In [16]:
```
# Ashishkumar Pemmaraju
# DS 489 - Final Project Python Code
# WIU ID: 923-19-4632
```

In [17]:
```
#Importing Packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#Setting up asethetics
pd.set_option("display.float_format", lambda x: "%.4f" % x)
np.set_printoptions(precision=4, suppress=True)
def print(*args):
    __builtins__.print(*("%.3f" % a if isinstance(a, float) else a
                         for a in args))
```

In [18]:
```
#Reading Data
data = pd.read_csv(r"C:\Users\91973\Desktop\Ds489 Final\wine.data.csv")

#Feature PreProcessing - Removing Missing Values
missing_values = data.isnull().sum()
print("Missing values in each column:")
print(missing_values)
data=data.dropna()
data.head()
```

```
Missing values in each column:
Alcohol                         0
Malic Acid                      0
Ash                             0
Alcalinity of ash               0
Magnesium                       0
Total Phenols                   0
Flavanoids                      0
Nonflavanoid phenols            0
Proanthocyanins                 0
Color Intensity                 0
Hue                             0
OD280/OD315 of diluted wines    0
Proline                         0
Number of bottles               0
dtype: int64
```

Out[18]:

	Alcohol	Malic Acid	Ash	Alcalinity of ash	Magnesium	Total Phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanir
0	14.2300	1.7100	2.4300	15.6000	127	2.8000	3.0600	0.2800	2.29C
1	13.2000	1.7800	2.1400	11.2000	100	2.6500	2.7600	0.2600	1.28C
2	13.1600	2.3600	2.6700	18.6000	101	2.8000	3.2400	0.3000	2.81C
3	14.3700	1.9500	2.5000	16.8000	113	3.8500	3.4900	0.2400	2.18C
4	13.2400	2.5900	2.8700	21.0000	118	2.8000	2.6900	0.3900	1.82C

In [19]:

```
# Feature Selection -  Removing low variance columns

from sklearn import feature_selection

low_var = feature_selection.VarianceThreshold(threshold=0.05) #traditional value consi
low_var.fit_transform(data)
feature_var = data[data.columns[low_var.get_support(indices=True)]]

#Columns names dropped
drop_var = list(set(data.columns).difference(set(feature_var.columns)))

#Printing new feature data
print(feature_var)
print(drop_var)
```

```
     Alcohol  Malic Acid    Ash  Alcalinity of ash  Magnesium  Total Phenols  \
0    14.2300      1.7100 2.4300            15.6000        127         2.8000
1    13.2000      1.7800 2.1400            11.2000        100         2.6500
2    13.1600      2.3600 2.6700            18.6000        101         2.8000
3    14.3700      1.9500 2.5000            16.8000        113         3.8500
4    13.2400      2.5900 2.8700            21.0000        118         2.8000
..       ...         ...    ...                ...        ...            ...
173  13.7100      5.6500 2.4500            20.5000         95         1.6800
174  13.4000      3.9100 2.4800            23.0000        102         1.8000
175  13.2700      4.2800 2.2600            20.0000        120         1.5900
176  13.1700      2.5900 2.3700            20.0000        120         1.6500
177  14.1300      4.1000 2.7400            24.5000         96         2.0500

     Flavanoids  Proanthocyanins  Color Intensity    Hue  \
0        3.0600           2.2900           5.6400 1.0400
1        2.7600           1.2800           4.3800 1.0500
2        3.2400           2.8100           5.6800 1.0300
3        3.4900           2.1800           7.8000 0.8600
4        2.6900           1.8200           4.3200 1.0400
..          ...              ...              ...    ...
173      0.6100           1.0600           7.7000 0.6400
174      0.7500           1.4100           7.3000 0.7000
175      0.6900           1.3500          10.2000 0.5900
176      0.6800           1.4600           9.3000 0.6000
177      0.7600           1.3500           9.2000 0.6100

     OD280/OD315 of diluted wines  Proline          Number of bottles
0                          3.9200            1065                  23
1                          3.4000            1050                 123
2                          3.1700            1185                 137
3                          3.4500            1480                 109
4                          2.9300             735                 130
..                            ...             ...                 ...
173                        1.7400             740                  95
174                        1.5600             750                 115
175                        1.5600             835                   7
176                        1.6200             840                  43
177                        1.6000             560                  94

[178 rows x 13 columns]
['Nonflavanoid phenols']
```

In [20]:

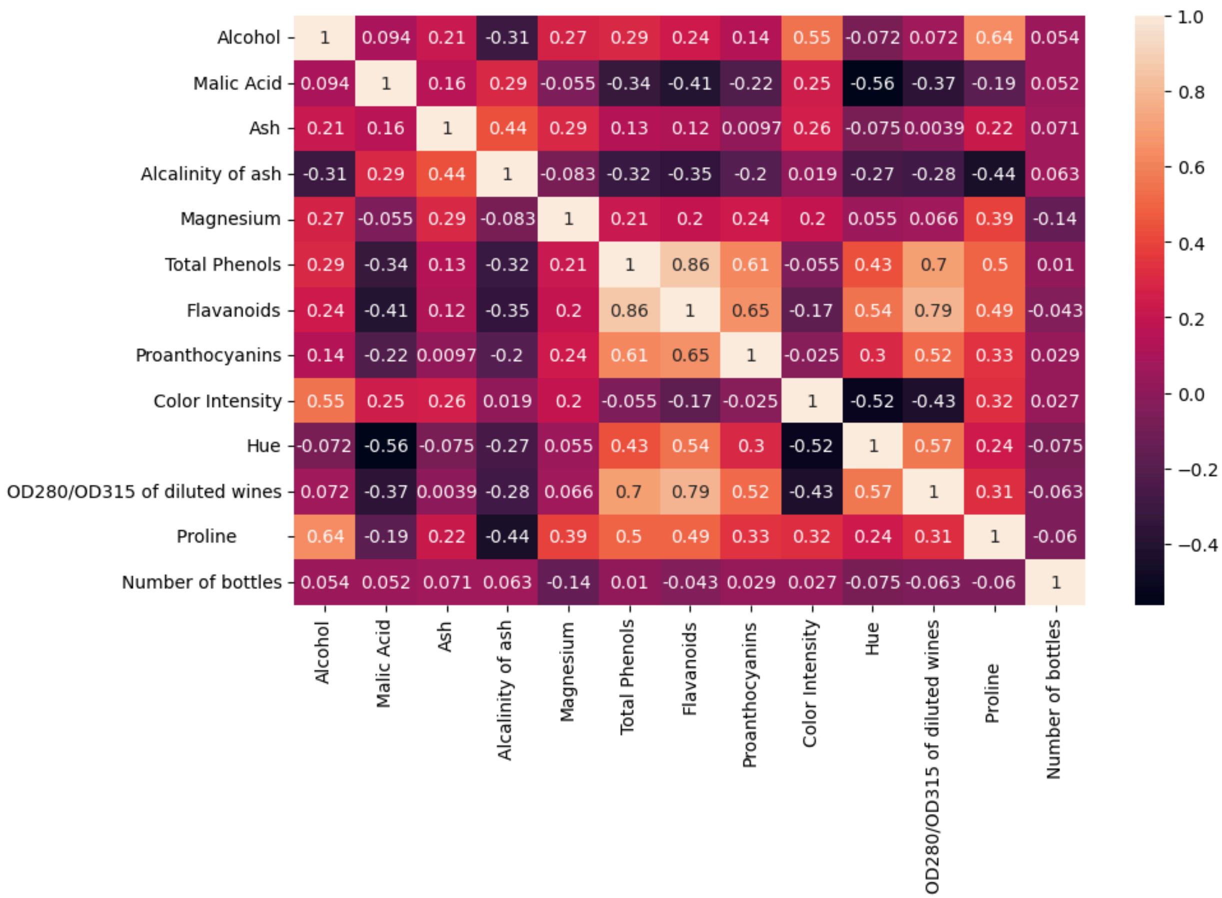
```
#Feature Selection - Removing High Correlation Values
```

```
#Correlation feature function

def corr_feature(df, corr = 0.95):
  corr_matrix = df.corr(method='pearson')
  up_tri = corr_matrix.where(np.triu(np.ones(corr_matrix.shape)).astype(bool)) # upper
  corr_table = up_tri.stack().reset_index()
  corr_table.columns = ['First_Feature', 'Second_Feature', 'Correlation']
  corr_table = corr_table[corr_table['First_Feature'] != corr_table['Second_Feature']]
  corr_high = corr_table[np.abs(corr_table['Correlation']) >= corr] # list of dropped
  #data after dropping
  feature_corr = df.drop(corr_high['First_Feature'], axis=1)
  return feature_corr, corr_high

#Applying function to high variance data
feature_corr, drop_corr = corr_feature(feature_var)
print(drop_corr)
```

```
Empty DataFrame
Columns: [First_Feature, Second_Feature, Correlation]
Index: []
```

In [21]:
```
#To prove that there are no highly correlated variables
corr_matrix = feature_var.corr(method='pearson')

#Correlation Matrix Heatmap
plt.figure(figsize = (10,6))
sns.heatmap(corr_matrix,annot=True)
```

Out[21]:
```
<Axes: >
```



	Alcohol	Malic Acid	Ash	Alcalinity of ash	Magnesium	Total Phenols	Flavanoids	Proanthocyanins	Color Intensity	Hue	OD280/OD315 of diluted wines	Proline	Number of bottles
Alcohol	1	0.094	0.21	-0.31	0.27	0.29	0.24	0.14	0.55	-0.072	0.072	0.64	0.054
Malic Acid	0.094	1	0.16	0.29	-0.055	-0.34	-0.41	-0.22	0.25	-0.56	-0.37	-0.19	0.052
Ash	0.21	0.16	1	0.44	0.29	0.13	0.12	0.0097	0.26	-0.075	0.0039	0.22	0.071
Alcalinity of ash	-0.31	0.29	0.44	1	-0.083	-0.32	-0.35	-0.2	0.019	-0.27	-0.28	-0.44	0.063
Magnesium	0.27	-0.055	0.29	-0.083	1	0.21	0.2	0.24	0.2	0.055	0.066	0.39	-0.14
Total Phenols	0.29	-0.34	0.13	-0.32	0.21	1	0.86	0.61	-0.055	0.43	0.7	0.5	0.01
Flavanoids	0.24	-0.41	0.12	-0.35	0.2	0.86	1	0.65	-0.17	0.54	0.79	0.49	-0.043
Proanthocyanins	0.14	-0.22	0.0097	-0.2	0.24	0.61	0.65	1	-0.025	0.3	0.52	0.33	0.029
Color Intensity	0.55	0.25	0.26	0.019	0.2	-0.055	-0.17	-0.025	1	-0.52	-0.43	0.32	0.027
Hue	-0.072	-0.56	-0.075	-0.27	0.055	0.43	0.54	0.3	-0.52	1	0.57	0.24	-0.075
OD280/OD315 of diluted wines	0.072	-0.37	0.0039	-0.28	0.066	0.7	0.79	0.52	-0.43	0.57	1	0.31	-0.063
Proline	0.64	-0.19	0.22	-0.44	0.39	0.5	0.49	0.33	0.32	0.24	0.31	1	-0.06
Number of bottles	0.054	0.052	0.071	0.063	-0.14	0.01	-0.043	0.029	0.027	-0.075	-0.063	-0.06	1

In [22]:
```
# Standardizing the fetaures after dropping low variance and highly correlated feature
```

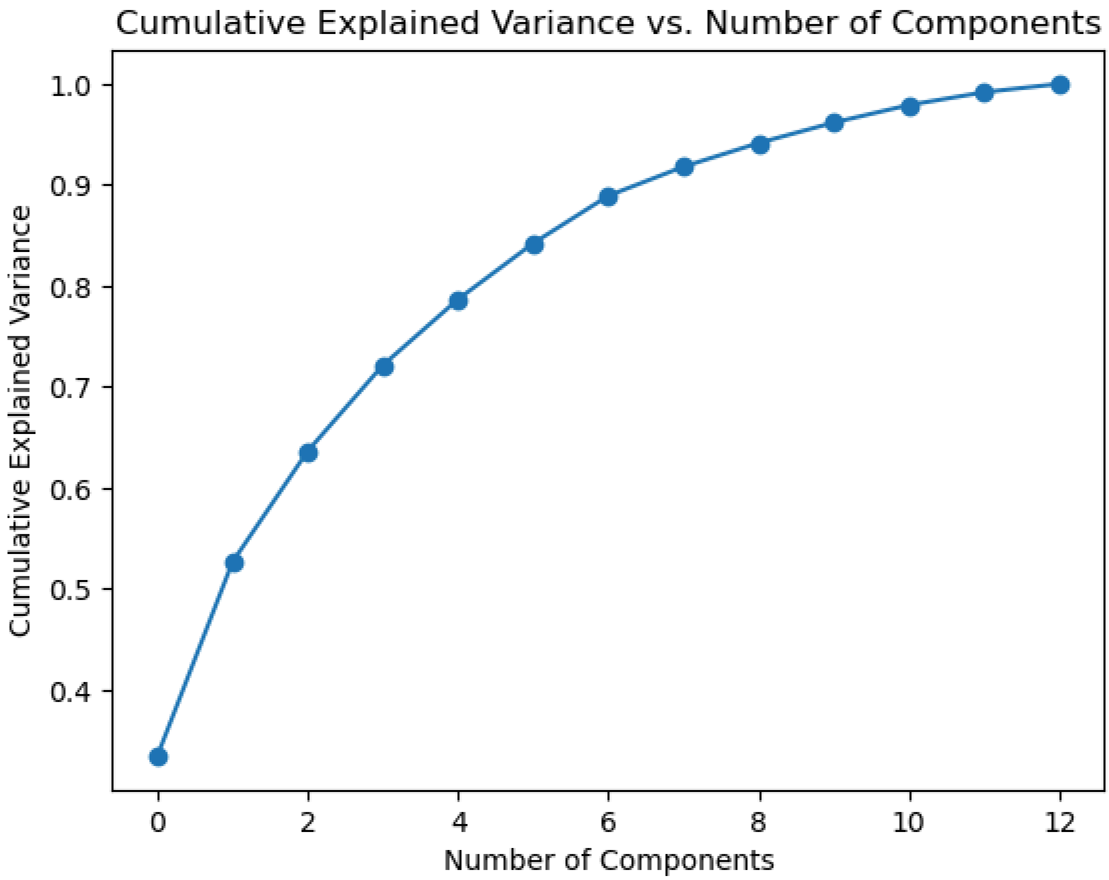
```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
feature_standardize = pd.DataFrame(scaler.fit_transform(feature_corr), columns = featu

feature_standardize.describe()

#After standardizing, the data should have a mean of 0 and variance close to 1
```

Out[22]:

	Alcohol	Malic Acid	Ash	Alcalinity of ash	Magnesium	Total Phenols	Flavanoids	Proanthocyanins	In
count	178.0000	178.0000	178.0000	178.0000	178.0000	178.0000	178.0000	178.0000	1
mean	-0.0000	-0.0000	-0.0000	-0.0000	-0.0000	0.0000	-0.0000	-0.0000	
std	1.0028	1.0028	1.0028	1.0028	1.0028	1.0028	1.0028	1.0028	
min	-2.4342	-1.4330	-3.6792	-2.6710	-2.0883	-2.1072	-1.6960	-2.0690	
25%	-0.7882	-0.6587	-0.5721	-0.6891	-0.8244	-0.8855	-0.8275	-0.5973	
50%	0.0610	-0.4231	-0.0238	0.0015	-0.1223	0.0960	0.1061	-0.0629	
75%	0.8361	0.6698	0.6981	0.6021	0.5096	0.8090	0.8491	0.6292	
max	2.2598	3.1092	3.1563	3.1545	4.3714	2.5395	3.0628	3.4851	

In [23]:

```python
#Finding the number of components

from sklearn.decomposition import PCA

pca = PCA()
PCA1=pca.fit(feature_standardize)

#Plotting the PCA
plt.plot(np.cumsum(PCA1.explained_variance_ratio_),marker="o")
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Cumulative Explained Variance vs. Number of Components')
plt.show()
```



In [24]:
```
#PRINCIPAL COMPONENT ANALYSIS

from sklearn.decomposition import PCA

# Number of components are 4 as in the above graph, after 4 components the graph is sh
pca_components = 4
pca = PCA(n_components=pca_components)
feature_pca = pd.DataFrame(pca.fit_transform(data))
feature_pca
```

Out[24]:

	0	1	2	3
0	318.9659	-47.9650	19.3819	2.9479
1	302.6681	52.9719	-3.0425	7.0750
2	437.5105	68.1693	-3.5272	-0.7938
3	732.9125	42.3584	2.0662	-0.7031
4	-12.0480	56.3262	21.0018	-0.2325
...	...	...	...	...
173	-7.1657	22.4360	-3.5440	-2.3613
174	2.7793	42.2010	4.2164	-4.0881
175	89.0022	-65.7102	15.8945	-2.4993
176	93.7004	-29.7057	17.3803	-1.8763
177	-187.1141	19.7665	0.6793	-5.5162

178 rows × 4 columns

In [25]:

```
# print individual components percentage explained variance
print("Percentage of the variance ", pca.explained_variance_ratio_[:4]*100)

# print overall percentage explained variance
print("Overall PCA explains {:.2f}% of the variance in features".format(pca.explained_
print("Raw explained variance", pca.explained_variance_)
```

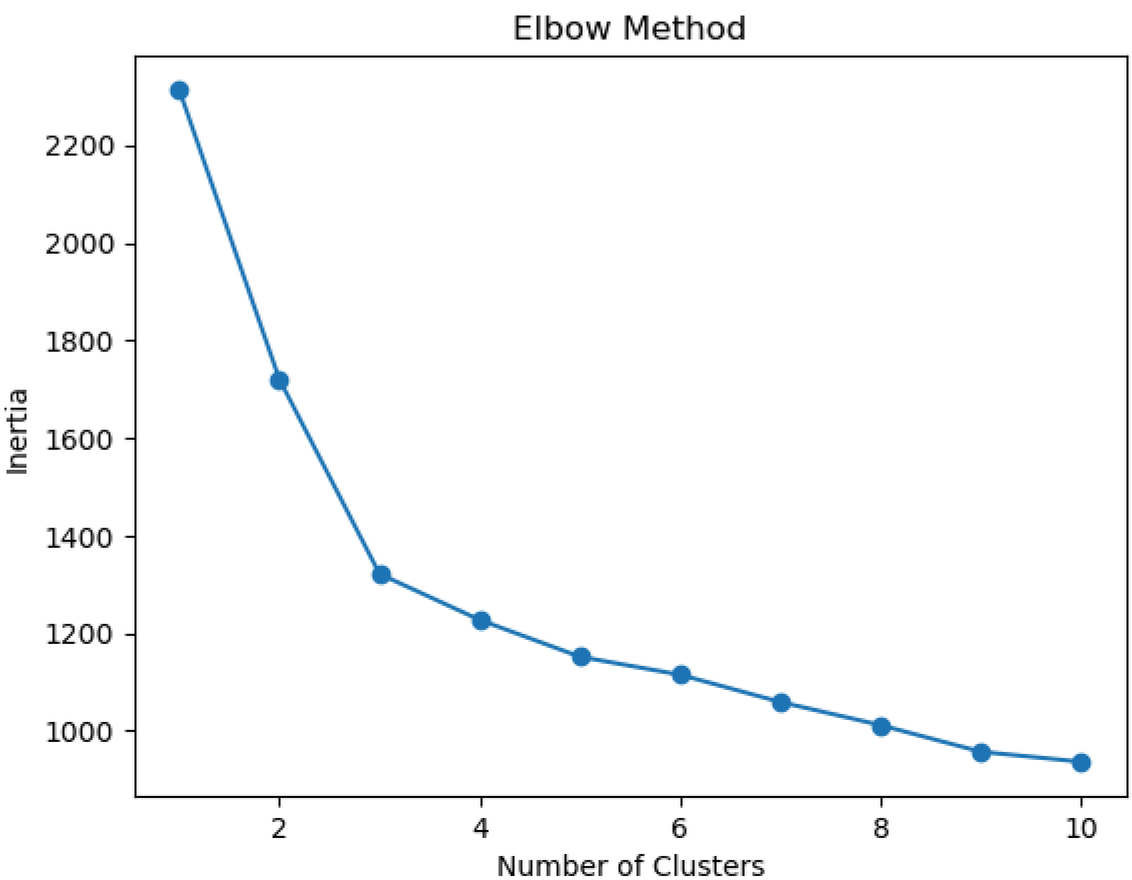
```
Percentage of the variance  [97.9878  1.8282  0.1672  0.0093]
Overall PCA explains 99.99% of the variance in features
Raw explained variance [99208.5402  1850.9453   169.2462     9.3935]
```

In [26]:

```
# Finding number of clusters
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(feature_standardize)
    inertia.append(kmeans.inertia_)

# Plot the elbow curve
plt.plot(range(1, 11), inertia, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method')
plt.show()
```

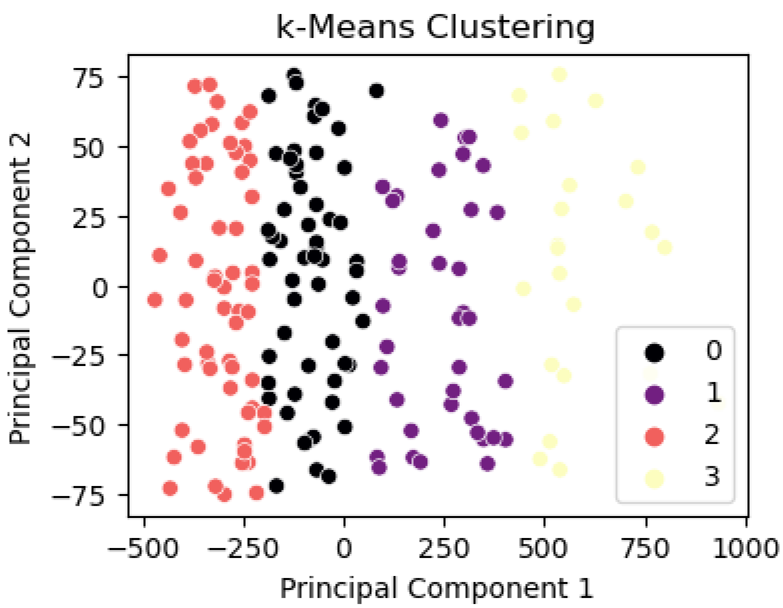
```
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
```

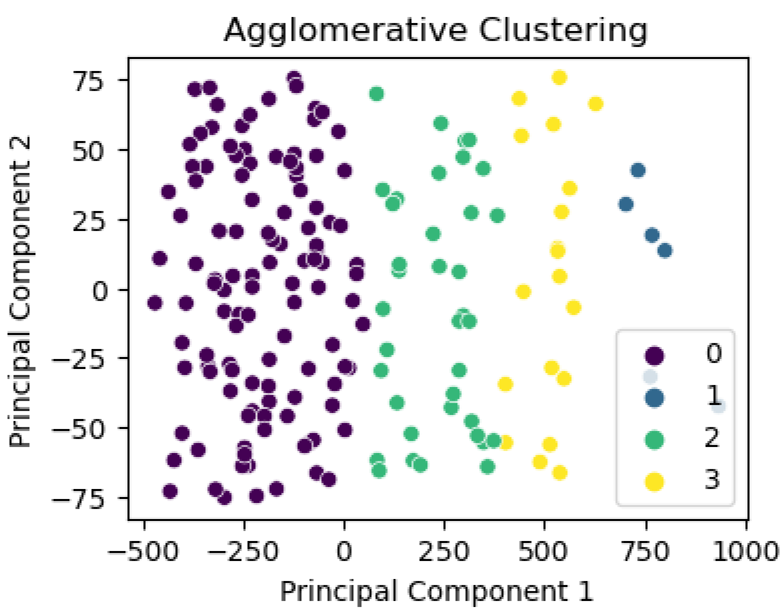
```
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
```

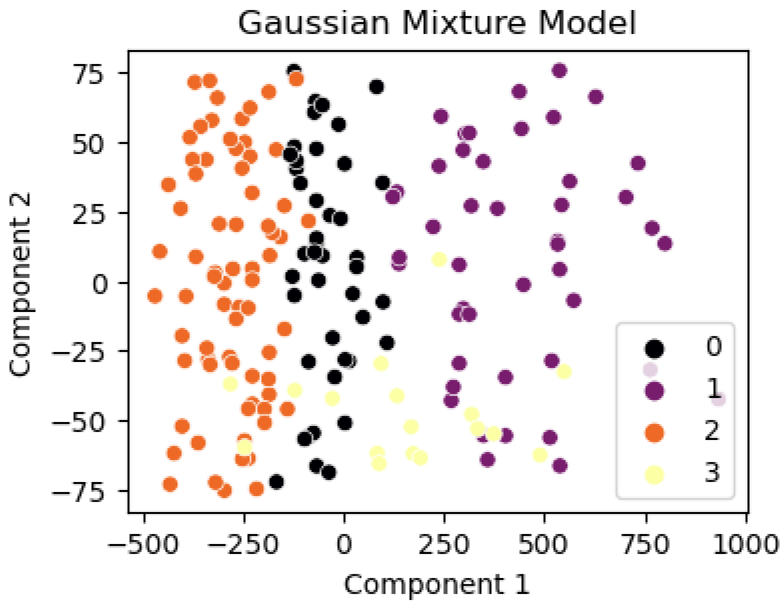


In [27]:
```
#The number of clusters is found out to be 4
from sklearn.cluster import AgglomerativeClustering, KMeans
from sklearn.mixture import GaussianMixture
import matplotlib.pyplot as plt
import seaborn as sns


# Allocating data
clustering_data = feature_pca
# Converting DataFrame to NumPy array
clustering_data_array = clustering_data.values if isinstance(clustering_data, pd.DataF

# Number of clusters
k = 4

labels_kmeans = []
labels_agg = []
labels_gauss = []

# k-Means clustering
obj_kmeans = KMeans(n_clusters=k, init='k-means++', random_state=2024)
labels_kmeans = obj_kmeans.fit_predict(clustering_data)

# Agglomerative Clustering
obj_agg = AgglomerativeClustering(n_clusters=k, linkage='average')
labels_agg = obj_agg.fit_predict(clustering_data)

# Gaussian Mixture Models
obj_gauss = GaussianMixture(n_components=k, covariance_type='full', random_state=2024)
labels_gauss = obj_gauss.fit_predict(clustering_data)

# Scatter plot for k-Means clustering
plt.figure(figsize=(4, 3))
sns.scatterplot(x=clustering_data_array[:, 0], y=clustering_data_array[:, 1], hue=labe
plt.title('k-Means Clustering')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()

# Scatter plot for Agglomerative Clustering
plt.figure(figsize=(4, 3))
sns.scatterplot(x=clustering_data_array[:, 0], y=clustering_data_array[:, 1], hue=labe
plt.title('Agglomerative Clustering')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()

# Scatter plot for Gaussian Mixture Model
plt.figure(figsize=(4,3))
sns.scatterplot(x=clustering_data_array[:, 0], y=clustering_data_array[:, 1], hue=labe
plt.title('Gaussian Mixture Model')
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.show()
```

```
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
C:\Users\91973\anaconda1\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
```







In [28]:
```
#Statistical Summary of the data
data['Number of bottles'].describe().T
```

Out[28]:
```
count    178.0000
mean      72.7022
std       43.0621
min        0.0000
25%       34.5000
50%       76.5000
75%      110.5000
max      148.0000
Name: Number of bottles, dtype: float64
```

In [29]:
```
#Negative Binomial Count Variable Regression
import statsmodels.api as sm
X = data.drop('Number of bottles',axis=1)
y = data['Number of bottles']
#As the variance of the data does exceeds the mean, we can use Negative Binomial Regre
nbr = sm.GLM(y, X, family=sm.families.NegativeBinomial()).fit()
print(nbr.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:      Number of bottles   No. Observations:                  178
Model:                            GLM   Df Residuals:                      165
Model Family:        NegativeBinomial   Df Model:                           12
Link Function:                    Log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -939.63
Date:                Tue, 07 May 2024   Deviance:                       114.08
Time:                        22:41:00   Pearson chi2:                     65.0
No. Iterations:                     8   Pseudo R-squ. (CS):            0.02841
Covariance Type:            nonrobust
=====================================================================================
===========
                                   coef    std err          z      P>|z|      [0.025
0.975]
-------------------------------------------------------------------------------------
-----------
Alcohol                          0.3655      0.083      4.390      0.000       0.202
0.529
Malic Acid                      -0.0732      0.087     -0.841      0.401      -0.244
0.097
Ash                              0.6223      0.408      1.525      0.127      -0.177
1.422
Alcalinity of ash               -0.0005      0.033     -0.016      0.988      -0.064
0.063
Magnesium                       -0.0102      0.006     -1.659      0.097      -0.022
0.002
Total Phenols                    0.2871      0.252      1.142      0.254      -0.206
0.780
Flavanoids                      -0.2083      0.199     -1.045      0.296      -0.599
0.182
Nonflavanoid phenols            -0.1387      0.805     -0.172      0.863      -1.717
1.439
Proanthocyanins                  0.2492      0.185      1.348      0.178      -0.113
0.612
Color Intensity                 -0.0722      0.056     -1.288      0.198      -0.182
0.038
Hue                             -0.2305      0.517     -0.446      0.656      -1.244
0.783
OD280/OD315 of diluted wines    -0.1500      0.206     -0.728      0.467      -0.554
0.254
Proline                         -0.0005      0.000     -1.285      0.199      -0.001
0.000
=====================================================================================
===========
```

```
C:\Users\91973\anaconda1\Lib\site-packages\statsmodels\genmod\families\family.py:136
7: ValueWarning: Negative binomial dispersion parameter alpha not set. Using default
value alpha=1.0.
  warnings.warn("Negative binomial dispersion parameter alpha not "
```