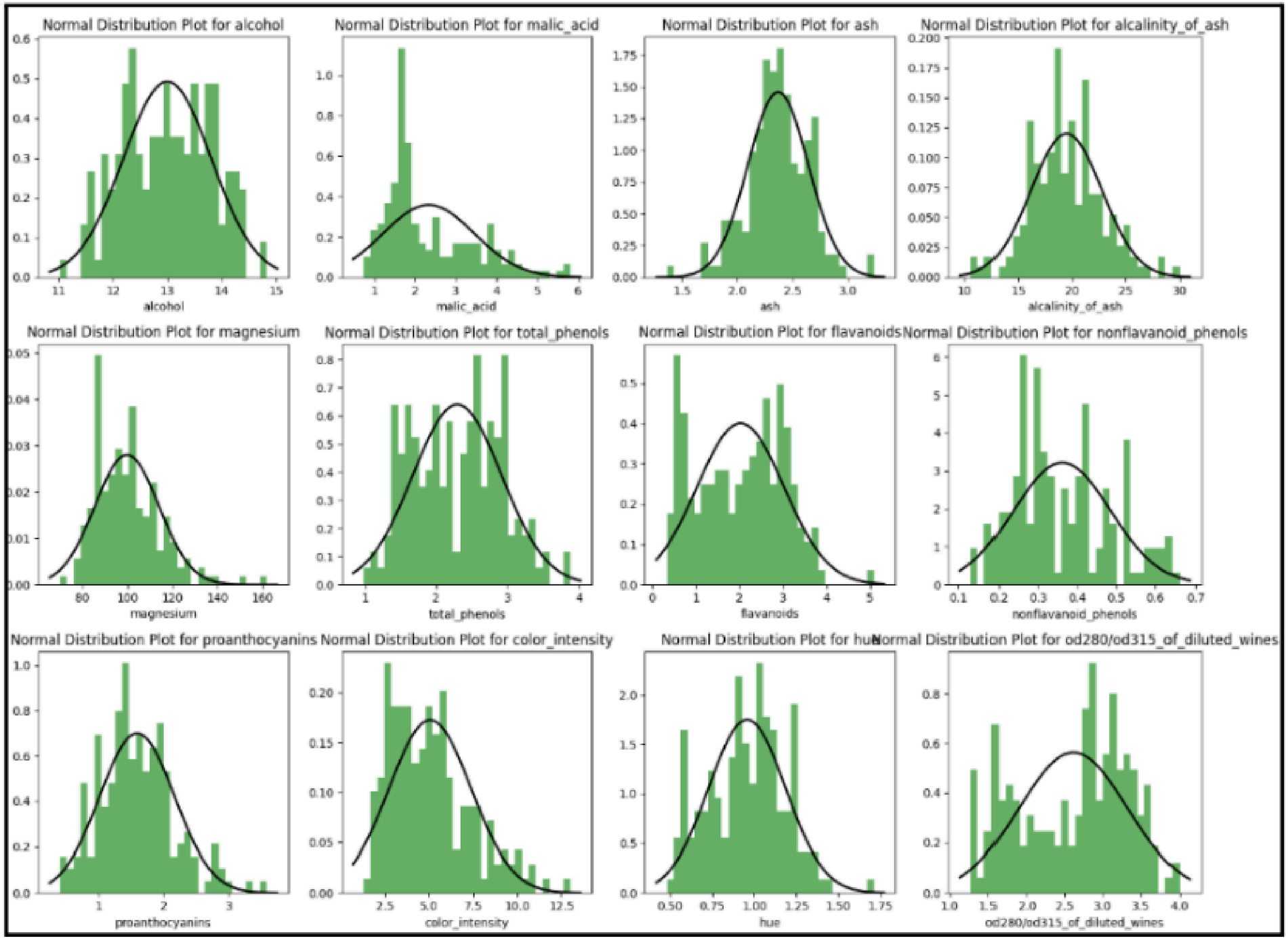
In [26]:
```
plot_triangle_correlation_heatmap(df_wine)
```



Pre-Processing






Normal Distribution Plot for alcohol
alcohol
Normal Distribution Plot for malic_acid
malic_acid
Normal Distribution Plot for ash
ash
Normal Distribution Plot for alcalinity_of_ash
alcalinity_of_ash
Normal Distribution Plot for magnesium
magnesium
Normal Distribution Plot for total_phenols
total_phenols
Normal Distribution Plot for flavanoids
flavanoids
Normal Distribution Plot for nonflavanoid_phenols
nonflavanoid_phenols
Normal Distribution Plot for proanthocyanins
proanthocyanins
Normal Distribution Plot for color_intensity
color_intensity
Normal Distribution Plot for hue
hue
Normal Distribution Plot for od280/od315_of_diluted_wines
od280/od315_of_diluted_wines

Machine Learning (KNN)

01. Normalization

02. Standardization

Normalization

Normalization and standardization are preprocessing techniques commonly used in machine learning, including KNN (K-Nearest Neighbors) classification, to improve the performance of the algorithms.

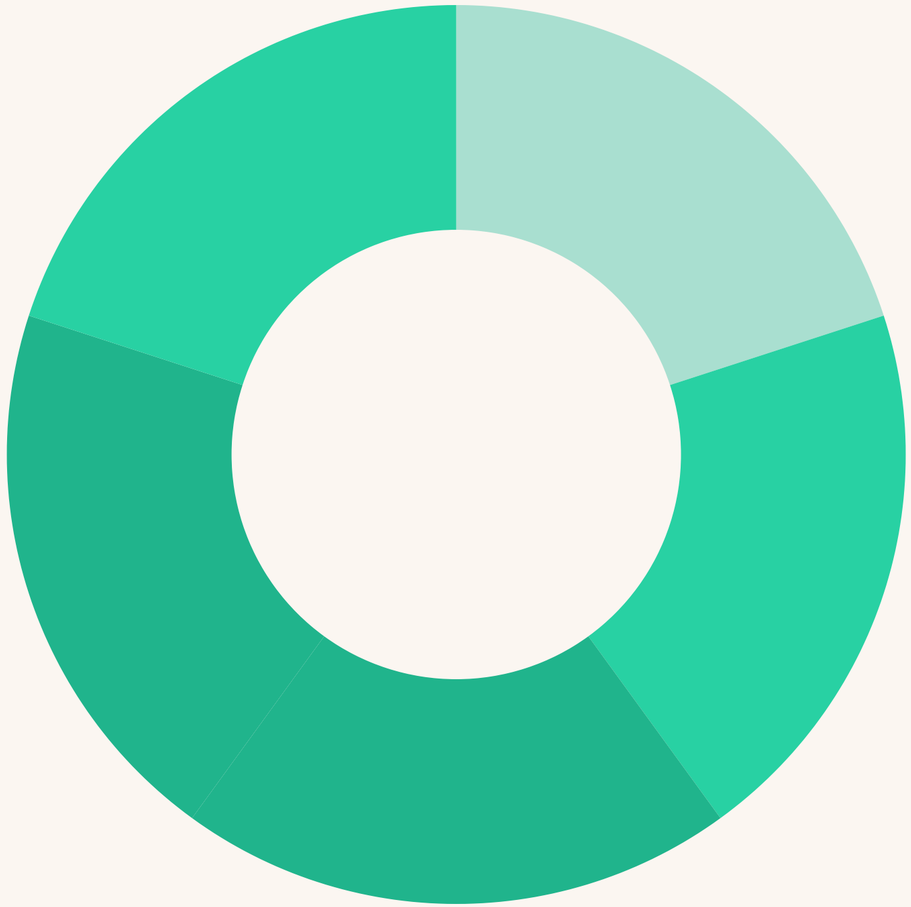
- Scaling numeric features to a uniform range, typically [0, 1].

- Normalization ensures all features contribute equally to distance computations in KNN.

- Without normalization, features with larger scales may dominate distance calculations.

- Normalization brings features to a similar scale, preventing bias in distance computations.

- It prevents features with larger numeric ranges from outweighing those with smaller ranges.



Standardization:

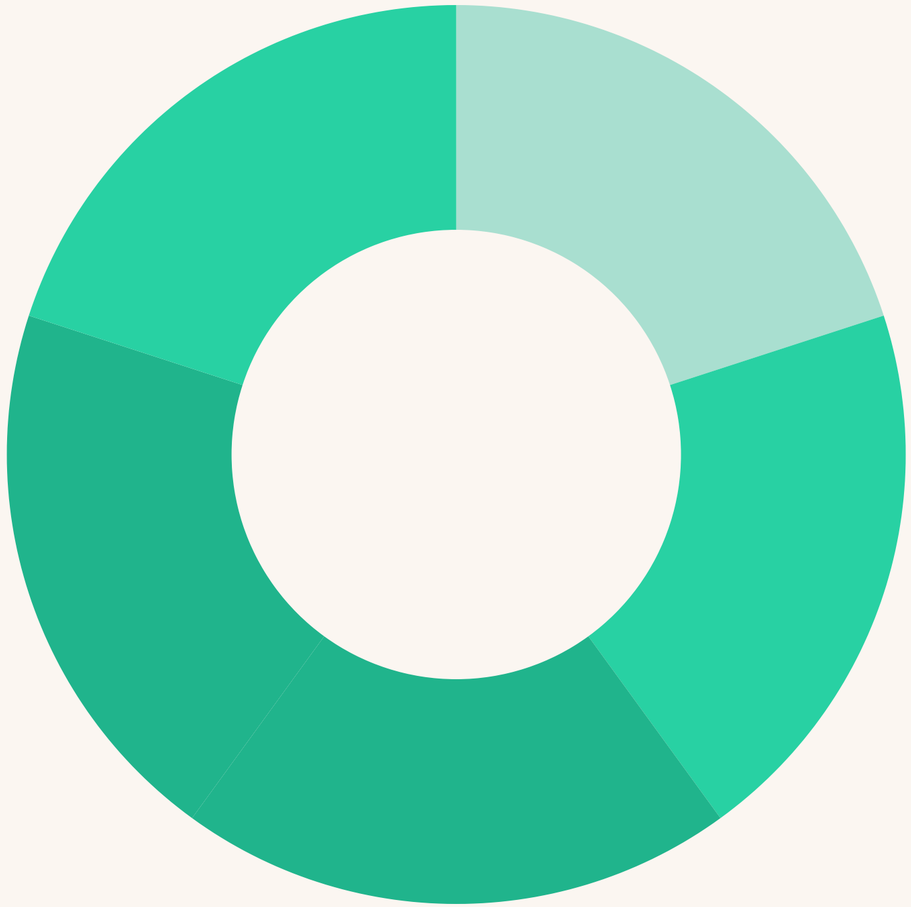
- Standardization transforms features to have a mean of 0 and a standard deviation of 1 (z-score normalization).

- It makes features comparable by scaling them to a standard scale.

- Useful in KNN when features have different units or scales.

- Ensures features are centered around zero with a comparable spread, enhancing the performance of distance-based algorithms.

- Less sensitive to outliers compared to normalization, as it centers the data around the mean.



Result (Normalization)

```
---------------------------------
Train Confusion Matrix | Normalized
---------------------------------
Prediction   0   1   2  All
Real
0           44   0   0   44
1            2  42   2   46
2            0   0  30   30
All         46  42  32  120
---------------------------------
Train Score: 0.9666666666666667
```

```
---------------------------------
Test Confusion Matrix | Normalized
---------------------------------
Prediction   0   1   2  All
Real
0           14   0   0   14
1            0  15   0   15
2            0   0  12   12
All         14  15  12   41
Test Score: 1.0
```

```
-----------------------------------------------------
Test Classification Metrics | Normalized
-----------------------------------------------------
              precision    recall  f1-score   support

     class_0       1.00      1.00      1.00        14
     class_1       1.00      1.00      1.00        15
     class_2       1.00      1.00      1.00        12

    accuracy                           1.00        41
   macro avg       1.00      1.00      1.00        41
weighted avg       1.00      1.00      1.00        41
```

Result (Standardization)

```
---------------------------------
Train Confusion Matrix | Standardized
---------------------------------
Prediction   0   1   2  All
Real
0           44   0   0   44
1            2  41   3   46
2            0   0  30   30
All         46  41  33  120
---------------------------------
Train Score: 0.9583333333333334
```

```
---------------------------------
Test Confusion Matrix | Standardized
---------------------------------
Prediction   0   1   2  All
Real
0           14   0   0   14
1            0  15   0   15
2            0   1  11   12
All         14  16  11   41
Test Score: 0.975609756097561
```

```
-----------------------------------------------------
Test Classification Metrics | Standardized
-----------------------------------------------------
              precision    recall  f1-score   support

     class_0       1.00      1.00      1.00        14
     class_1       0.94      1.00      0.97        15
     class_2       1.00      0.92      0.96        12

    accuracy                           0.98        41
   macro avg       0.98      0.97      0.97        41
weighted avg       0.98      0.98      0.98        41
```

Result (RandomForest)

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

def train_and_evaluate_random_forest(scale_name, X, y, n_estimators=100, random_state=37):
    rf = RandomForestClassifier(n_estimators=n_estimators, random_state=random_state)

    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=random_state)


    rf.fit(X_train, y_train)


    y_pred = rf.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)

    print(f'Accuracy of Random Forest classifier ({scale_name}): {accuracy:.2f}')


X = df_wine.drop(columns=['class'])
y = df_wine['class']

train_and_evaluate_random_forest('Original', X, y)
```

```
Accuracy of Random Forest classifier (Original): 0.96
```

Conclusion

Both normalization and standardization resulted in test and training set accuracies above 95%, suggesting various factors contributing to this outcome:

1. Overfitting:
   Achieving 100% accuracy on the test set may indicate overfitting. When a model fits the training data too closely, it struggles to generalize to new data, memorizing patterns instead of learning from them.

2. Size of the Test Set:
   - The small size of the test set could contribute to the high accuracy observed. A small test set may inadvertently align well with the nearest neighbors in KNN, leading to seemingly optimal performance without truly reflecting the model's capability.

3. Good Separability:
   - KNN performs well when data in the test set are easily separable. Clear distinctions between groups and well-spread test examples facilitate accurate predictions as KNN relies on proximity for classification..

4. Proper Pre-processing:

- Effective data pre-processing, including normalization, outlier treatment, and appropriate preparation of training and test data, significantly impacts model performance. Ensuring proper data treatment enhances the model's ability to learn and generalize from the data.

5. Appropriate Choice of Hyperparameters:

- Hyperparameters like the number of neighbors (k) play a crucial role in KNN's performance. Selecting suitable hyperparameters is essential, as they can greatly influence the model's behavior and ability to generalize effectively.

Considering these factors collectively, achieving high accuracy in both training and test sets reflects not only the effectiveness of pre-processing techniques but also the interplay of various factors influencing KNN's performance.

Thank you very much!