GroupX_BC1

March 2, 2022

1 Wonderful Wines of the World

1.1 Initial Setup

[1]:
```
## import libraries

import pandas as pd
import numpy as np
import scipy as sp
import seaborn as sns

import os
from math import ceil
```

[2]:
```
%matplotlib inline
import matplotlib.pyplot as plt
from matplotlib import gridspec
from matplotlib.colors import LinearSegmentedColormap
import matplotlib.cm as cm


import umap
import umap.plot
```

[3]:
```
from sklearn.base import clone

from sklearn.impute import KNNImputer
from sklearn.decomposition import PCA, FactorAnalysis

from sklearn.preprocessing import MinMaxScaler, StandardScaler, OneHotEncoder,␣
↪PowerTransformer, RobustScaler
from sklearn.preprocessing import OrdinalEncoder

from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import MeanShift, DBSCAN, estimate_bandwidth
```

```
from sklearn.neighbors import NearestNeighbors
from sklearn.mixture import GaussianMixture
from sklearn.manifold import TSNE

from sklearn.metrics import silhouette_score, silhouette_samples

from sklearn import tree
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.model_selection import train_test_split

from scipy.cluster.hierarchy import linkage, fcluster, dendrogram

import graphviz
```

[4]:
```
import warnings
warnings.filterwarnings('ignore')

# https://stackoverflow.com/questions/56618739/
 ↪matplotlib-throws-warning-message-because-of-findfont-python
import logging
logging.getLogger('matplotlib.font_manager').disabled = True

# importing sompy enables more logging
logging.disable(logging.INFO)
```

[5]:
```
import sompy
from sompy.visualization.mapview import View2D
from sompy.visualization.bmuhits import BmuHitsView
from sompy.visualization.hitmap import HitMapView
```

[6]:
```
#!pip install kneed
```

[7]:
```
from kneed import KneeLocator
```

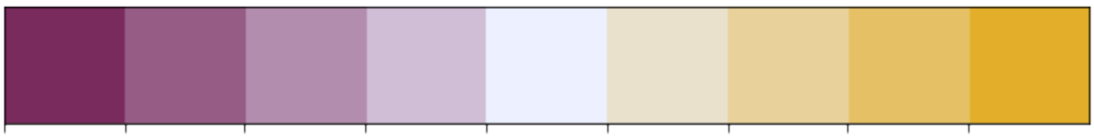
[8]:
```
## Note versions used

print('Pandas version ',pd.__version__)
print('Numpy version ',np.__version__)
print('Scipy version ',sp.__version__)
print('Seaborn version ',sns.__version__)
```

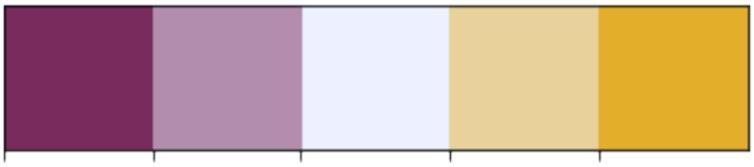
```
Pandas version  1.3.3
Numpy version  1.20.3
Scipy version  1.7.0
Seaborn version  0.11.2
```

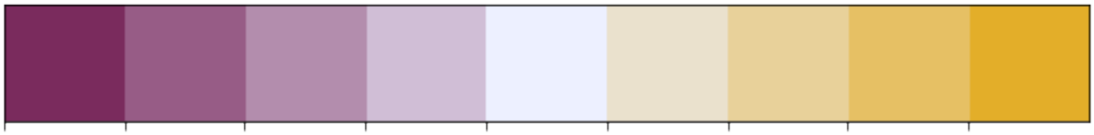
1.2 Define some constant variables

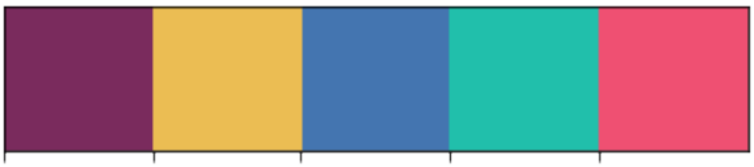
[9]:
```
COLORS = ['#7a2b5d', '#975c86', '#b38dad', '#d0bed6', '#edf0ff', '#eae1cd',␣
 ↪'#e8d19a', '#e6c064', '#e3ae29']
CAT_COLORS = ['#7a2b5d', '#EBBD53', '#4475B0', '#21BFAB', '#EF5072']
CONTRAST_COLORS = [COLORS[i] for i in range(len(COLORS)) if (i%2) == 0]

DIV_COLORS = COLORS.copy()

DEFAULT_PALETTE = sns.color_palette(COLORS)
CONTRAST_PALETTE = sns.color_palette(CONTRAST_COLORS)
DIVERGENT_PALETTE = sns.color_palette(DIV_COLORS)
CAT_PALETTE = sns.color_palette(CAT_COLORS)


DIV_CMAP = LinearSegmentedColormap.from_list("div_colors", DIV_COLORS)
CAT_CMAP = LinearSegmentedColormap.from_list("cat_colors", CAT_COLORS)


SHOW_PLOTS = True
SAVE_PLOTS = False
RANDOM_STATE = 0

show_plots = True
img_counter = 0
random_state = 0

IMG_PATH = '../../out/imgs/'
```

[10]:
```
sns.palplot(DEFAULT_PALETTE)
sns.palplot(CONTRAST_PALETTE)
sns.palplot(DIVERGENT_PALETTE)
sns.palplot(CAT_PALETTE)
plt.show()
```









[11]:
```
sns.set(style="white")

sns.set_context("paper")
sns.set_palette(DEFAULT_PALETTE)


plt.rcParams['figure.dpi'] = 70
```

[ ]:
```

```

1.3 Define some functions

[12]:
```
def save_fig(title, fig):
    if SAVE_PLOTS == True:
        fn = IMG_PATH + title.replace(' ','-') + '.png'
        fig.savefig(fn, bbox_inches='tight')
```

[ ]:
```

```

[13]:
```
## Function to plot histograms of numeric features for specified dataframe
def plot_histograms_boxplots(df, features, rows=4, title = "Histograms of␣
 ↪Numeric Variables"):
    if SHOW_PLOTS:
        cols = ceil(len(features) / rows)
        fig = plt.figure(figsize=(4*cols,4*rows),
                         constrained_layout=True)

        subfigs = fig.subfigures(rows, cols)
```

```
        for subf, feat in zip(subfigs.flatten(), features):
            axs = subf.subplots(2, 1, sharex=True, \
                                gridspec_kw={'height_ratios': [4,1]})

            axs[0].hist(df[feat], color=COLORS[0])
            axs[0].set_ylabel(None)
            axs[0].set_title(feat, y=1, fontsize=6*rows)

            axs[1].set_xlabel(None)
            flierprops = dict(markerfacecolor='None', markersize=6, ␣
↪markeredgecolor=COLORS[0])
            sns.boxplot(x=df[feat], ax=axs[1], color=COLORS[0],␣
↪flierprops=flierprops)
            axs[1].set_xlabel(None)

            subf.suptitle(None)

        plt.suptitle(title, fontsize=8*rows)
        if SAVE_PLOTS:
            save_fig(title, fig)

        plt.show()
    else:
        print("show_plots is currently set to False")
```

[ ]:

[14]:
```
def getIQR(df, colname) :
    q25 = df[colname].quantile(.25)
    q75 = df[colname].quantile(.75)
    iqr = (q75 - q25)

    upper_lim = q75 + 2 * iqr
    lower_lim = q25 - 2 * iqr

    above_ul = df.loc[df[colname]>upper_lim]
    below_ll = df.loc[df[colname]<lower_lim]

    if len(above_ul) > 0 :
        print(str(len(above_ul)) + " or " + str(round((100*len(above_ul)/
↪len(df)),4)) + "% of rows are above the UL ["+ colname + "].")
    if len(below_ll) > 0 :
        print(str(len(below_ll)) + " or " + str(round((100*len(below_ll)/
↪len(df)),4)) + "% of rows are below the LL ["+ colname + "].")
```

```
    return upper_lim, lower_lim, len(above_ul), len(below_ll)
```

1.4 Load Dataset

[15]:
```
df = pd.read_excel('../source/WonderfulWinesoftheWorld.xlsx')

## Remove last row
df.drop(df.tail(1).index,inplace=True)
df['Custid'] = df['Custid'].astype(int)
df.set_index('Custid', inplace=True)

df_original = df.copy(deep=True)

df.head(3)
```

[15]:

Custid	Dayswus	Age	Edu	Income	Freq	Recency	Monetary	LTV
5325	653.0	55.0	20.0	78473.0	20.0	18.0	826.0	445.0
3956	1041.0	75.0	18.0	105087.0	36.0	33.0	1852.0	539.0
3681	666.0	18.0	12.0	27984.0	4.0	56.0	39.0	-7.0

Custid	Perdeal	Dryred	Sweetred	Drywh	Sweetwh	Dessert	Exotic
5325	7.0	67.0	4.0	26.0	2.0	1.0	1.0
3956	2.0	49.0	0.0	46.0	1.0	3.0	0.0
3681	88.0	4.0	29.0	14.0	32.0	21.0	48.0

Custid	WebPurchase	WebVisit
5325	36.0	5.0
3956	20.0	4.0
3681	60.0	8.0

[16]:
```
if SAVE_PLOTS:
    df.describe().to_csv('../../out/data/data_summary.csv')
```

1.5 Data Understanding

1.5.1 Check datatypes

[17]:
```
df.dtypes
```

[17]:
```
Dayswus        float64
Age            float64
Edu            float64
Income         float64
Freq           float64
Recency        float64
```

```
Monetary       float64
LTV            float64
Perdeal        float64
Dryred         float64
Sweetred       float64
Drywh          float64
Sweetwh        float64
Dessert        float64
Exotic         float64
WebPurchase    float64
WebVisit       float64
dtype: object
```

All variables are numeric.

1.5.2 Check for duplicates

[18]:
```
print(df[df.duplicated(keep=False)])
```

```
Empty DataFrame
Columns: [Dayswus, Age, Edu, Income, Freq, Recency, Monetary, LTV, Perdeal,
Dryred, Sweetred, Drywh, Sweetwh, Dessert, Exotic, WebPurchase, WebVisit]
Index: []
```

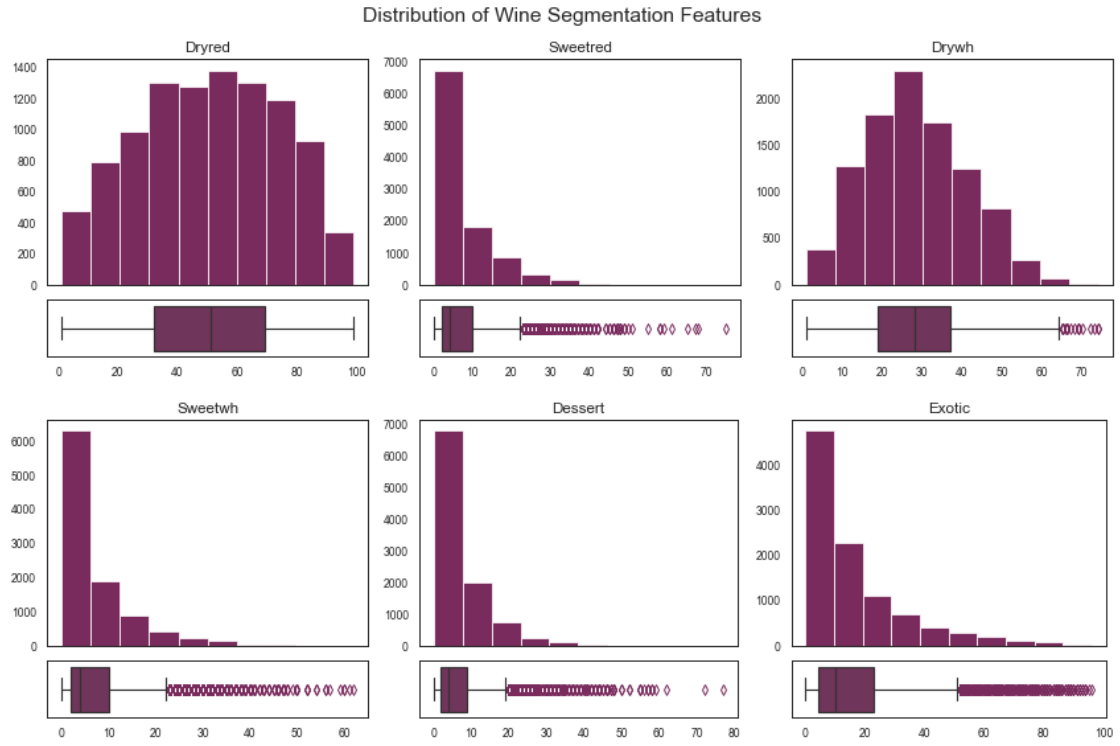
1.5.3 Identify features for segmentation

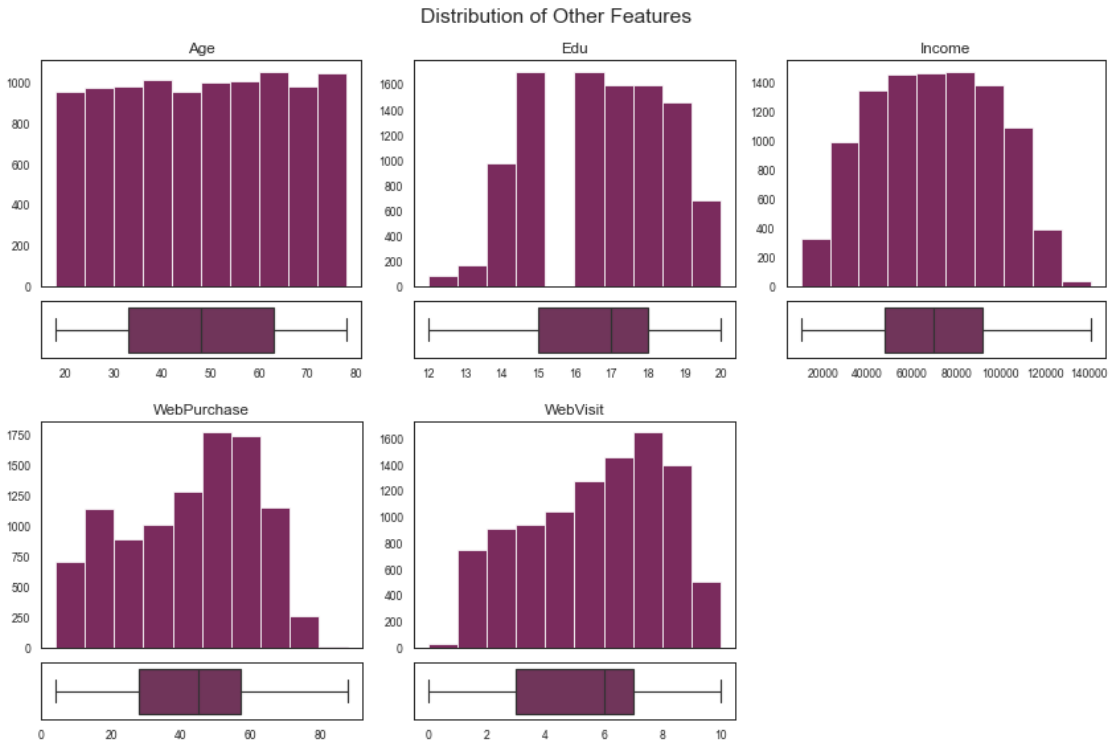
[19]:
```
## Wine preference features
wine_features = ['Dryred', 'Sweetred', 'Drywh', 'Sweetwh', 'Dessert', 'Exotic']

## Value segmentation features
value_features = ['Dayswus', 'Freq', 'Recency', 'Monetary', 'LTV', 'Perdeal']

## Separate Demographic features for cluster descriptions later
demog_features = ['Age', 'Edu', 'Income']

## Separate other features
other_features = ['WebPurchase', 'WebVisit']
```

1.5.4 Distributions of values for variables

[20]:
```
plot_histograms_boxplots(df, value_features, rows=2, title='Distribution of␣
↪Value Segmentation Features')
plot_histograms_boxplots(df, wine_features, rows=2, title='Distribution of Wine␣
↪Segmentation Features')
plot_histograms_boxplots(df, (demog_features+other_features), rows=2,␣
↪title='Distribution of Other Features')
```


Distribution of Value Segmentation Features
Dayswus
Freq
Recency
Monetary
LTV
Perdeal


Distribution of Wine Segmentation Features
Dryred
Sweetred
Drywh
Sweetwh
Dessert
Exotic



1.5.5 Explore the Recency variable

Because of it's unusual distribution

[21]:
```
fig, axes = plt.subplots(2,round(len(value_features)/2),figsize=(15,9),␣
↪sharex=True)

for f, ax in zip(range(len(value_features)),axes.flatten()):
    sns.scatterplot(data=df, x='Recency', y=value_features[f],
            alpha=.25, s=80, marker='x', palette=CAT_CMAP,
                    ax=ax
            )
plt.suptitle('Recency vs Value Features', fontsize=20)
plt.tight_layout()
plt.show()
```

[22]:
```
fig, axes = plt.subplots(2,round(len(wine_features)/2),figsize=(15,9),␣
↪sharex=True)

for f, ax in zip(range(len(wine_features)),axes.flatten()):
    sns.scatterplot(data=df, x='Recency', y=wine_features[f],
           alpha=.25, s=80, marker='x', palette=CAT_CMAP,
                    ax=ax
           )
plt.suptitle('Recency vs Wine Features', fontsize=20)
plt.tight_layout()
plt.show()
```

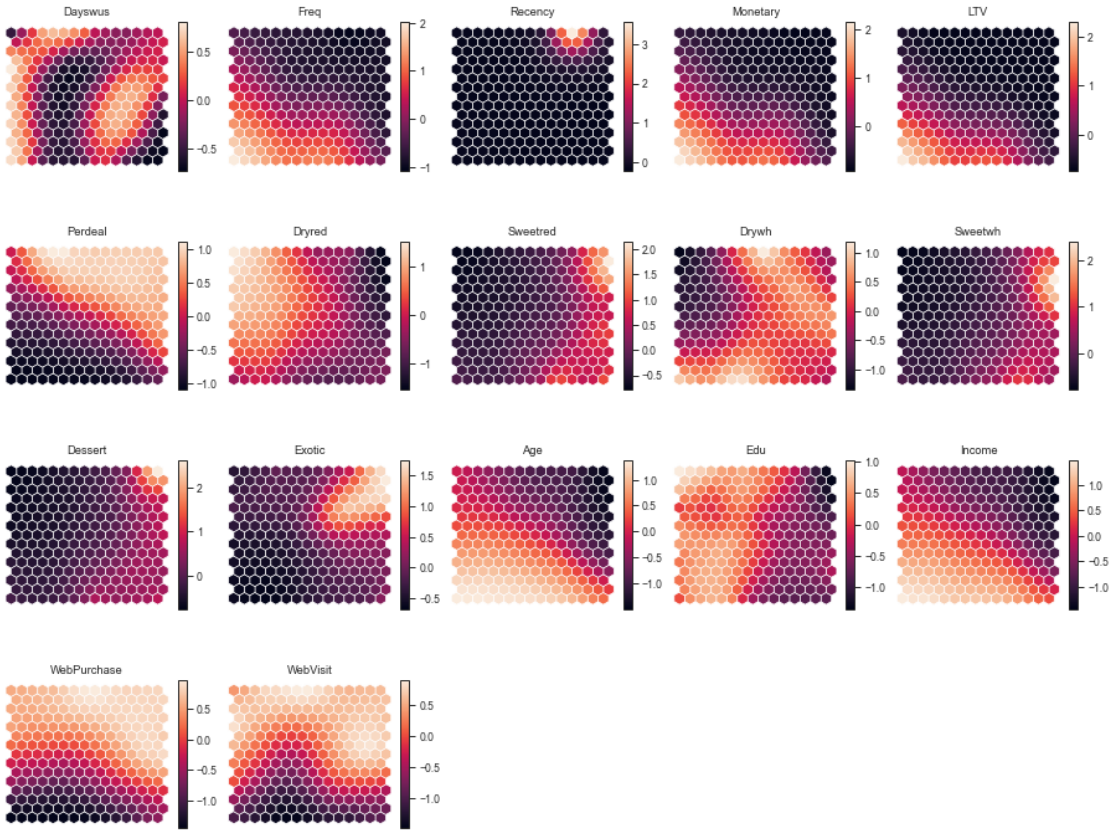
[23]:
```
do_features = demog_features+other_features
fig, axes = plt.subplots(2,round(len(do_features)/2),figsize=(15,9),␣
 ↪sharex=True)

for f, ax in zip(range(len(do_features)),axes.flatten()):
    sns.scatterplot(data=df, x='Recency', y=do_features[f],
            alpha=.25, s=80, marker='x', palette=CAT_CMAP,
                    ax=ax
            )
plt.suptitle('Recency vs Other Features', fontsize=20)
plt.tight_layout()
plt.show()
```

Consider the points at Recency > 100 as "Lost Customers" in its own cluster. Observing in particular the relationship between Recency and Frequency, it appears that customers are unlikely to repurchase if their last transaction has been more than 100 days past.

1.5.6 Visualize Component Planes

[24]:
```
som_vars = value_features + wine_features + demog_features + other_features

## From Lab 11

np.random.seed(RANDOM_STATE)

sm = sompy.SOMFactory().build(
    df[som_vars].values,
    mapsize=[15,15],
    initialization='random',
    neighborhood='gaussian',
    training='batch',
    lattice='hexa',
    component_names=som_vars
)
sm.train(n_job=4, train_rough_len=50, train_finetune_len=50)
```

[25]:
```
# Visualizing the Component planes (feature values)
view2D = View2D(2, 2, "", text_size=20)
view2D.show(sm, col_sz=5, what='codebook')
```

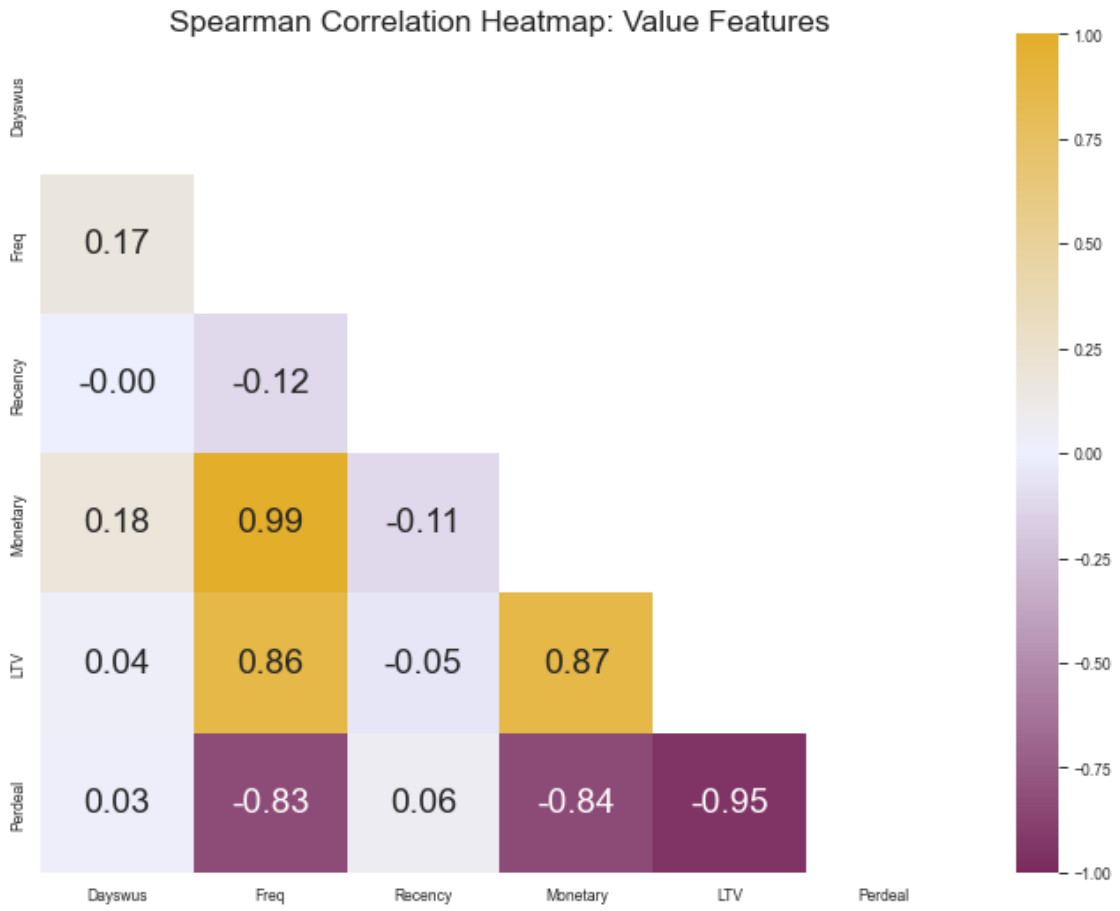


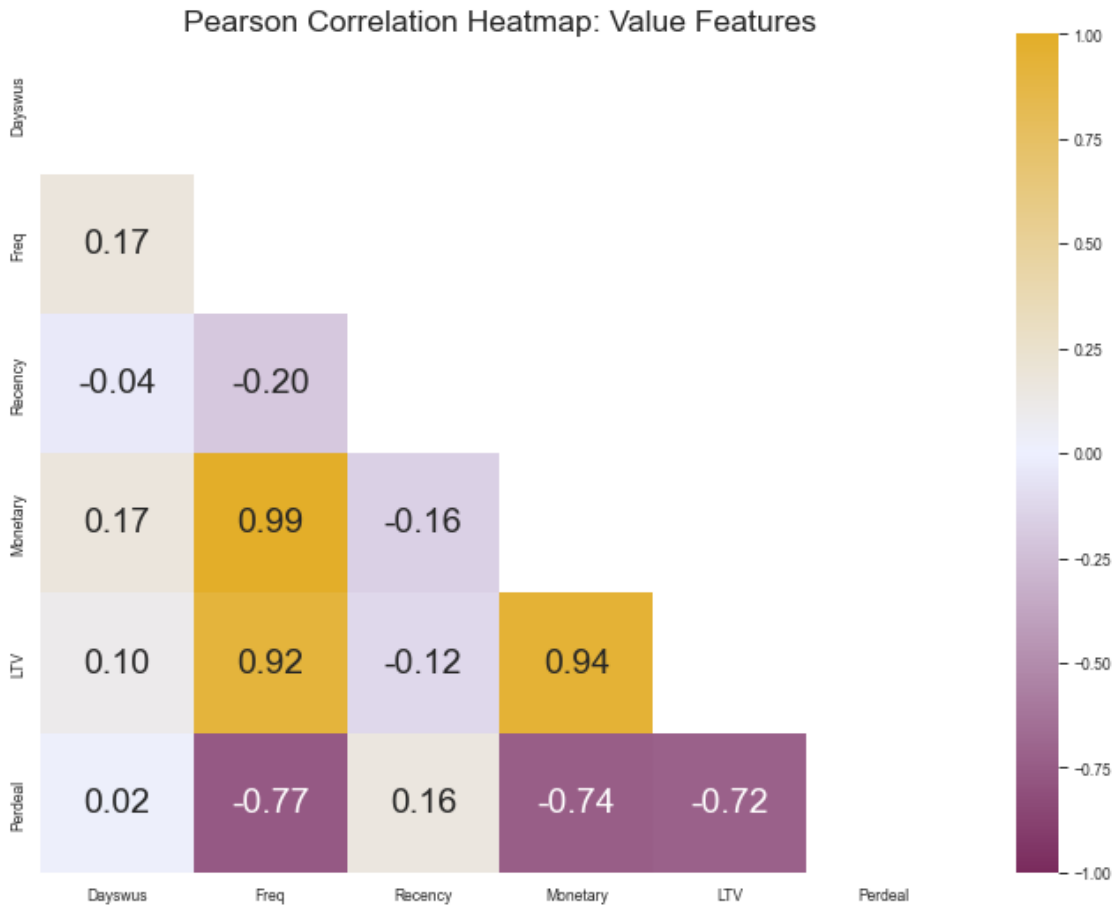
1.6 Data Preparation

1.6.1 Check for correlation

[26]:
```
def make_corr_heatmap(df, method, title="Triangle Correlation Heatmap"):
    l = len(df.columns.tolist())
    fig = plt.figure(figsize=(2*l, 1.5*l))
    mask = np.triu(np.ones_like(df.corr(method=method), dtype=bool))
    heatmap = sns.heatmap(df.corr(method=method), mask=mask, vmin=-1, vmax=1,␣
 ↪annot=True, fmt='.2f' ,cmap=DIV_CMAP)
    heatmap.set_title(title, fontdict={'fontsize':18}, pad=2);
    save_fig(title, fig)
    plt.show()
```

[27]:
```
make_corr_heatmap(df[value_features], 'spearman', title="Spearman Correlation␣
 ↪Heatmap: Value Features")
make_corr_heatmap(df[value_features], 'pearson', title="Pearson Correlation␣
 ↪Heatmap: Value Features")
```

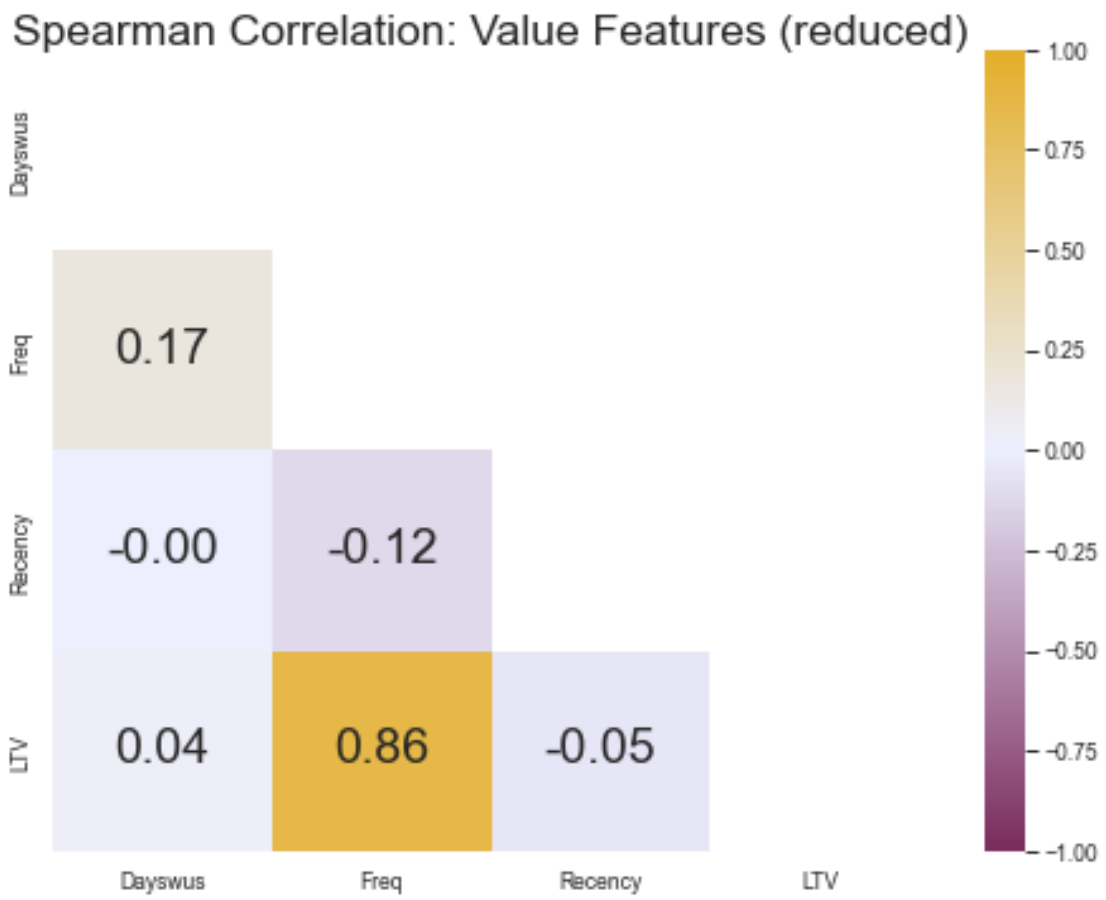

Spearman Correlation Heatmap: Value Features
Dayswus
Freq
Recency
Monetary
LTV
Perdeal
0.17
-0.00
-0.12
0.18
0.99
-0.11
0.04
0.86
-0.05
0.87
0.03
-0.83
0.06
-0.84
-0.95
Dayswus
Freq
Recency
Monetary
LTV
Perdeal
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00



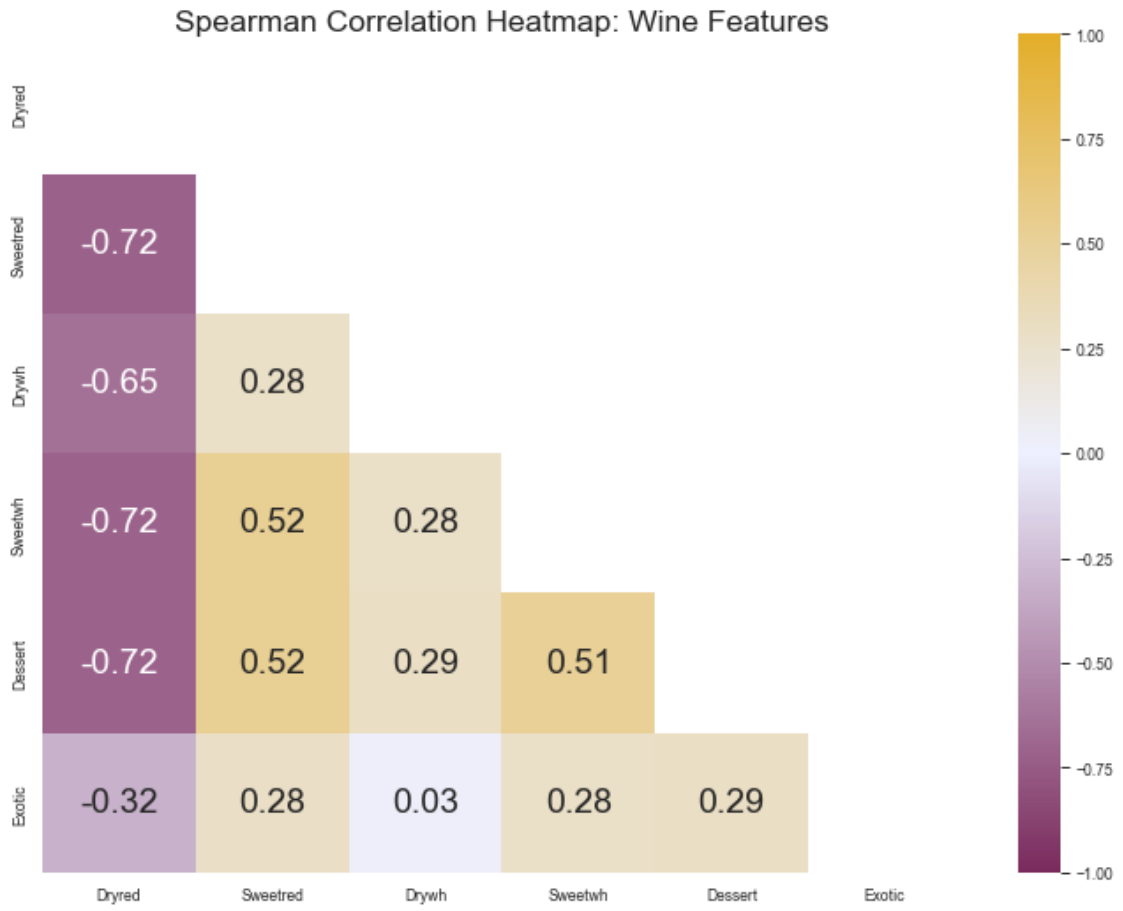
[28]:
```
## Remove Perdeal because highly correlated with multiple feats
value_features.remove('Perdeal')
other_features.append('Perdeal')

## Remove Monetary because highly correlated with Freq, LTV
value_features.remove('Monetary')
other_features.append('Monetary')


## Remove Frequency and LTV because highly correlated with Monetary
value_features2 = ['Dayswus', 'Recency', 'Monetary']

value_features
```

[28]:
```
['Dayswus', 'Freq', 'Recency', 'LTV']
```

[29]:
```
make_corr_heatmap(df[value_features], 'spearman', title="Spearman Correlation:␣
↪Value Features (reduced)")
```



[30]:
```
make_corr_heatmap(df[wine_features], 'spearman', title="Spearman Correlation␣
↪Heatmap: Wine Features")
make_corr_heatmap(df[wine_features], 'pearson', title="Pearson Correlation␣
↪Heatmap: Wine Features")
```


Spearman Correlation Heatmap: Wine Features
Dryred
Sweetred
Drywh
Sweetwh
Dessert
Exotic
-0.72
-0.65
0.28
-0.72
0.52
0.28
-0.72
0.52
0.29
0.51
-0.32
0.28
0.03
0.28
0.29
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

1.6.2 Check for outliers

[31]:
```
for v in df.columns.tolist():
    getIQR(df, v)
```

```
363 or 3.63% of rows are above the UL [Recency].
117 or 1.17% of rows are above the UL [LTV].
358 or 3.58% of rows are above the UL [Sweetred].
2 or 0.02% of rows are above the UL [Drywh].
365 or 3.65% of rows are above the UL [Sweetwh].
474 or 4.74% of rows are above the UL [Dessert].
331 or 3.31% of rows are above the UL [Exotic].
```

Too many outliers to remove just based on IQR

[ ]:
```
```

[32]:
```
df['Outlier'] = 0
```

[33]:
```
df.loc[df['Dessert']>70,['Outlier']] = 1
df.loc[df['Sweetred']>70,['Outlier']] = 1
```

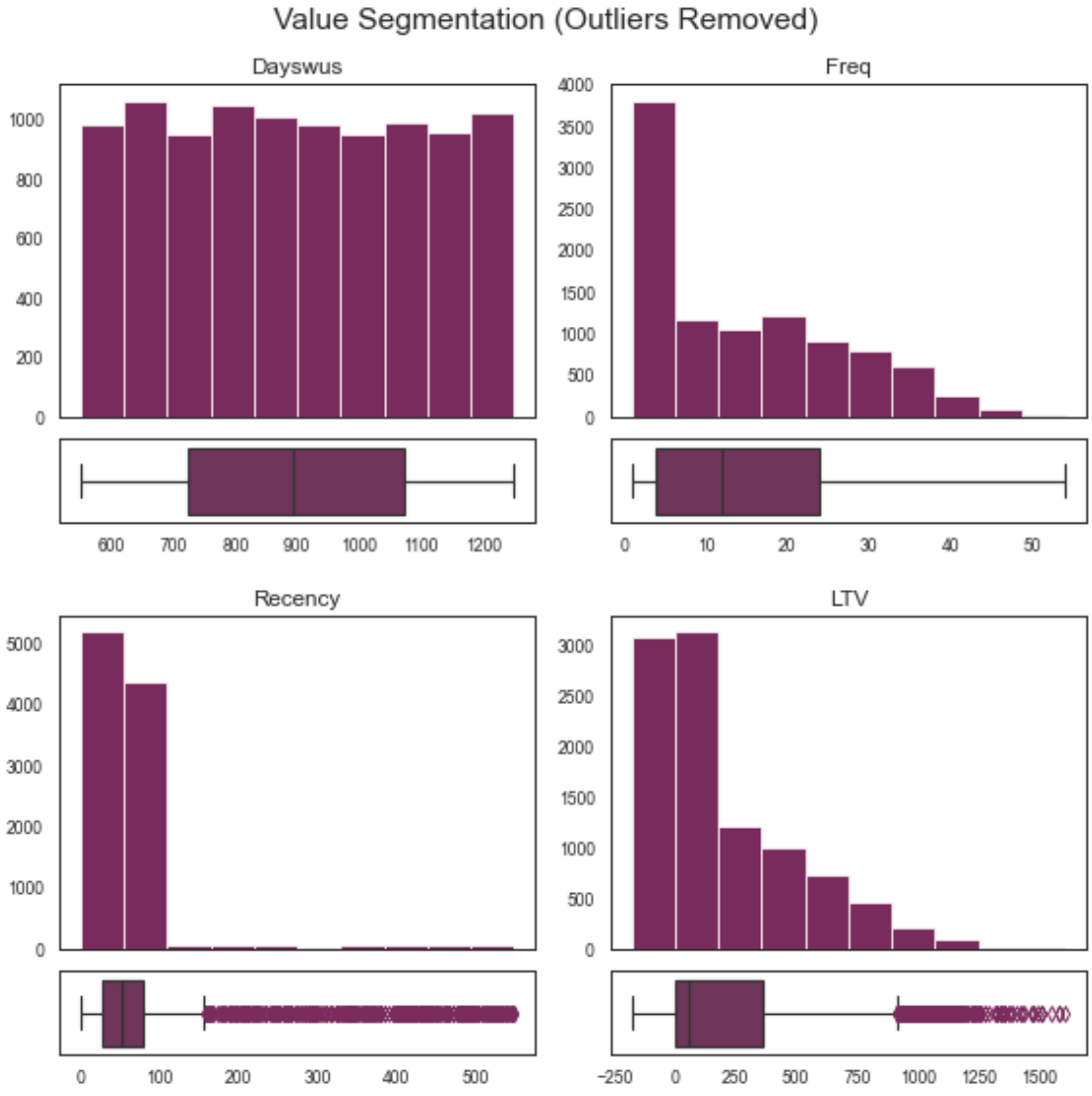
[34]:
```
df.loc[df['Freq']>55,['Outlier']] = 1
df.loc[df['LTV']>1750,['Outlier']] = 1
```

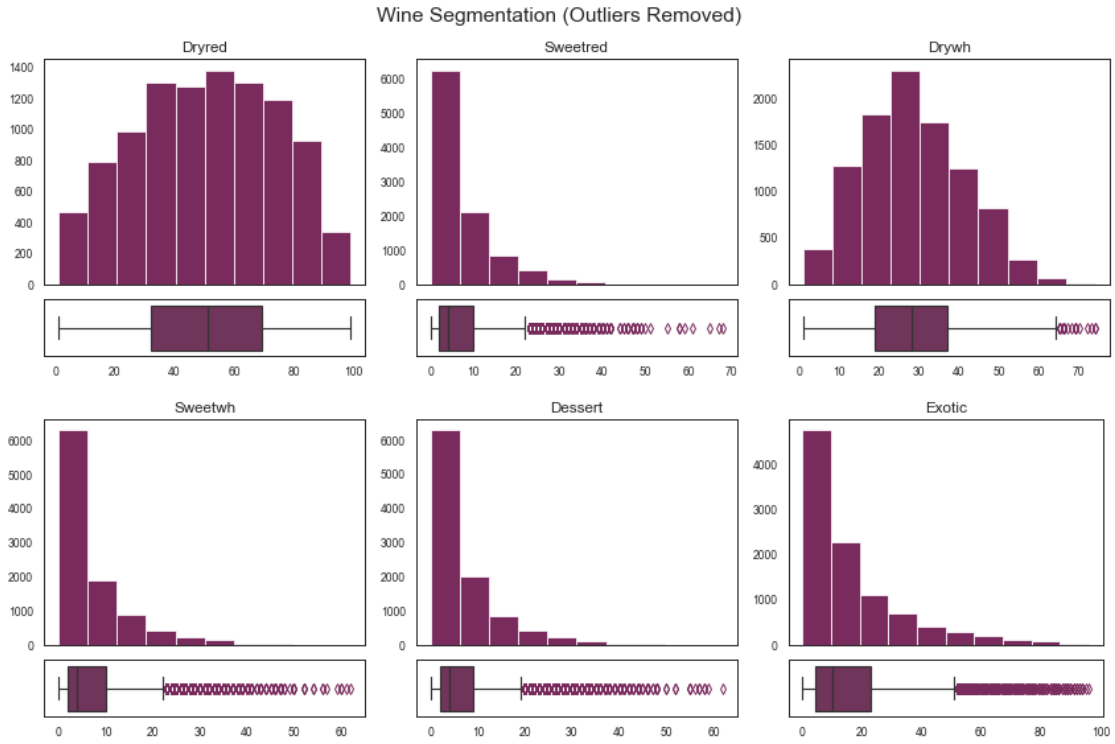
[35]:
```
df_hasoutliers = df.copy()
```

[36]:
```
df = df[df['Outlier']==0]
```

[37]:
```
df_outliers = df[df['Outlier']==1]
```

[38]:
```
plot_histograms_boxplots(df, value_features, rows=2, title='Value Segmentation␣
↪(Outliers Removed)')
plot_histograms_boxplots(df, wine_features, rows=2, title='Wine Segmentation␣
↪(Outliers Removed)')
```


Value Segmentation (Outliers Removed)
Dayswus
Freq
Recency
LTV



[ ]:

1.7 Transform Variables

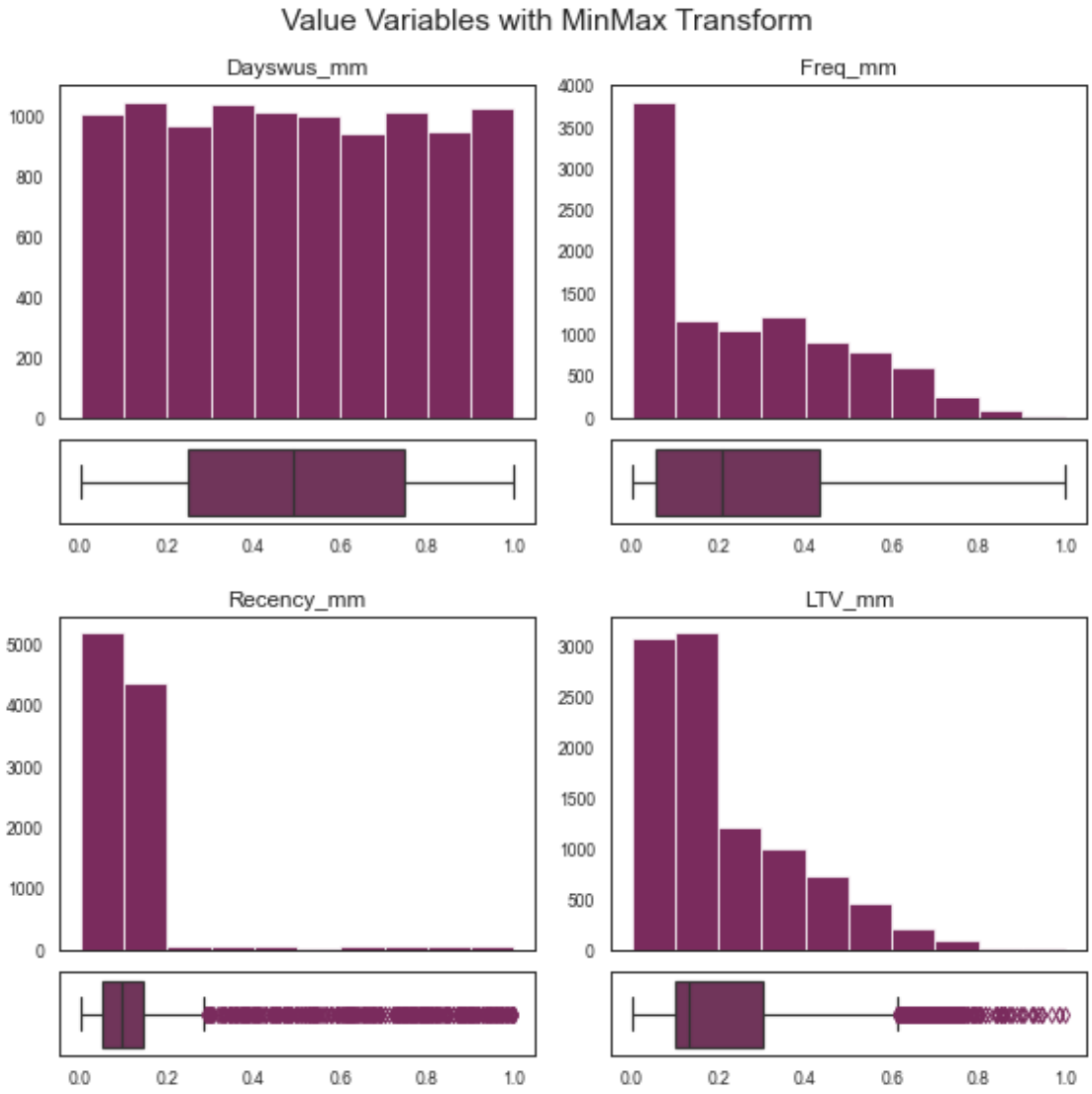
[39]:
```
##### Uncomment to separate out the 'Lost' Customers cluster
# df = df.loc[df['Recency']<=100,:]
```

1.7.1 MinMax Scaler

We only scale the Value Segmentation Features because the Wine Segmentation Features are all in the same scale (percentage) already.

[40]:
```
mmscaler = MinMaxScaler()

value_feats_mm = value_features.copy()

for fi in range(len(value_feats_mm)):
    t = value_feats_mm[fi] + '_mm'
    value_feats_mm[fi] = t
    df[t] = df[value_features[fi]]

df[value_feats_mm] = mmscaler.fit_transform(df[value_feats_mm])
```

[41]:
```
plot_histograms_boxplots(df, value_feats_mm, rows=2, title='Value Variables␣
↪with MinMax Transform')
```

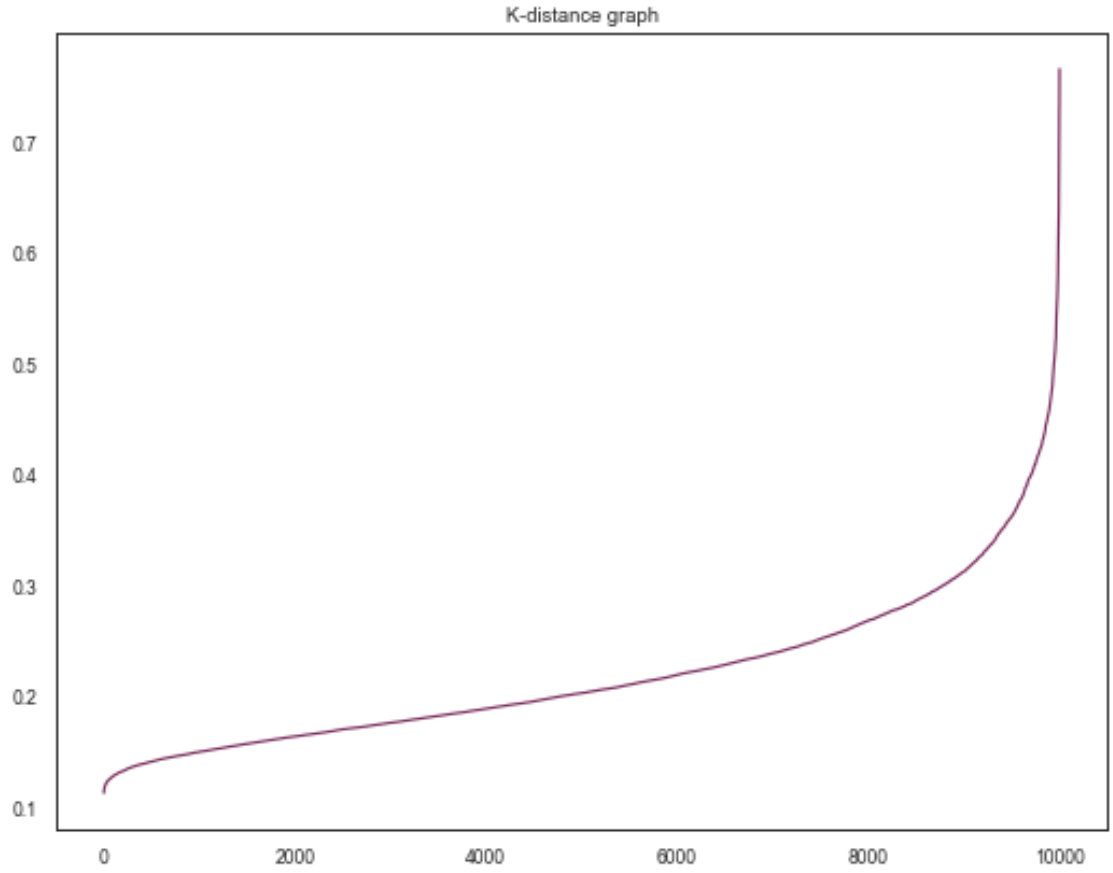


1.7.2 Transform wine features to decimal (from percentage)

[42]:
```
wine_feats_dec = wine_features.copy()

for fi in range(len(wine_feats_dec)):
    t = wine_feats_dec[fi] + '_dec'
    wine_feats_dec[fi] = t
    df[t] = df[wine_features[fi]]/100
```

1.8 Use DBSCAN to identify 'noise' rows

[43]:
```
vars_ = wine_feats_dec + value_feats_mm
neigh = NearestNeighbors(n_neighbors=50)
neigh.fit(df[vars_])
distances, _ = neigh.kneighbors(df[vars_])
distances = np.sort(distances[:, -1])

fig, axis = plt.subplots(figsize=(9,7))
plt.plot(distances, color=COLORS[0])
plt.title('K-distance graph')
plt.show()

save_fig('K-distance graph', fig)
```



[44]:
```
# Perform DBSCAN clustering
dbscan = DBSCAN(eps=.35, min_samples=20, n_jobs=4)
dbscan_labels = dbscan.fit_predict(df[vars_])
```

```
dbscan_n_clusters = len(np.unique(dbscan_labels))
print("Number of estimated clusters : %d" % dbscan_n_clusters)
```

```
Number of estimated clusters : 2
```

[45]:
```
# Concatenating the labels to df
df_dbscan = pd.concat([df[vars_], pd.Series(dbscan_labels, index=df.index,␣
↪name="dbscan_labels")], axis=1)
df_dbscan.groupby(['dbscan_labels']).count()
```

[45]:
```
                Dryred_dec  Sweetred_dec  Drywh_dec  Sweetwh_dec  Dessert_dec  \
dbscan_labels
-1                      30            30         30           30           30
 0                    9966          9966       9966         9966         9966

                Exotic_dec  Dayswus_mm  Freq_mm  Recency_mm  LTV_mm
dbscan_labels
-1                      30          30       30          30      30
 0                    9966        9966     9966        9966    9966
```

[46]:
```
df_ = pd.concat([df, pd.Series(dbscan_labels, index=df.index, name="Noise")],␣
↪axis=1)
df_noise = df_.loc[df_['Noise'] == -1]
df = df_.loc[df_['Noise'] != -1]
df_nonoise = df_.loc[df_['Noise'] != -1]
```

[47]:
```
df.drop(columns=['Noise'], inplace=True)
df
```

[47]:
```
         Dayswus   Age   Edu    Income  Freq  Recency  Monetary    LTV  \
Custid
5325       653.0  55.0  20.0   78473.0  20.0     18.0     826.0  445.0
3956      1041.0  75.0  18.0  105087.0  36.0     33.0    1852.0  539.0
3681       666.0  18.0  12.0   27984.0   4.0     56.0      39.0   -7.0
2829      1049.0  42.0  16.0   61748.0   2.0     46.0      37.0   -6.0
8788       837.0  47.0  16.0   65789.0   2.0      3.0      36.0    4.0
…            …    …     …        …      …        …         …      …
1383      1132.0  57.0  20.0   81033.0  19.0     59.0     776.0  187.0
4070       596.0  66.0  15.0   84714.0  18.0     45.0     720.0  391.0
7909       619.0  18.0  12.0   40466.0   3.0     65.0      47.0    5.0
4158      1107.0  33.0  16.0   53661.0   1.0    368.0      15.0    2.0
4914       979.0  55.0  16.0   94926.0  25.0     28.0    1148.0  293.0

         Perdeal  Dryred  …  Dayswus_mm   Freq_mm  Recency_mm    LTV_mm  \
Custid                    …
5325         7.0    67.0  …    0.147143  0.358491    0.032787  0.348824
3956         2.0    49.0  …    0.701429  0.660377    0.060109  0.401456
3681        88.0     4.0  …    0.165714  0.056604    0.102004  0.095745
```

```
2829       70.0    86.0  …    0.712857  0.018868    0.083789  0.096305
8788       35.0    85.0  …    0.410000  0.018868    0.005464  0.101904
…           …      …     …         …         …           …         …
1383       22.0    78.0  …    0.831429  0.339623    0.107468  0.204367
4070        5.0    30.0  …    0.065714  0.320755    0.081967  0.318589
7909       23.0     6.0  …    0.098571  0.037736    0.118397  0.102464
4158       35.0    18.0  …    0.795714  0.000000    0.670310  0.100784
4914        7.0    63.0  …    0.612857  0.452830    0.051002  0.263718

        Dryred_dec  Sweetred_dec  Drywh_dec  Sweetwh_dec  Dessert_dec  \
Custid
5325          0.67          0.04       0.26         0.02         0.01
3956          0.49          0.00       0.46         0.01         0.03
3681          0.04          0.29       0.14         0.32         0.21
2829          0.86          0.01       0.11         0.01         0.01
8788          0.85          0.00       0.12         0.02         0.01
…              …             …          …            …            …
1383          0.78          0.00       0.20         0.01         0.01
4070          0.30          0.12       0.36         0.10         0.12
7909          0.06          0.24       0.10         0.38         0.22
4158          0.18          0.13       0.45         0.11         0.13
4914          0.63          0.10       0.13         0.11         0.03

        Exotic_dec
Custid
5325          0.01
3956          0.00
3681          0.48
2829          0.55
8788          0.28
…              …
1383          0.11
4070          0.13
7909          0.41
4158          0.13
4914          0.04

[9966 rows x 28 columns]
```

[48]:
```
print('Percentage of rows considered noise:')
100*len(df_dbscan.loc[df_dbscan['dbscan_labels'] == -1])/len(df)
```

```
Percentage of rows considered noise:
```

[48]:
```
0.30102347983142685
```

[49]:
```
len(df_noise.loc[df_noise['Recency']>100,:])/len(df_noise)
```

[49]: 0.9

1.9 Summary of Variables

[50]:
```
## Original value features
print(value_features)
df[value_features].head(3)
```

```
['Dayswus', 'Freq', 'Recency', 'LTV']
```

[50]:

Custid	Dayswus	Freq	Recency	LTV
5325	653.0	20.0	18.0	445.0
3956	1041.0	36.0	33.0	539.0
3681	666.0	4.0	56.0	-7.0

[51]:
```
## Original wine features
print(wine_features)
df[wine_features].head(3)
```

```
['Dryred', 'Sweetred', 'Drywh', 'Sweetwh', 'Dessert', 'Exotic']
```

[51]:

Custid	Dryred	Sweetred	Drywh	Sweetwh	Dessert	Exotic
5325	67.0	4.0	26.0	2.0	1.0	1.0
3956	49.0	0.0	46.0	1.0	3.0	0.0
3681	4.0	29.0	14.0	32.0	21.0	48.0

[52]:
```
## MinMax transformed value features
print(value_feats_mm)
df[value_feats_mm].head(3)
```

```
['Dayswus_mm', 'Freq_mm', 'Recency_mm', 'LTV_mm']
```

[52]:

Custid	Dayswus_mm	Freq_mm	Recency_mm	LTV_mm
5325	0.147143	0.358491	0.032787	0.348824
3956	0.701429	0.660377	0.060109	0.401456
3681	0.165714	0.056604	0.102004	0.095745

[53]:
```
## Decimal wine features
print(wine_feats_dec)
df[wine_feats_dec].head(3)
```

```
['Dryred_dec', 'Sweetred_dec', 'Drywh_dec', 'Sweetwh_dec', 'Dessert_dec',
'Exotic_dec']
```

[53]:

Custid	Dryred_dec	Sweetred_dec	Drywh_dec	Sweetwh_dec	Dessert_dec	\
5325	0.67	0.04	0.26	0.02	0.01	

```
3956          0.49          0.00        0.46          0.01          0.03
3681          0.04          0.29        0.14          0.32          0.21

         Exotic_dec
Custid
5325          0.01
3956          0.00
3681          0.48
```

2 Clustering

2.1 RFM Analysis

[54]:
```
bins = 5
rfm_df = df.loc[df['Recency']<=100,:]
rfm_df['Recency_Rank'] = bins - pd.qcut(rfm_df['Recency'], bins, labels=False)
rfm_df['Frequency_Rank'] = 1 + pd.qcut(rfm_df['Freq'], bins, labels=False)
rfm_df['Monetary_Rank'] = 1+ pd.qcut(rfm_df['Monetary'], bins, labels=False)
rfm_df['RFM_Ave'] = rfm_df.loc[:
 ↪,['Recency_Rank','Frequency_Rank','Monetary_Rank']].mean(axis=1)
```

[55]:
```
bins = 5
df['Recency_Rank'] = 0
df['Frequency_Rank'] = 0
df['Monetary_Rank'] = 0

df.loc[df['Recency']<=100,'Recency_Rank'] = bins - \
    pd.qcut(df.loc[df['Recency']<=100,'Recency'], bins, labels=False)



df.loc[df['Recency']<=100,'Frequency_Rank'] = 1 + pd.qcut(df.
 ↪loc[df['Recency']<=100,'Freq'], bins, labels=False)
df.loc[df['Recency']<=100,'Monetary_Rank'] = 1+ pd.qcut(df.
 ↪loc[df['Recency']<=100,'Monetary'], bins, labels=False)

#df.loc[df['Recency']<=100,'RFM_Ave'] = df.
 ↪loc[df['Recency']<=100,['Recency_Rank','Frequency_Rank','Monetary_Rank']].
 ↪mean(axis=1)

df['RFM_Ave'] = df.loc[:,['Recency_Rank','Frequency_Rank','Monetary_Rank']].
 ↪mean(axis=1)
```

[56]:
```
df.loc[:
 ↪,['Recency','Recency_Rank','Freq','Frequency_Rank','Monetary_Rank','RFM_Ave']]
```

[56]:

Custid	Recency	Recency_Rank	Freq	Frequency_Rank	Monetary_Rank	RFM_Ave
5325	18.0	5	20.0	4	4	4.333333
3956	33.0	4	36.0	5	5	4.666667
3681	56.0	3	4.0	1	1	1.666667
2829	46.0	3	2.0	1	1	1.666667
8788	3.0	5	2.0	1	1	2.333333
…	…	…	…	…	…	…
1383	59.0	3	19.0	4	4	3.666667
4070	45.0	3	18.0	4	4	3.666667
7909	65.0	2	3.0	1	1	1.333333
4158	368.0	0	1.0	0	0	0.000000
4914	28.0	4	25.0	4	4	4.000000

[9966 rows x 6 columns]

[ ]:

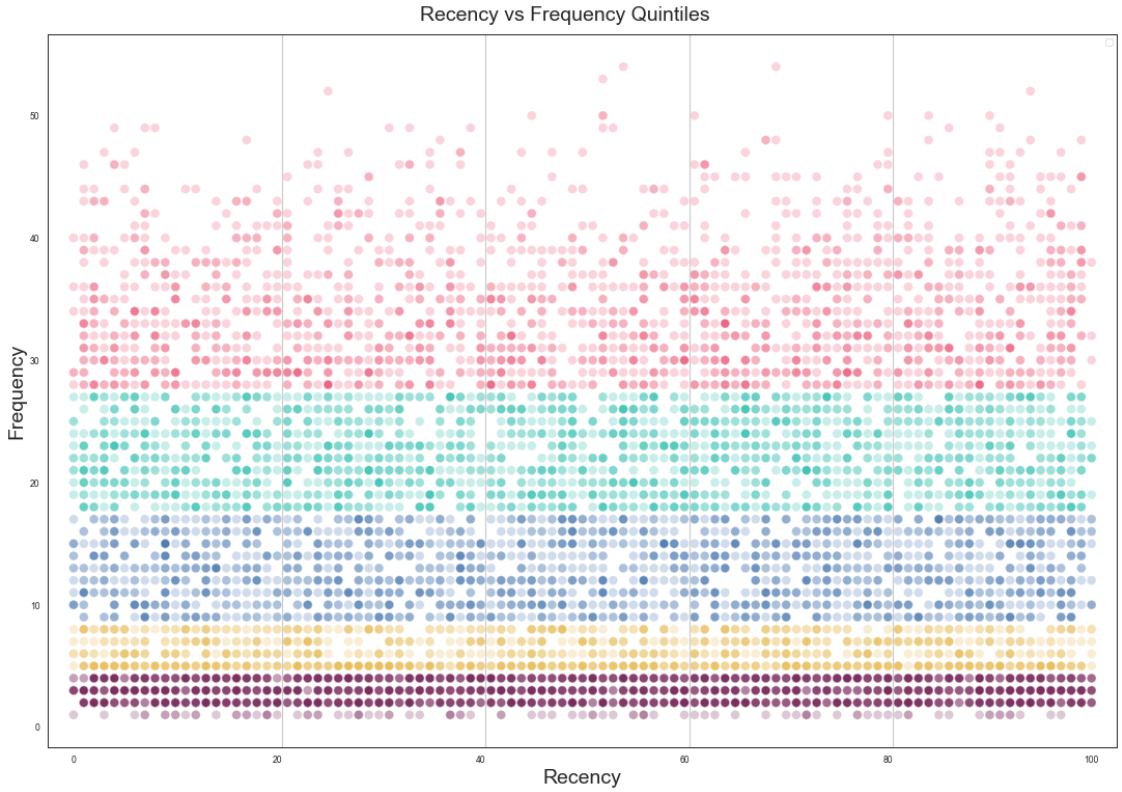
[ ]:

[57]:
```
rfm_rank_features = ['Frequency_Rank', 'Recency_Rank', 'Monetary_Rank',␣
↪'RFM_Ave']
rfm_features = ['Freq', 'Recency', 'Monetary',]
```

[58]:
```
quantile_values = [rfm_df['Recency'].quantile(i) for i in [.2,.4,.6,.8]]
freq_quints = [rfm_df['Freq'].quantile(i) for i in [.2,.4,.6,.8]]

#quantile_values
#freq_quints
```

[59]:
```
fig, ax = plt.subplots(figsize=(19,13))

for q in range(len(quantile_values)):
    ax.axvline(quantile_values[q]+.49,
               label=str(int(quantile_values[q]))+'th Percentile',
               color='#CCCCCC')

    #ax.axhline(freq_quints[q]+.5,
    #           label=str(int(quantile_values[q]))+'th Percentile',
    #           color='#CCCCCC')

scatter = sns.scatterplot(data=rfm_df, x='Recency', y='Freq',␣
↪hue='Frequency_Rank',
          alpha=.25, s=80, marker='o', palette=CAT_CMAP,
                        #style='Recency_Rank'
                       )
```

```
ax.set_ylabel('Frequency', fontsize=20)
ax.set_xlabel('Recency', fontsize=20)


ax.legend([])
rfm_title='Recency vs Frequency Quintiles'

plt.suptitle(rfm_title, fontsize=20, y=.91)
plt.margins(0.025, 0.05)
if SAVE_PLOTS:
    save_fig(rfm_title, fig)
plt.show()
```



[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

2.2 Some helpful functions

[60]:
```
## Code from Lab #13

def get_ss(df):
    """Computes the sum of squares for all variables given a dataset
    """
    ss = np.sum(df.var() * (df.count() - 1))
    return ss  # return sum of sum of squares of each df variable

def r2_(df, labels):
    sst = get_ss(df)
    ssw = np.sum(df.groupby(labels).apply(get_ss))
    return 1 - ssw/sst

def r2(dft, df, labels, feats):
    sst = get_ss(dft[feats])
    ssw = np.sum(df.groupby(labels).apply(get_ss))
    return 1 - ssw/sst

def get_r2_scores(df, clusterer, min_k=2, max_k=10):
    """
    Loop over different values of k. To be used with sklearn clusterers.
    """
    r2_clust = {}
    for n in range(min_k, max_k):
        clust = clone(clusterer).set_params(n_clusters=n)
        labels = clust.fit_predict(df)
        r2_clust[n] = r2_(df, labels)
    return r2_clust


def showR2Plot(r2_scores, title=""):
    # Visualizing the R² scores for each cluster solution on demographic␣
 ↪variables
    fig, ax = plt.subplots(figsize=(10,7))
    pd.DataFrame(r2_scores).plot(ax=ax)
    plt.title("R² Plot: " + title)
    ax.set_xlabel("Number of clusters")
    ax.set_ylabel("R² metric")
    save_fig('R2_Plot_'+ title, fig)
    plt.show()
```

[61]:
```
def show_cluster_heatmap(df, label1, cols, title='Heatmap of Cluster Means',␣
 ↪cluster_labels=[]):
```

```
    fig, ax = plt.subplots(constrained_layout=True , figsize=(len(cols)*1.
↪3,len(cols)*.7), sharey=True)

    label_counts = df.groupby(label1)[label1].count()
    df_i = df.groupby(label1)[cols].mean().index


    if len(cluster_labels) == 0:
        cluster_labels = df_i

    xticks = [(str(cluster_labels[l]) + "\n" + str(label_counts[l])) for l in␣
↪range(len(label_counts))]

    sns.heatmap(df.groupby(label1)[cols].mean().loc[df_i,:].T, \
            vmin=0, vmax=1,
            cmap=DIV_CMAP, xticklabels=xticks, annot=True, fmt='.2f',␣
↪annot_kws={"fontsize":14})

    k = len(df[label1].unique().tolist())

    ax.set_xlabel(('Cluster Sizes and Labels, K = '+str(k)))
    fig.suptitle(title, y=1.1, fontsize=20)

    if SAVE_PLOTS:
        save_fig(title, fig)

    plt.show()
```

[ ]:

[62]:
```
def compare_cluster_heatmap(df, k1, k2, label1, label2, cols, title='Heatmap of␣
↪Cluster Means'):
    fig, ax = plt.subplots(1,2,gridspec_kw={'width_ratios': [k1, k2]} ,\
                           constrained_layout=True , figsize=(14,len(cols)*.8),␣
↪sharey=True)

    label_counts = df.groupby(label1)[label1].count()

    xticks = [(str(l) + "\n" + str(label_counts[l])) for l in␣
↪range(len(label_counts))]
    sns.heatmap(df.groupby(label1)[cols].mean().loc[range(0,k1),:].T, \
            vmin=0, vmax=1,\
            cmap=DIV_CMAP, ax=ax[0], xticklabels=xticks, annot=True, fmt='.2f',␣
↪annot_kws={"fontsize":16})

    label_counts = df.groupby(label2)[label2].count()
```

```
    xticks = [(str(l) + "\n" + str(label_counts[l])) for l in␣
↪range(len(label_counts))]
    sns.heatmap(df.groupby(label2)[cols].mean().loc[range(0,k2),:].T, \
            vmin=0, vmax=1,\
            cmap=DIV_CMAP, ax=ax[1], xticklabels=xticks, annot=True, fmt='.2f',␣
↪annot_kws={"fontsize":16})

    ax[0].set_xlabel('Cluster Sizes and Labels, K='+str(k1))
    ax[1].set_xlabel('Cluster Sizes and Labels, K='+str(k2))
    fig.suptitle(title, y=1.1, fontsize=20)
    save_fig(title, fig)

    plt.show()
```

[63]:
```
def show_elbow_silhouette(df, features, max_k=40):
    n_clusters = range(2,max_k)
    silhouette_scores = []
    sum_squared_dist = []
    r_scores = []

    for num_clusters in n_clusters:
        kmeans = KMeans(n_clusters=num_clusters, init='k-means++',␣
↪max_iter=1500,n_init=100,random_state=RANDOM_STATE)

        kmeans.fit(df[features])
        cluster_labels = kmeans.labels_
        sum_squared_dist.append(kmeans.inertia_)
        silhouette_scores.append(silhouette_score(df[features], cluster_labels,\
                                                  metric='euclidean', \
                                                  sample_size=None,␣
↪random_state=None))
        df['labels'] = cluster_labels
        r_scores.append(r2_(df, 'labels'))

    fig, ax1 = plt.subplots(figsize=(11,7))
    ax2 = ax1.twinx()

    ax1.plot(n_clusters,sum_squared_dist, color=CAT_COLORS[0], label='Inertia')

    #ax2.plot(n_clusters, r_scores, color=CAT_COLORS[2], label='R2' )

    ax1.set_xlabel('Values of K')
    ax1.set_ylabel('Inertia', color=COLORS[0])

    #ax2.set_ylabel('R2 Score', color=CAT_COLORS[2])
    #ax2.yaxis.set_label_coords(.97, 0.5)
```

```
    kl = KneeLocator(range(1, len(sum_squared_dist)+1), sum_squared_dist,␣
↪curve="convex", direction="decreasing")
    ax1.axvline(x=kl.elbow, label='Elbow', color=CAT_COLORS[0],␣
↪linestyle='dashed')

    fig.legend(loc="center right", bbox_to_anchor=(.85, .5))
    plt.xticks(range(1,max_k+1,2))
    plt.title('Inertia Plot')
    save_fig('Inertia R2 Plot', fig)



    plt.show()

    print('Knee located at k=', kl.elbow)
```

[64]:
```
def plot_clusters(df, labels, title='Cluster Visualization', label_map=None):
    l = labels
    if label_map:
        l2 = labels.copy()
        lmap = {}
        for i in range(len(label_map)):
            lmap[label_map[i]] = i
        l = l2.replace(lmap)

    fig, ax = plt.subplots(figsize=(10,10))

    scatter = ax.scatter(x=df[0], y=df[1], c=l, cmap=CAT_CMAP,\
               s=2, marker="o", alpha=.75, label='Final Clusters')

    ncol = len(labels.unique().tolist())
    ncol = round(ncol/2) + 1 if ncol > 7 else ncol
    if label_map == None:
        legend1 = ax.legend(*scatter.legend_elements(),
                        bbox_to_anchor=(.5,1,.5,1), loc="lower left",␣
↪frameon=False,
                        mode='expand', borderaxespad=0, ncol=ncol,
                       )

    ax.set_xticklabels('')
    ax.set_yticklabels('')
    plt.title(title, loc='left')

    if SAVE_PLOTS:
        save_fig(title, fig)
```

```
    plt.show()
```

[65]:
```
## Code based from Lab 09
def plot_dendrogram(model, title, **kwargs):
    sns.set_palette(CAT_PALETTE)

    # Create linkage matrix and then plot the dendrogram

    fig, ax = plt.subplots(figsize=(11,5))

    # create the counts of samples under each node
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack(
        [model.children_, model.distances_, counts]
    ).astype(float)

    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs)

    plt.hlines(kwargs['color_threshold'], 0, 1000, colors="r",␣
↪linestyles="dashed")


    plt.xticks(rotation=90)
    plt.title(title)

    plt.xlabel("Number of points in node (or index of point if no parenthesis).
↪")
    save_fig(title, fig)
    plt.show()

    sns.set_palette(DEFAULT_PALETTE)
```

[66]:
```
## Scoring
r_scores_wine = pd.DataFrame(columns=['method','r2','clusters'])
```

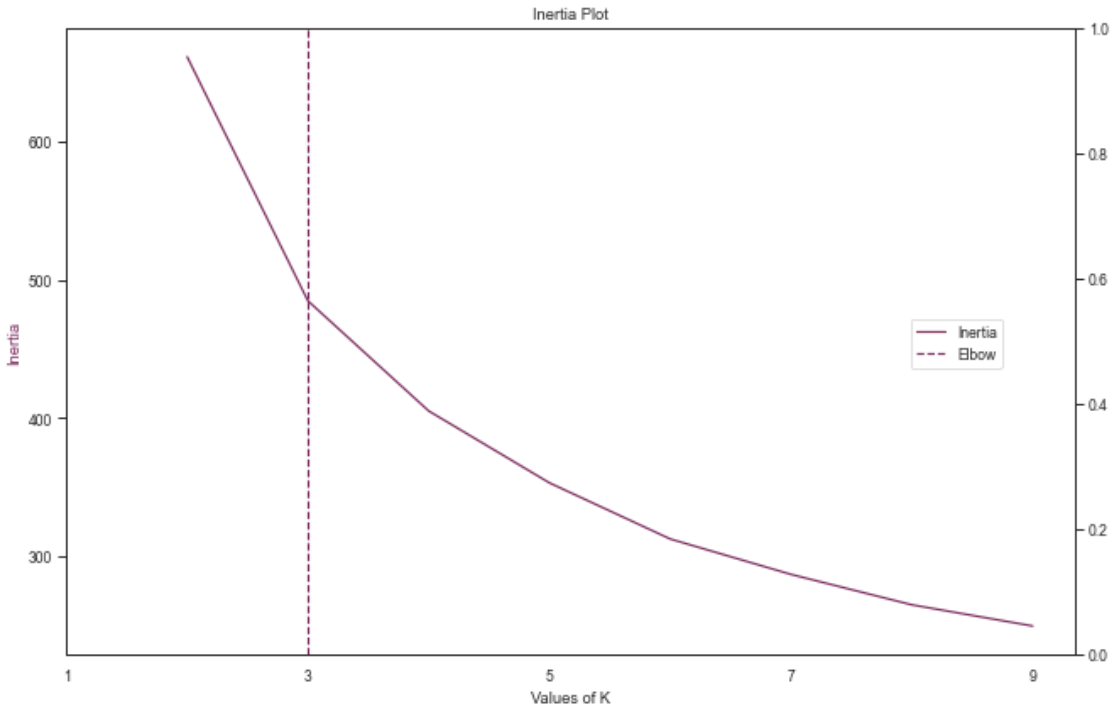
```
r_scores_value = pd.DataFrame(columns=['method','r2','clusters'])
```

2.3 Wine Segmentation

[67]:
```
df_wine_kmeans = df_nonoise[wine_feats_dec].copy()
```

[ ]:
```

```

[68]:
```
show_elbow_silhouette(df_wine_kmeans, wine_feats_dec, max_k=10)
```




```
Knee located at k= 3
```

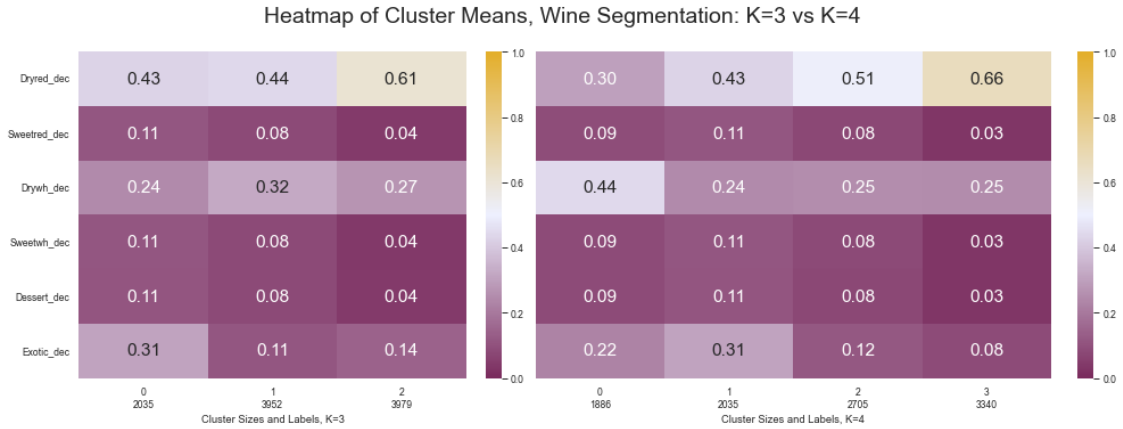
[69]:
```
#show_elbow_silhouette(df_wine_kmeans, wine_feats_dec, max_k=20)
```

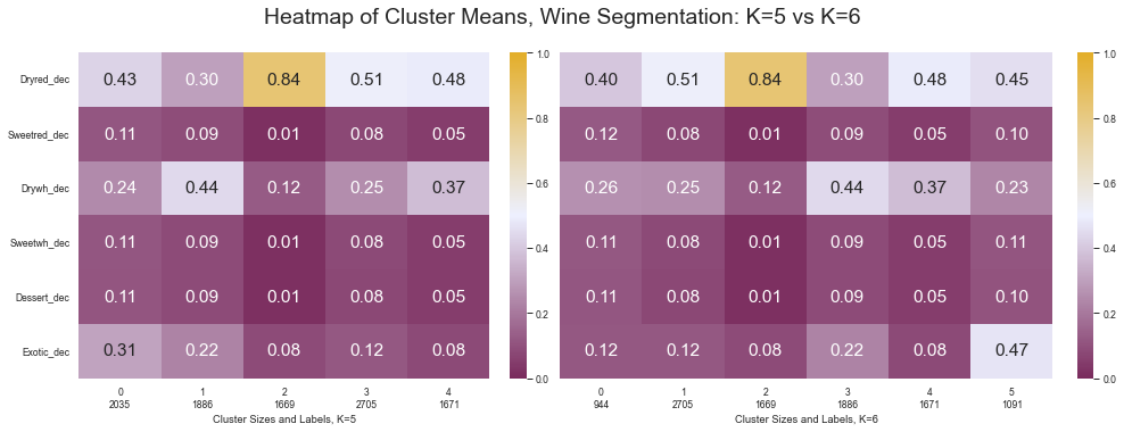
[70]:
```
df_wk = df_wine_kmeans.copy()

for k in range(2,10):
    wine_kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=1000,␣
 ↪n_init=30, random_state=RANDOM_STATE)
    wine_k_labels = wine_kmeans.fit_predict(df_wk)

    df_wk['wine_'+str(k)] = wine_k_labels
    r2_(df_wk[wine_feats_dec+['wine_'+str(k)]],'wine_'+str(k))
```

```
    r_append = {
        'method': 'kmeans',
        'r2': r2_(df_wk[wine_feats_dec+['wine_'+str(k)]],'wine_'+str(k)),
        'clusters':k
    }
    r_scores_wine = r_scores_wine.append(r_append, ignore_index=True)
```

2.3.1 Compare cluster means of different K

[71]:
```
compare_cluster_heatmap(df_wk, 3, 4, 'wine_3', 'wine_4', wine_feats_dec,␣
↪'Heatmap of Cluster Means, Wine Segmentation: K=3 vs K=4')

compare_cluster_heatmap(df_wk, 5, 6, 'wine_5', 'wine_6', wine_feats_dec,␣
↪'Heatmap of Cluster Means, Wine Segmentation: K=5 vs K=6')
```





2.3.2 Characterize final wine segmentation clusters

[72]:
```
r_scores_wine
```

[72]:

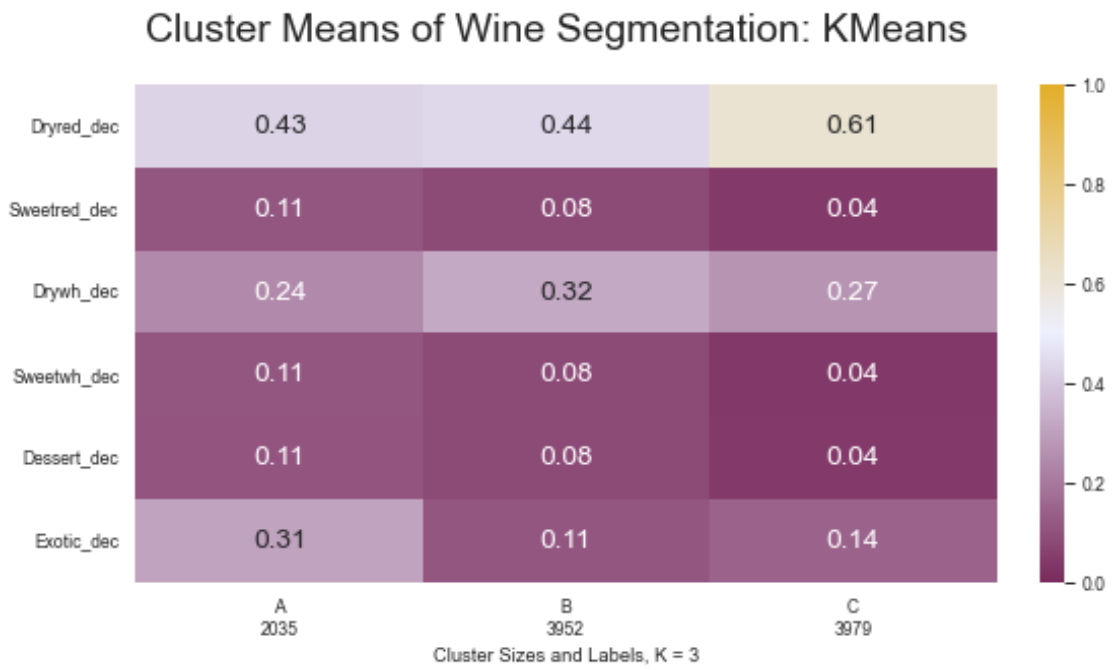
	method	r2	clusters
0	kmeans	0.698357	2
1	kmeans	0.850825	3
2	kmeans	0.937769	4
3	kmeans	0.966988	5
4	kmeans	0.974940	6
5	kmeans	0.984187	7
6	kmeans	0.991201	8
7	kmeans	0.995349	9

[73]:
```
if SAVE_PLOTS:
    r_scores_wine.to_csv('../../out/data/r_scores_wine.csv')
```

[74]:
```
wine_k = 3
wine_kmeans = KMeans(n_clusters=wine_k, init='k-means++', max_iter=1000000,␣
 ↪n_init=30, random_state=RANDOM_STATE)
wine_k_labels = wine_kmeans.fit_predict(df_wine_kmeans)

df_wine_kmeans['wine_labels'] = wine_k_labels
r2_(df_wine_kmeans,'wine_labels')
```

[74]:
```
0.8984956072253663
```

[75]:
```
show_cluster_heatmap(df_wine_kmeans, 'wine_labels', wine_feats_dec, 'Cluster␣
 ↪Means of Wine Segmentation: KMeans', ["A","B","C"])
```



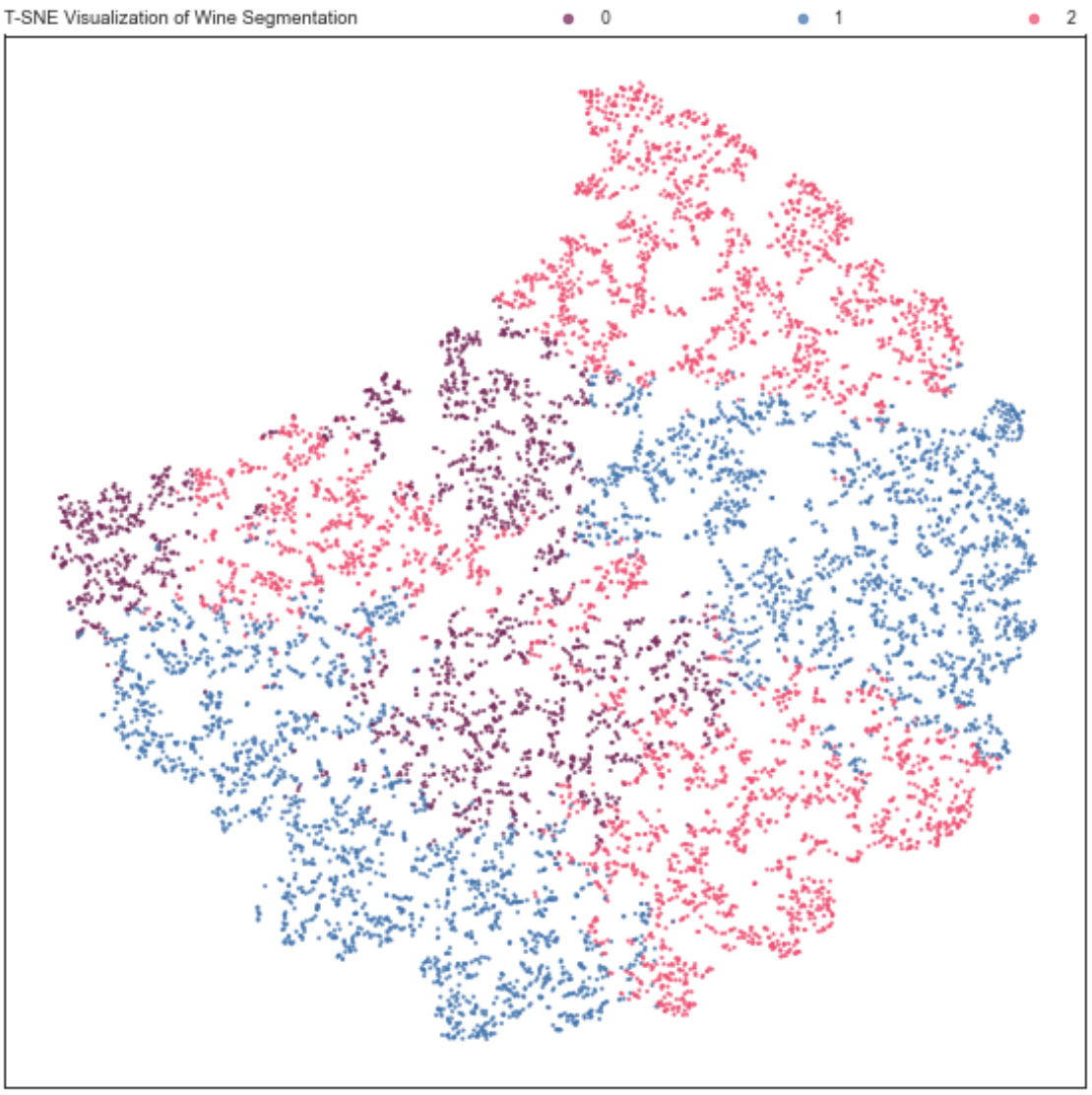
[ ]:

2.3.3 Visualize wine segmentation clusters

[ ]:

[76]:
```
wine_tsne = TSNE(random_state=RANDOM_STATE).
↪fit_transform(df_wine_kmeans[wine_feats_dec])
```

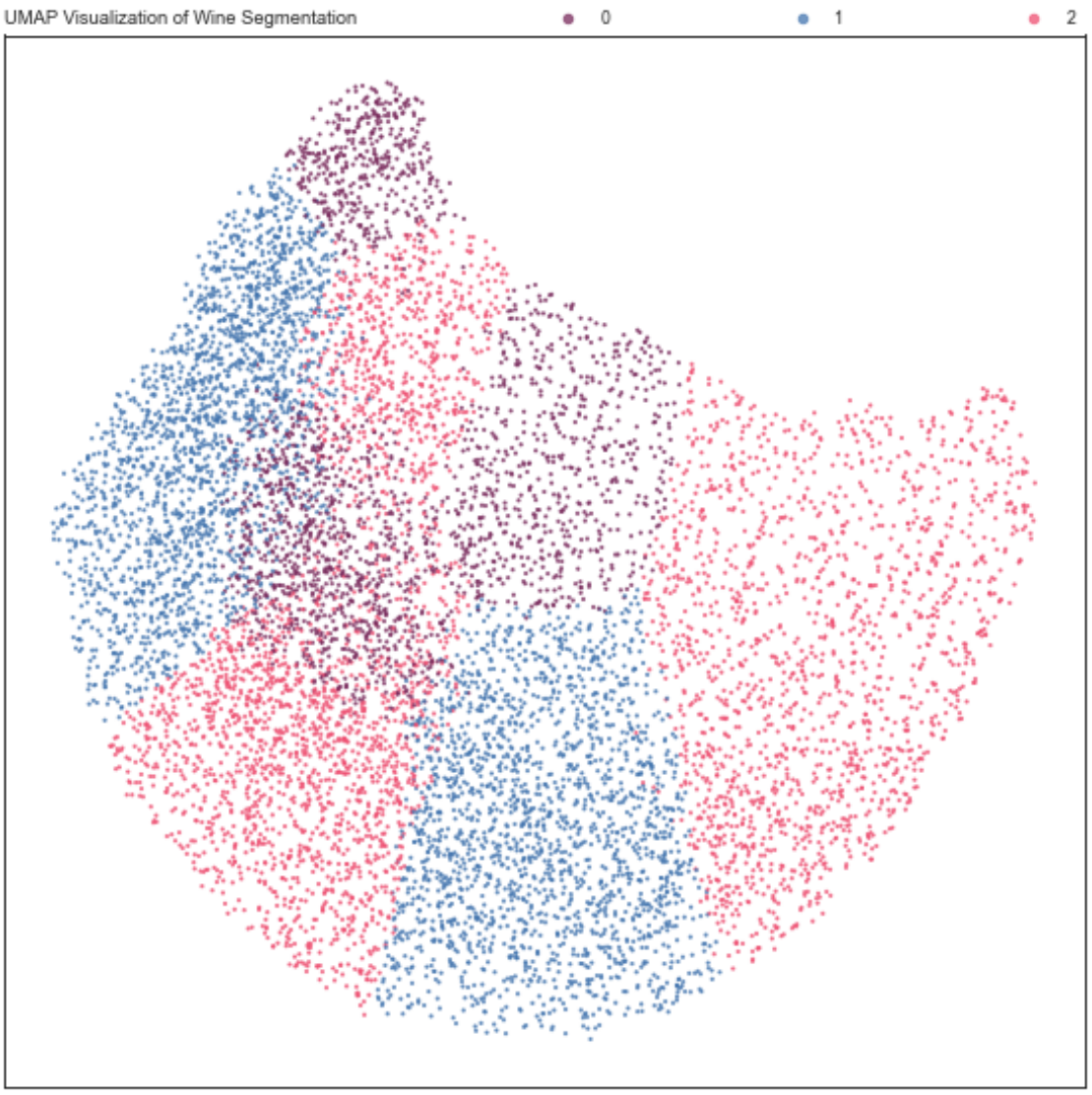
[77]:
```
tsne_wine_df = pd.DataFrame(wine_tsne)
plot_clusters(tsne_wine_df, df_wine_kmeans['wine_labels'], 'T-SNE Visualization␣
↪of Wine Segmentation')
```



[78]:
```
wine_umap = umap.UMAP(random_state=RANDOM_STATE, metric='euclidean',␣
↪min_dist=1, n_neighbors=150, n_components=2)\
                    .fit_transform(df_wine_kmeans[wine_feats_dec])
umap_wine_df = pd.DataFrame(wine_umap)
```

```
OMP: Info #271: omp_set_nested routine deprecated, please use
omp_set_max_active_levels instead.
```

[79]:
```
plot_clusters(umap_wine_df, df_wine_kmeans['wine_labels'], 'UMAP Visualization␣
↪of Wine Segmentation')
```



[ ]:

[ ]:

2.4 Value Segmentation

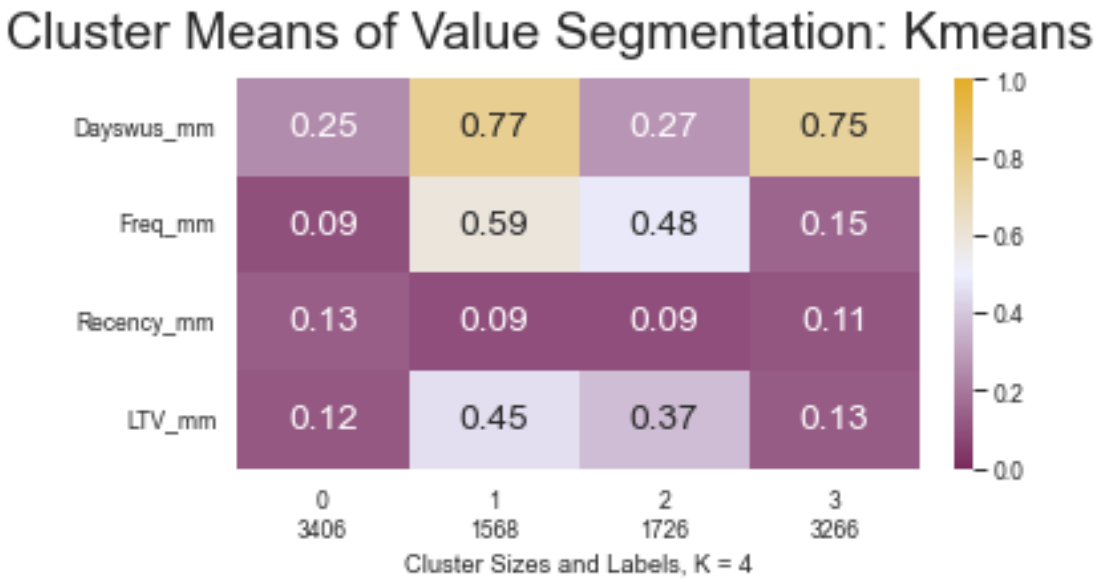
[80]:
```
df_value_kmeans = df[value_feats_mm].copy()
df_value_hclust = df[value_feats_mm].copy()
```

[ ]:

[81]:
```
#show_elbow_silhouette(df_value_kmeans, value_feats_mm, max_k=20)
```

[82]:
```
value_k = 4
value_kmeans = KMeans(n_clusters=value_k, init='k-means++', n_init=100,␣
↪random_state=RANDOM_STATE)
value_k_labels = value_kmeans.fit_predict(df_value_kmeans)
df_value_kmeans['value_labels'] = value_k_labels
```

[83]:
```
show_cluster_heatmap(df_value_kmeans, 'value_labels', value_feats_mm, 'Cluster␣
↪Means of Value Segmentation: Kmeans')
```



[84]:
```
r2_(df_value_kmeans,'value_labels')
```

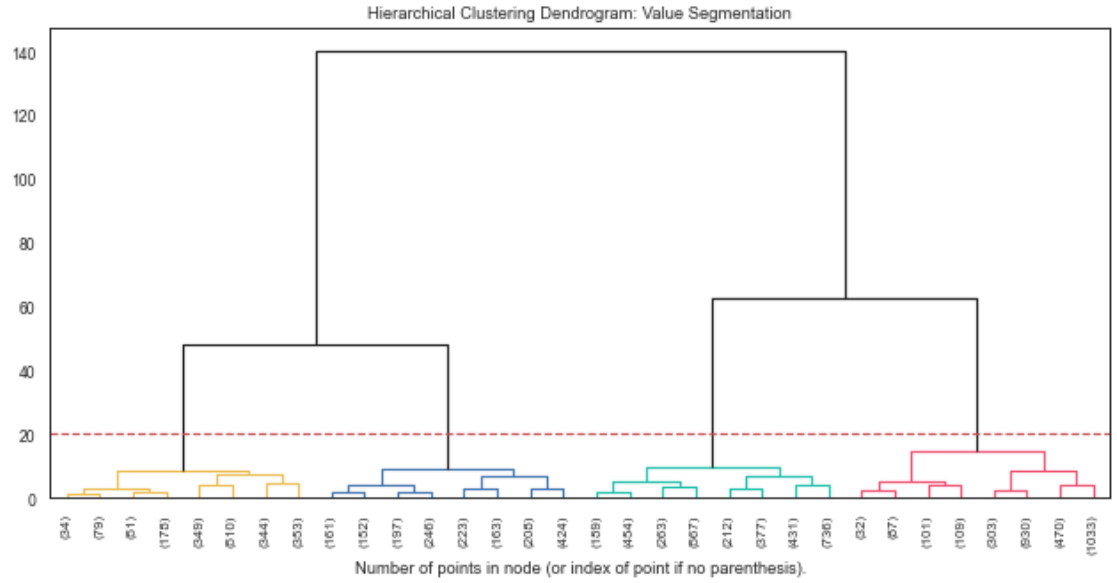
[84]:
```
0.9686037696186998
```

[85]:
```
linkage = 'ward'
distance = 'euclidean'
value_hclust = AgglomerativeClustering(linkage=linkage, affinity=distance,␣
↪n_clusters=value_k)
value_labels = value_hclust.fit_predict(df_value_hclust)

df_value_hclust['value_labels'] = value_labels
```

[ ]:
```

```

[86]:
```
hclust = AgglomerativeClustering(
    linkage='ward',
    affinity='euclidean',
    distance_threshold=0,
```

```
    n_clusters=None
)
hclust_labels = hclust.fit_predict(df_value_hclust)
plot_dendrogram(hclust, title='Hierarchical Clustering Dendrogram: Value␣
↪Segmentation',\
                truncate_mode="level", p=4, \
                above_threshold_color='k', color_threshold=20)
```



[87]:
```
df_vh = df_value_kmeans.copy()

for k in range(2,10):
    value_hclust = AgglomerativeClustering(linkage=linkage, affinity=distance,␣
↪n_clusters=k)
    value_h_labels = value_hclust.fit_predict(df_vh)

    df_vh['value_'+str(k)] = value_h_labels

    r_append = {
        'method': 'hclust',
        'r2': r2_(df_vh[value_feats_mm+['value_'+str(k)]],'value_'+str(k)),
        'clusters':k
    }
    r_scores_value = r_scores_value.append(r_append, ignore_index=True)


#r_scores_value
```

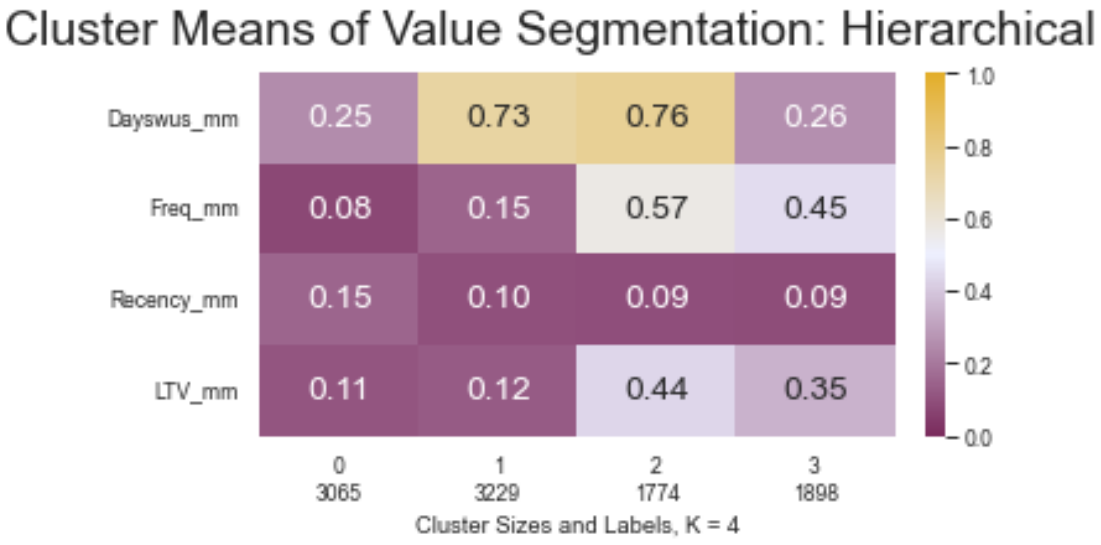
[88]:
```
df_vk = df_value_kmeans.copy()

for k in range(2,10):
    value_kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=1000,␣
 ↪n_init=30, random_state=RANDOM_STATE)
    value_k_labels = value_kmeans.fit_predict(df_vk)

    df_vk['value_'+str(k)] = value_k_labels

    r_append = {
        'method': 'kmeans',
        'r2': r2_(df_vk[value_feats_mm+['value_'+str(k)]],'value_'+str(k)),
        'clusters':k
    }
    r_scores_value = r_scores_value.append(r_append, ignore_index=True)


#r_scores_value
```

[ ]:

[ ]:

2.4.1 Characterize value segmentation clusters

[89]:
```
show_cluster_heatmap(df_value_hclust, 'value_labels', value_feats_mm, 'Cluster␣
 ↪Means of Value Segmentation: Hierarchical')
```

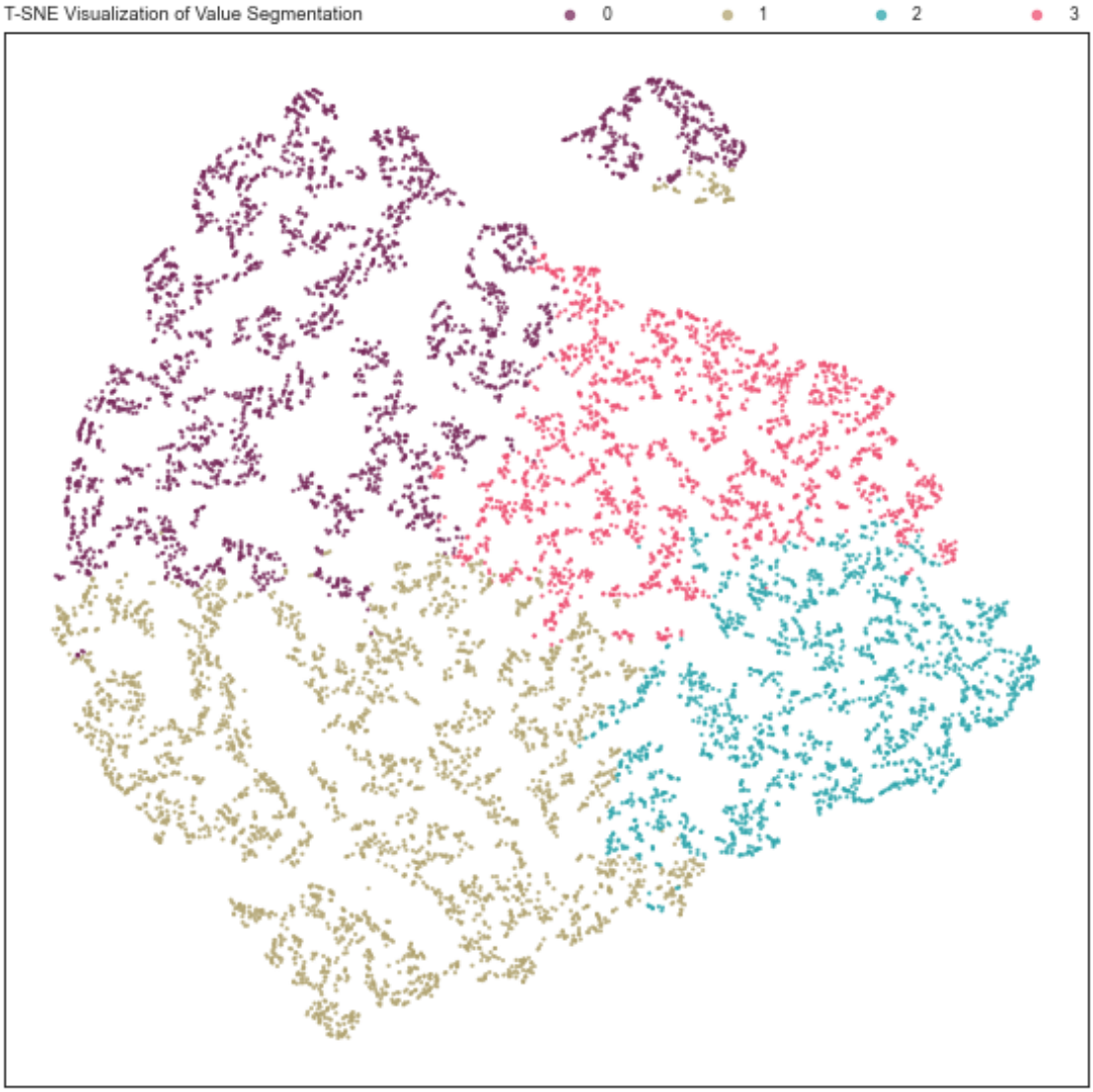


[ ]:

2.4.2 Visualize Value Segmentation

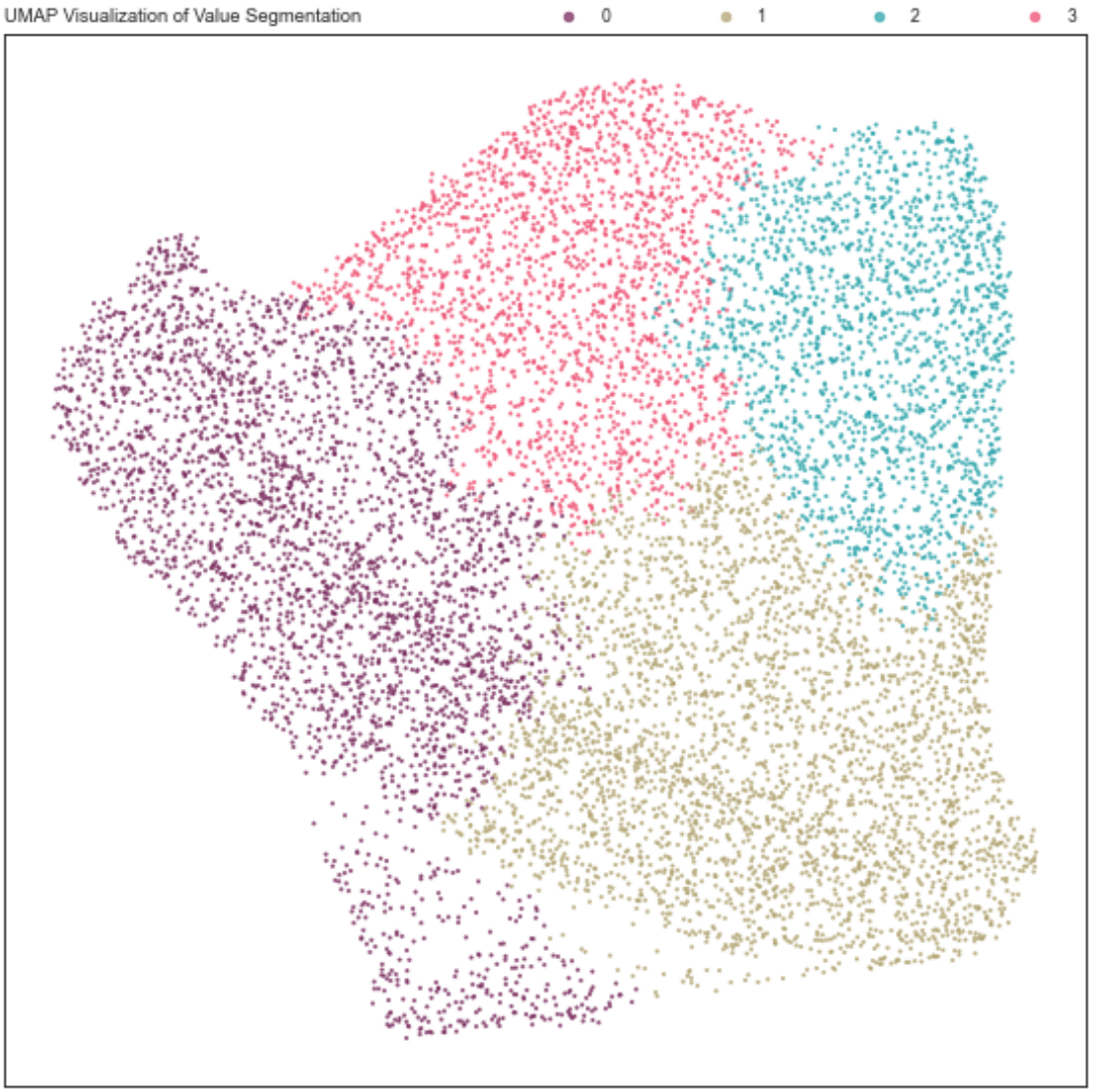
[90]:
```
value_tsne = TSNE(random_state=RANDOM_STATE).
↪fit_transform(df_value_hclust[value_feats_mm])
```

[91]:
```
tsne_value_df = pd.DataFrame(value_tsne)
plot_clusters(tsne_value_df, df_value_hclust['value_labels'], 'T-SNE␣
↪Visualization of Value Segmentation')
```



[92]:
```
value_umap = umap.UMAP(random_state=RANDOM_STATE, metric='euclidean',␣
↪min_dist=1, n_neighbors=150, n_components=2)\
                    .fit_transform(df_value_hclust[value_feats_mm])
umap_value_df = pd.DataFrame(value_umap)
```

[93]:
```
plot_clusters(umap_value_df, df_value_hclust['value_labels'], 'UMAP␣
↪Visualization of Value Segmentation')
```



2.5 Merging cluster solutions

[94]:
```
rfm_df[rfm_features]
```

[94]:
```
         Freq  Recency  Monetary
Custid
5325     20.0     18.0     826.0
3956     36.0     33.0    1852.0
3681      4.0     56.0      39.0
2829      2.0     46.0      37.0
8788      2.0      3.0      36.0
…         …        …         …
7989      4.0     33.0      59.0
1383     19.0     59.0     776.0
4070     18.0     45.0     720.0
7909      3.0     65.0      47.0
4914     25.0     28.0    1148.0

[9547 rows x 3 columns]
```

[95]:
```
df_merged = pd.merge(df_value_hclust, df_wine_kmeans, left_index=True,␣
 ↪right_index=True)
#df_merged = pd.merge(df_merged, rfm_df[rfm_features + rfm_rank_features],␣
 ↪left_index=True, right_index=True)

df_merged
```

[95]:
```
         Dayswus_mm   Freq_mm  Recency_mm    LTV_mm  value_labels  Dryred_dec  \
Custid
5325       0.147143  0.358491    0.032787  0.348824             3        0.67
3956       0.701429  0.660377    0.060109  0.401456             2        0.49
3681       0.165714  0.056604    0.102004  0.095745             0        0.04
2829       0.712857  0.018868    0.083789  0.096305             1        0.86
8788       0.410000  0.018868    0.005464  0.101904             0        0.85
…             …         …         …         …            …          …
1383       0.831429  0.339623    0.107468  0.204367             1        0.78
4070       0.065714  0.320755    0.081967  0.318589             3        0.30
7909       0.098571  0.037736    0.118397  0.102464             0        0.06
4158       0.795714  0.000000    0.670310  0.100784             0        0.18
4914       0.612857  0.452830    0.051002  0.263718             2        0.63

         Sweetred_dec  Drywh_dec  Sweetwh_dec  Dessert_dec  Exotic_dec  labels  \
Custid
5325             0.04       0.26         0.02         0.01        0.01       0
3956             0.00       0.46         0.01         0.03        0.00       4
3681             0.29       0.14         0.32         0.21        0.48       6
2829             0.01       0.11         0.01         0.01        0.55       7
8788             0.00       0.12         0.02         0.01        0.28       5
…                   …          …            …            …           …       …
1383             0.00       0.20         0.01         0.01        0.11       5
4070             0.12       0.36         0.10         0.12        0.13       2
```

```
7909            0.24       0.10         0.38         0.22         0.41       6
4158            0.13       0.45         0.11         0.13         0.13       2
4914            0.10       0.13         0.11         0.03         0.04       0

        wine_labels
Custid
5325              1
3956              2
3681              0
2829              0
8788              2
…               …
1383              2
4070              1
7909              0
4158              1
4914              1

[9966 rows x 13 columns]
```

[96]:
```
#df_merged = df_value_hclust.copy()
#df_merged = df_merged.merge(df_wine_kmeans, left_index=True, right_index=True )
```

[97]:
```
df_merged
df_merged.columns
```

[97]:
```
Index(['Dayswus_mm', 'Freq_mm', 'Recency_mm', 'LTV_mm', 'value_labels',
       'Dryred_dec', 'Sweetred_dec', 'Drywh_dec', 'Sweetwh_dec', 'Dessert_dec',
       'Exotic_dec', 'labels', 'wine_labels'],
      dtype='object')
```

[ ]:
```

```

[98]:
```
r_scores_value.sort_values(by='r2', ascending=False)
```

[98]:
```
    method        r2 clusters
7   hclust  0.995631        9
15  kmeans  0.994842        9
6   hclust  0.994579        8
14  kmeans  0.991816        8
13  kmeans  0.991080        7
5   hclust  0.990391        7
4   hclust  0.986736        6
12  kmeans  0.984106        6
3   hclust  0.968897        5
11  kmeans  0.967528        5
2   hclust  0.958701        4
```

```
10  kmeans  0.958701        4
1   hclust  0.906906        3
9   kmeans  0.843697        3
0   hclust  0.690580        2
8   kmeans  0.603878        2
```

[99]:
```
wine_letters = {0:"A", 1: "B", 2: "C"}

df_merged['wine_letters'] = df_merged['wine_labels']
df_merged = df_merged.replace({'wine_letters':wine_letters})
```

[100]:
```
df_merged.groupby(['value_labels', 'wine_labels'])\
    .size()\
    .to_frame()\
    .reset_index()\
    .pivot('value_labels', 'wine_labels', 0).fillna('-')
```

[100]:
```
wine_labels     0     1     2
value_labels
0             747  1150  1168
1             752  1153  1324
2             261   807   706
3             275   842   781
```

[101]:
```
df_merged['merged_labels'] = df_merged['value_labels'].astype(str) +␣
 ↪df_merged['wine_letters']

merged_labels_list = sorted(df_merged['merged_labels'].unique().tolist())

df_merged
```

[101]:
```
         Dayswus_mm   Freq_mm  Recency_mm    LTV_mm  value_labels  Dryred_dec  \
Custid
5325       0.147143  0.358491    0.032787  0.348824             3        0.67
3956       0.701429  0.660377    0.060109  0.401456             2        0.49
3681       0.165714  0.056604    0.102004  0.095745             0        0.04
2829       0.712857  0.018868    0.083789  0.096305             1        0.86
8788       0.410000  0.018868    0.005464  0.101904             0        0.85
…               …         …           …         …             …           …
1383       0.831429  0.339623    0.107468  0.204367             1        0.78
4070       0.065714  0.320755    0.081967  0.318589             3        0.30
7909       0.098571  0.037736    0.118397  0.102464             0        0.06
4158       0.795714  0.000000    0.670310  0.100784             0        0.18
4914       0.612857  0.452830    0.051002  0.263718             2        0.63

         Sweetred_dec  Drywh_dec  Sweetwh_dec  Dessert_dec  Exotic_dec  labels  \
Custid
```

```
5325            0.04       0.26         0.02         0.01         0.01       0
3956            0.00       0.46         0.01         0.03         0.00       4
3681            0.29       0.14         0.32         0.21         0.48       6
2829            0.01       0.11         0.01         0.01         0.55       7
8788            0.00       0.12         0.02         0.01         0.28       5
…                …          …            …            …            …          …
1383            0.00       0.20         0.01         0.01         0.11       5
4070            0.12       0.36         0.10         0.12         0.13       2
7909            0.24       0.10         0.38         0.22         0.41       6
4158            0.13       0.45         0.11         0.13         0.13       2
4914            0.10       0.13         0.11         0.03         0.04       0

        wine_labels wine_letters merged_labels
Custid
5325              1            B            3B
3956              2            C            2C
3681              0            A            0A
2829              0            A            1A
8788              2            C            0C
…               …            …            …
1383              2            C            1C
4070              1            B            3B
7909              0            A            0A
4158              1            B            0B
4914              1            B            2B

[9966 rows x 15 columns]
```

[102]:
```
vw_feats = value_feats_mm+wine_feats_dec
df_centroids = df_merged.groupby(['value_labels', 'wine_labels'])[vw_feats].
 ↪mean()
```

[103]:
```
df_centroids
```

[103]:
```
                           Dayswus_mm   Freq_mm  Recency_mm    LTV_mm  \
value_labels wine_labels
0            0               0.251484  0.067717    0.144578  0.107790
             1               0.242666  0.070664    0.160470  0.107864
             2               0.254052  0.087216    0.147466  0.110933
1            0               0.734599  0.133154    0.096481  0.120885
             1               0.728457  0.135905    0.095708  0.120105
             2               0.725268  0.162016    0.099134  0.124515
2            0               0.771423  0.524181    0.095820  0.403657
             1               0.752700  0.590307    0.089935  0.456416
             2               0.766641  0.560532    0.094622  0.425182
3            0               0.252681  0.419828    0.085564  0.324290
             1               0.256659  0.463586    0.090372  0.360893
```

```
                2              0.260298  0.446332    0.090750  0.340963

                                Dryred_dec  Sweetred_dec  Drywh_dec  Sweetwh_dec  \
value_labels wine_labels
0            0                  0.419357      0.115730   0.243280     0.113333
             1                  0.417548      0.087243   0.318130     0.088017
             2                  0.611378      0.041122   0.264092     0.040856
1            0                  0.426463      0.112061   0.236981     0.112633
             1                  0.426513      0.087580   0.307667     0.091674
             2                  0.620748      0.038799   0.259418     0.039207
2            0                  0.446322      0.105057   0.257816     0.097356
             1                  0.457633      0.067361   0.341586     0.068092
             2                  0.606969      0.038399   0.280113     0.036771
3            0                  0.428364      0.104364   0.254655     0.107018
             1                  0.462910      0.069181   0.332280     0.071033
             2                  0.600948      0.040717   0.282996     0.036965

                                Dessert_dec  Exotic_dec
value_labels wine_labels
0            0                    0.108072    0.359920
             1                    0.088930    0.141443
             2                    0.042175    0.187757
1            0                    0.112021    0.364668
             1                    0.086765    0.142715
             2                    0.041208    0.179585
2            0                    0.093180    0.152069
             1                    0.065266    0.072144
             2                    0.037535    0.074943
3            0                    0.105455    0.140691
             1                    0.064252    0.071532
             2                    0.037657    0.075583
```

[104]:
```
df_ = df_merged.copy()
```

[105]:
```
df_.groupby('merged_labels').count()[vw_feats]
```

[105]:
```
               Dayswus_mm  Freq_mm  Recency_mm  LTV_mm  Dryred_dec  \
merged_labels
0A                    747      747         747     747         747
0B                   1150     1150        1150    1150        1150
0C                   1168     1168        1168    1168        1168
1A                    752      752         752     752         752
1B                   1153     1153        1153    1153        1153
1C                   1324     1324        1324    1324        1324
2A                    261      261         261     261         261
2B                    807      807         807     807         807
2C                    706      706         706     706         706
```

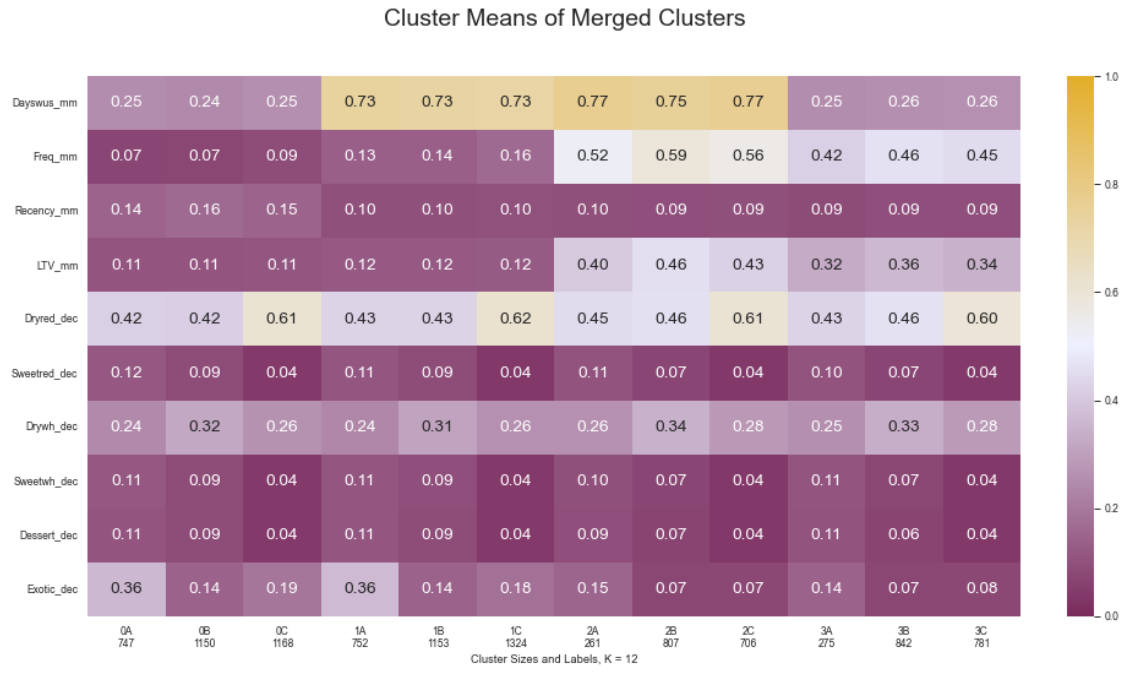
3A	275	275	275	275	275
3B	842	842	842	842	842
3C	781	781	781	781	781

merged_labels	Sweetred_dec	Drywh_dec	Sweetwh_dec	Dessert_dec	Exotic_dec
0A	747	747	747	747	747
0B	1150	1150	1150	1150	1150
0C	1168	1168	1168	1168	1168
1A	752	752	752	752	752
1B	1153	1153	1153	1153	1153
1C	1324	1324	1324	1324	1324
2A	261	261	261	261	261
2B	807	807	807	807	807
2C	706	706	706	706	706
3A	275	275	275	275	275
3B	842	842	842	842	842
3C	781	781	781	781	781

[106]:
```
if SAVE_PLOTS:
    df_.to_csv('../../out/data/mergedclusters.csv')
```

2.5.1 Visualizing merged cluster solution

[107]:
```
show_cluster_heatmap(df_, 'merged_labels', vw_feats, 'Cluster Means of Merged␣
↪Clusters')
```



[108]:
```
merged_tsne = TSNE(random_state=RANDOM_STATE).fit_transform(df_[vw_feats])
```

[109]:
```
df_[vw_feats]
```

[109]:

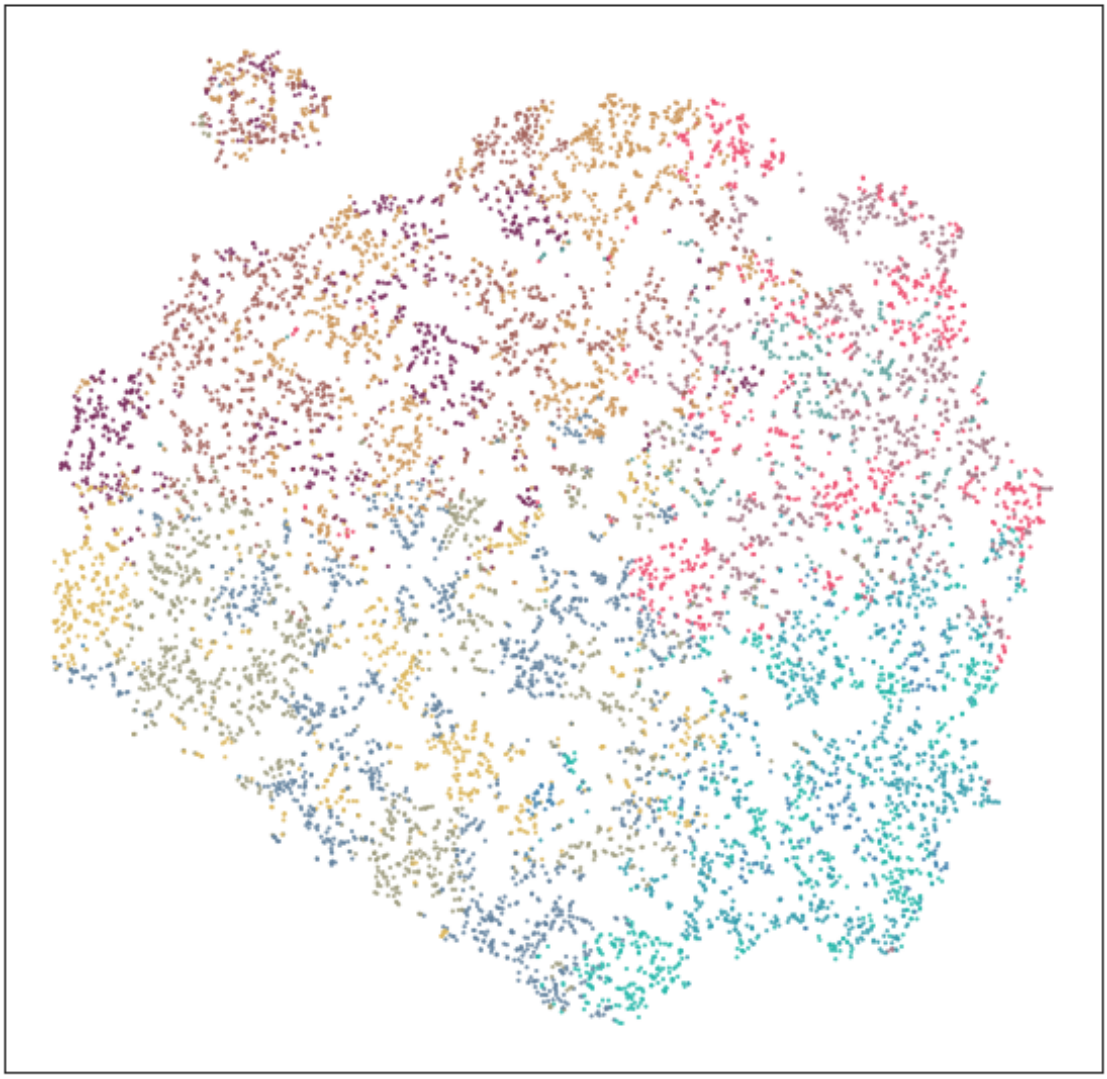
Custid	Dayswus_mm	Freq_mm	Recency_mm	LTV_mm	Dryred_dec	Sweetred_dec	\
5325	0.147143	0.358491	0.032787	0.348824	0.67	0.04	
3956	0.701429	0.660377	0.060109	0.401456	0.49	0.00	
3681	0.165714	0.056604	0.102004	0.095745	0.04	0.29	
2829	0.712857	0.018868	0.083789	0.096305	0.86	0.01	
8788	0.410000	0.018868	0.005464	0.101904	0.85	0.00	
…	…	…	…	…	…	…	
1383	0.831429	0.339623	0.107468	0.204367	0.78	0.00	
4070	0.065714	0.320755	0.081967	0.318589	0.30	0.12	
7909	0.098571	0.037736	0.118397	0.102464	0.06	0.24	
4158	0.795714	0.000000	0.670310	0.100784	0.18	0.13	
4914	0.612857	0.452830	0.051002	0.263718	0.63	0.10	

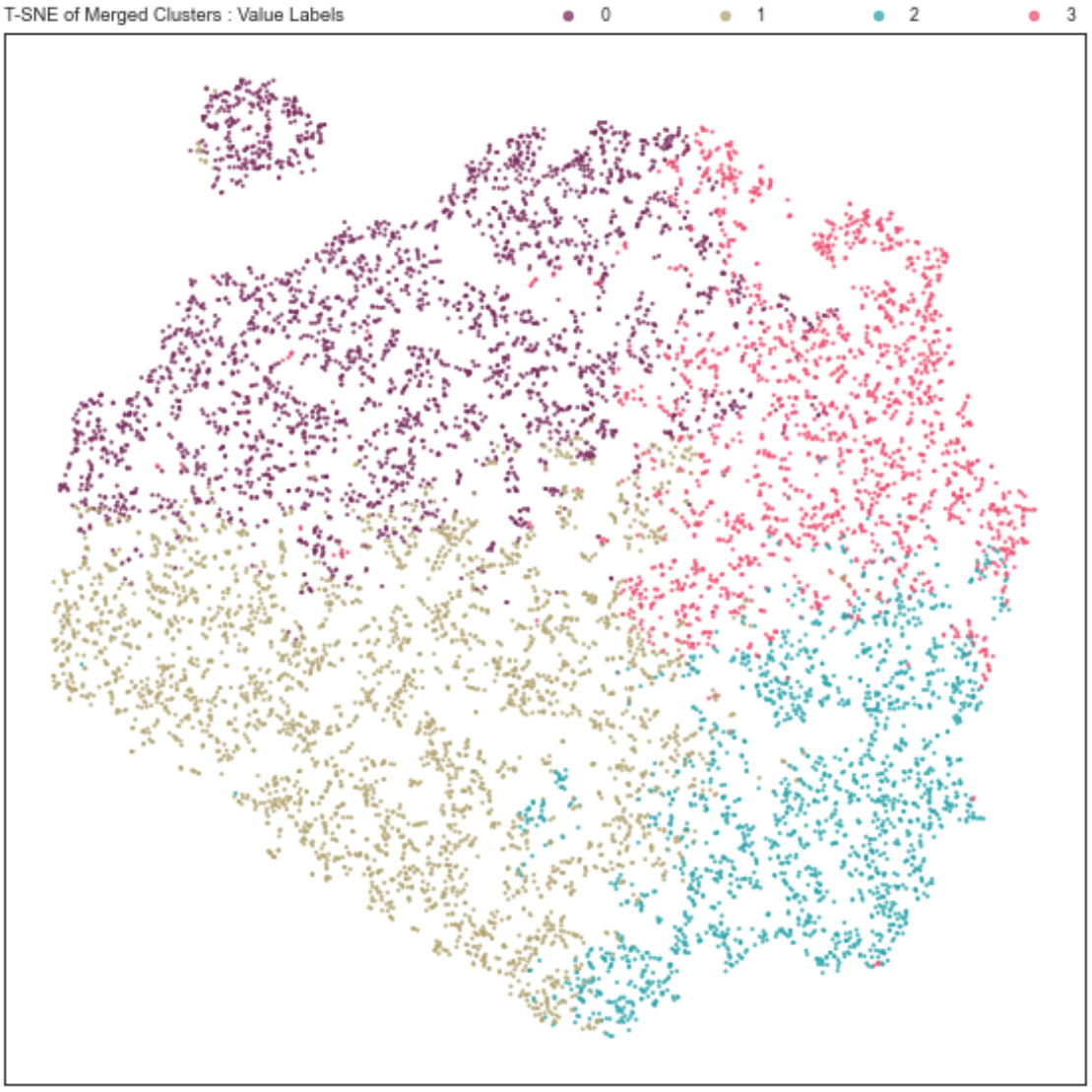
Custid	Drywh_dec	Sweetwh_dec	Dessert_dec	Exotic_dec
5325	0.26	0.02	0.01	0.01
3956	0.46	0.01	0.03	0.00
3681	0.14	0.32	0.21	0.48
2829	0.11	0.01	0.01	0.55
8788	0.12	0.02	0.01	0.28
…	…	…	…	…
1383	0.20	0.01	0.01	0.11
4070	0.36	0.10	0.12	0.13
7909	0.10	0.38	0.22	0.41
4158	0.45	0.11	0.13	0.13
4914	0.13	0.11	0.03	0.04

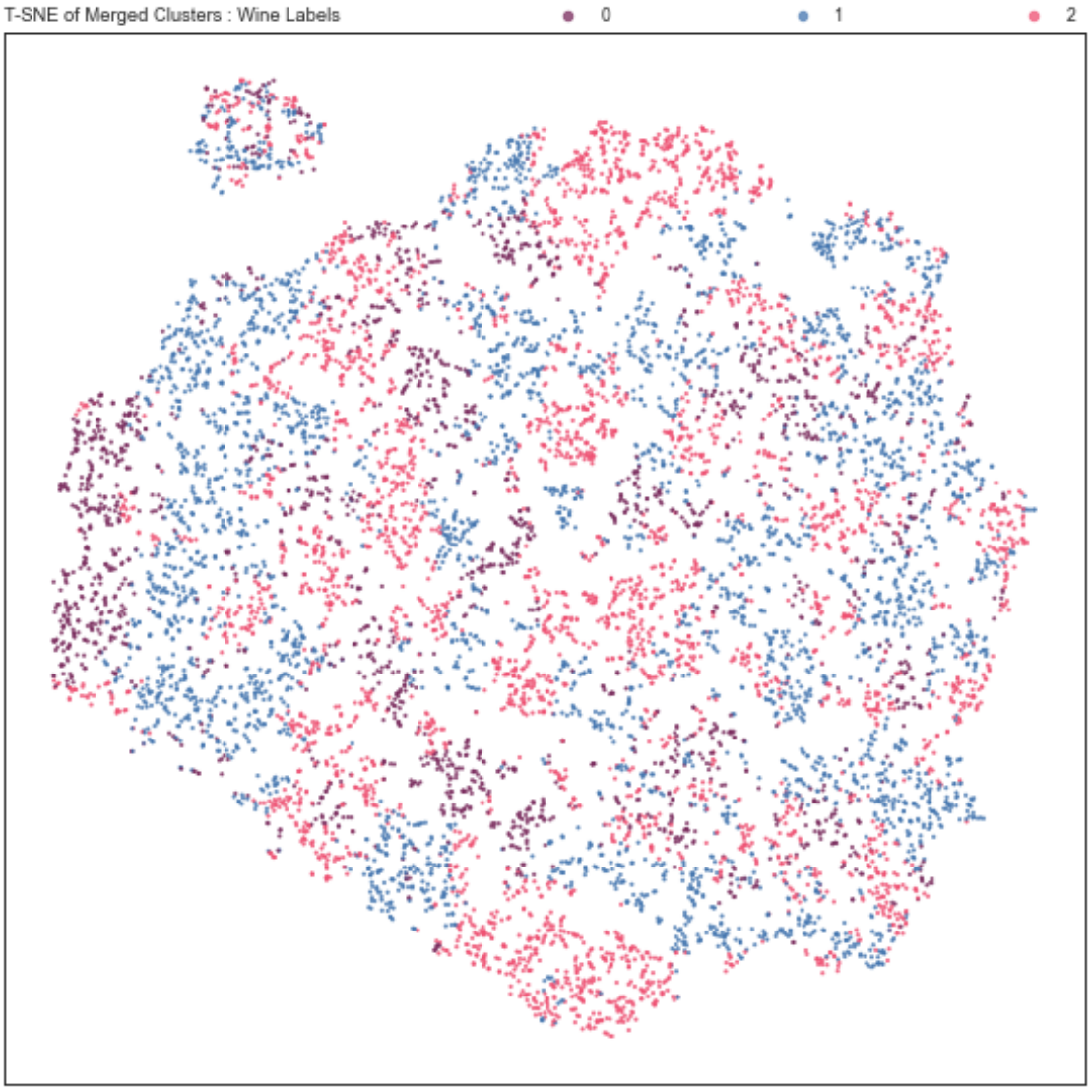
[9966 rows x 10 columns]

[110]:
```
tsne_merged_df = pd.DataFrame(merged_tsne)

plot_clusters(tsne_merged_df, df_['merged_labels'], 'T-SNE Visualization of␣
↪Merged Clusters', merged_labels_list)
plot_clusters(tsne_merged_df, df_['value_labels'], 'T-SNE of Merged Clusters :␣
↪Value Labels')
plot_clusters(tsne_merged_df, df_['wine_labels'], 'T-SNE of Merged Clusters :␣
↪Wine Labels')
```

T-SNE Visualization of Merged Clusters



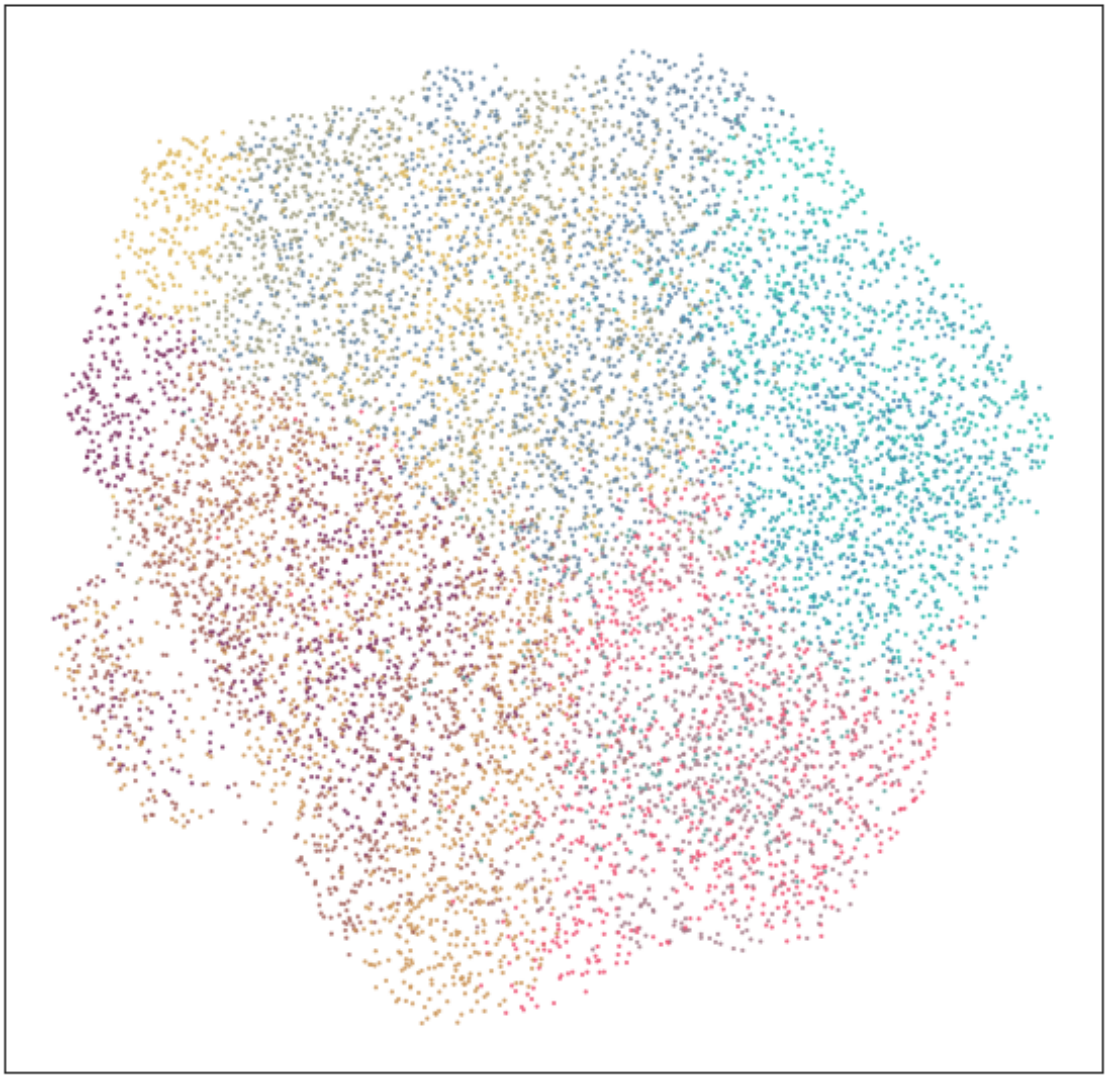

T-SNE of Merged Clusters : Value Labels
0
1
2
3



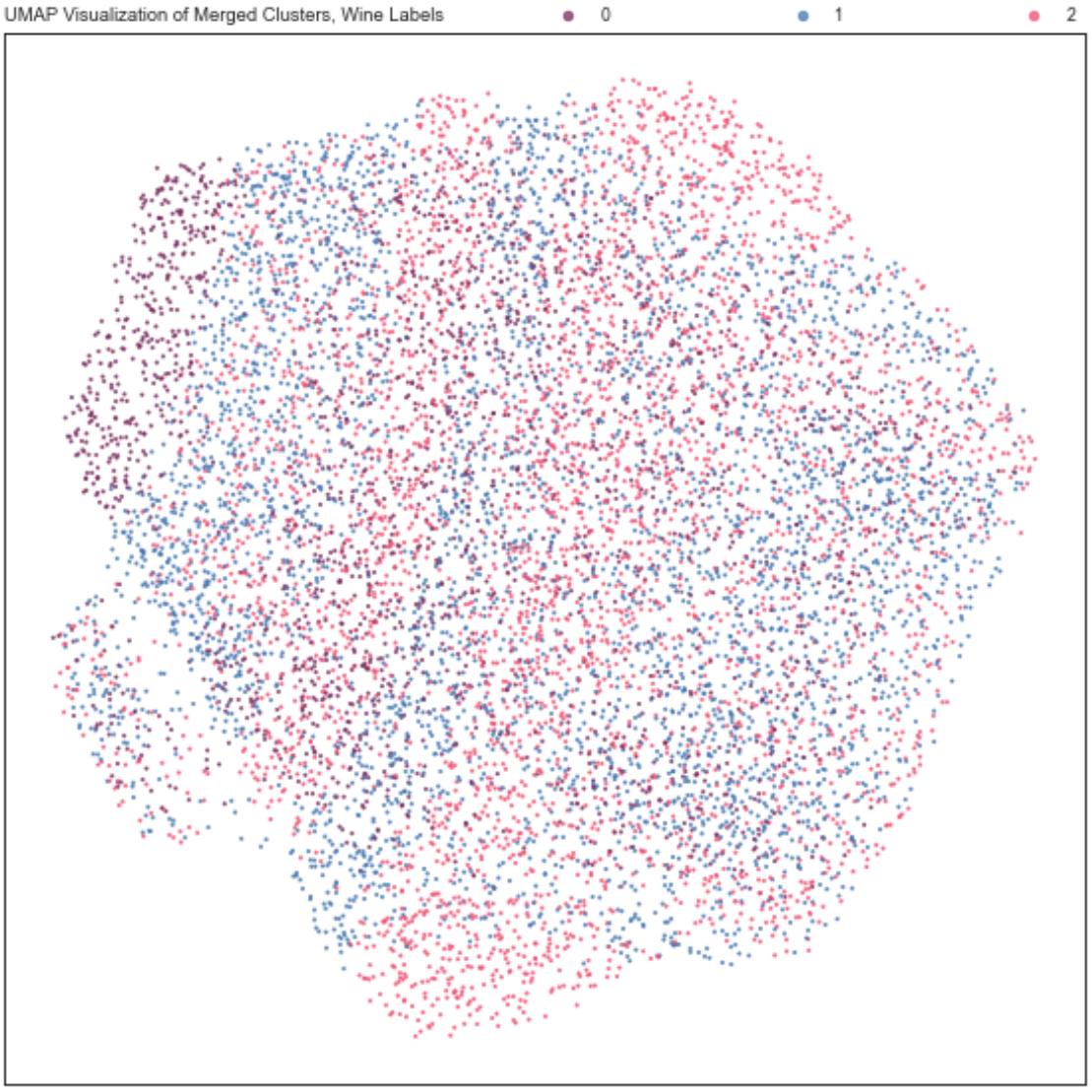
[111]:
```
merged_umap = umap.UMAP(random_state=RANDOM_STATE, metric='euclidean',␣
↪min_dist=1, n_neighbors=150, n_components=2)\
                    .fit_transform(df_[vw_feats])
umap_merged_df = pd.DataFrame(merged_umap)
```

[112]:
```
plot_clusters(umap_merged_df, df_['merged_labels'], 'UMAP Visualization of␣
↪Merged Clusters', merged_labels_list)
plot_clusters(umap_merged_df, df_['value_labels'], 'UMAP Visualization of␣
↪Merged Clusters, Value Labels')
plot_clusters(umap_merged_df, df_['wine_labels'], 'UMAP Visualization of Merged␣
↪Clusters, Wine Labels')
```

UMAP Visualization of Merged Clusters




UMAP Visualization of Merged Clusters, Value Labels
0
1
2
3



[113]:
```
#plot_clusters(umap_merged_df, df_['wine_labels'], 'UMAP of Merged Clusters: Wine Labels')
#plot_clusters(umap_merged_df, df_['value_labels'], 'UMAP of Merged Clusters: Value Labels')
```

[114]:
```
df_
```

[114]:

Custid	Dayswus_mm	Freq_mm	Recency_mm	LTV_mm	value_labels	Dryred_dec	\
5325	0.147143	0.358491	0.032787	0.348824	3	0.67	
3956	0.701429	0.660377	0.060109	0.401456	2	0.49	
3681	0.165714	0.056604	0.102004	0.095745	0	0.04	

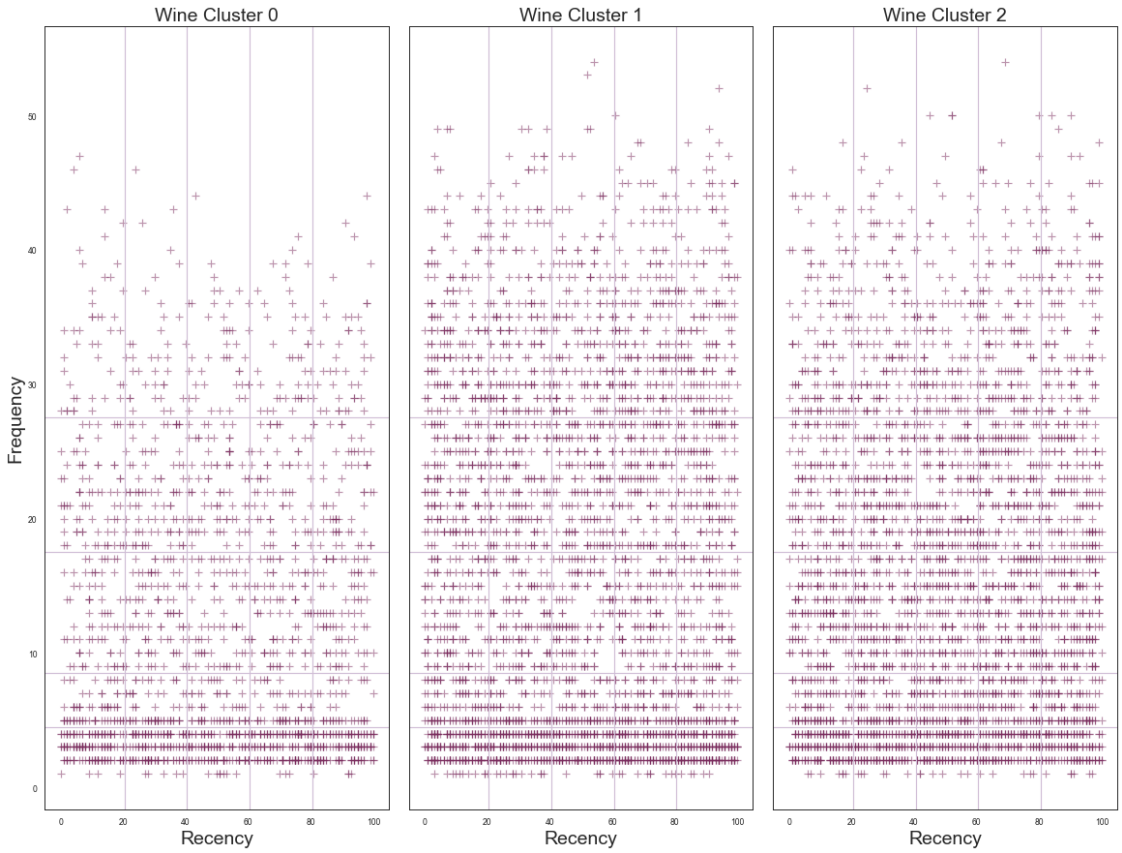
```
2829      0.712857  0.018868    0.083789  0.096305             1         0.86
8788      0.410000  0.018868    0.005464  0.101904             0         0.85
…              …         …           …         …             …            …
1383      0.831429  0.339623    0.107468  0.204367             1         0.78
4070      0.065714  0.320755    0.081967  0.318589             3         0.30
7909      0.098571  0.037736    0.118397  0.102464             0         0.06
4158      0.795714  0.000000    0.670310  0.100784             0         0.18
4914      0.612857  0.452830    0.051002  0.263718             2         0.63

        Sweetred_dec  Drywh_dec  Sweetwh_dec  Dessert_dec  Exotic_dec  labels  \
Custid
5325            0.04       0.26         0.02         0.01        0.01       0
3956            0.00       0.46         0.01         0.03        0.00       4
3681            0.29       0.14         0.32         0.21        0.48       6
2829            0.01       0.11         0.01         0.01        0.55       7
8788            0.00       0.12         0.02         0.01        0.28       5
…                  …          …            …            …           …       …
1383            0.00       0.20         0.01         0.01        0.11       5
4070            0.12       0.36         0.10         0.12        0.13       2
7909            0.24       0.10         0.38         0.22        0.41       6
4158            0.13       0.45         0.11         0.13        0.13       2
4914            0.10       0.13         0.11         0.03        0.04       0

        wine_labels wine_letters merged_labels
Custid
5325              1            B            3B
3956              2            C            2C
3681              0            A            0A
2829              0            A            1A
8788              2            C            0C
…               …            …            …
1383              2            C            1C
4070              1            B            3B
7909              0            A            0A
4158              1            B            0B
4914              1            B            2B

[9966 rows x 15 columns]
```

2.6 Merge Wine and RFM

[115]:
```
#df_wine_rfm = df_wine_kmeans.copy()
df_wine_rfm = pd.concat([df_wine_kmeans,rfm_df],axis=1)
df_wine_rfm['wine_labels'].unique().tolist()
```

[115]:
```
[1, 2, 0]
```

[116]:
```
wine_labels_list = sorted(df_wine_rfm['wine_labels'].unique().tolist())

fig, axes = plt.subplots(1,len(wine_labels_list), figsize=(17,13), sharey=True)

wine_markers = ['.','+','x']
for l, ax in zip(wine_labels_list, axes.flatten()):
    for q in range(len(quantile_values)):
        ax.axvline(quantile_values[q]+.49,
               label=str(int(quantile_values[q]))+'th Percentile',
               color=COLORS[3])

        ax.axhline(freq_quints[q]+.5,
               label=str(int(quantile_values[q]))+'th Percentile',
               color=COLORS[3])

    ax.scatter(data=df_wine_rfm.loc[df_wine_rfm['wine_labels']==l,:],␣
 ↪x='Recency', y='Freq', #hue='wine_labels',
               alpha=.5, s=80, marker=wine_markers[1], cmap=CAT_CMAP,
                         )

    if l== 0:
        ax.set_ylabel('Frequency', fontsize=20)
    ax.set_xlabel('Recency', fontsize=20)

    ax.set_title('Wine Cluster '+str(l), fontsize=20)
    #ax.legend([])
    #rfm_title='Recency vs Frequency Quintiles'

    #plt.suptitle(rfm_title, fontsize=20, y=.91)
    #plt.margins(0.025, 0.05)
    #if SAVE_PLOTS:
    #    save_fig(rfm_title, fig)
plt.tight_layout()
plt.show()
```



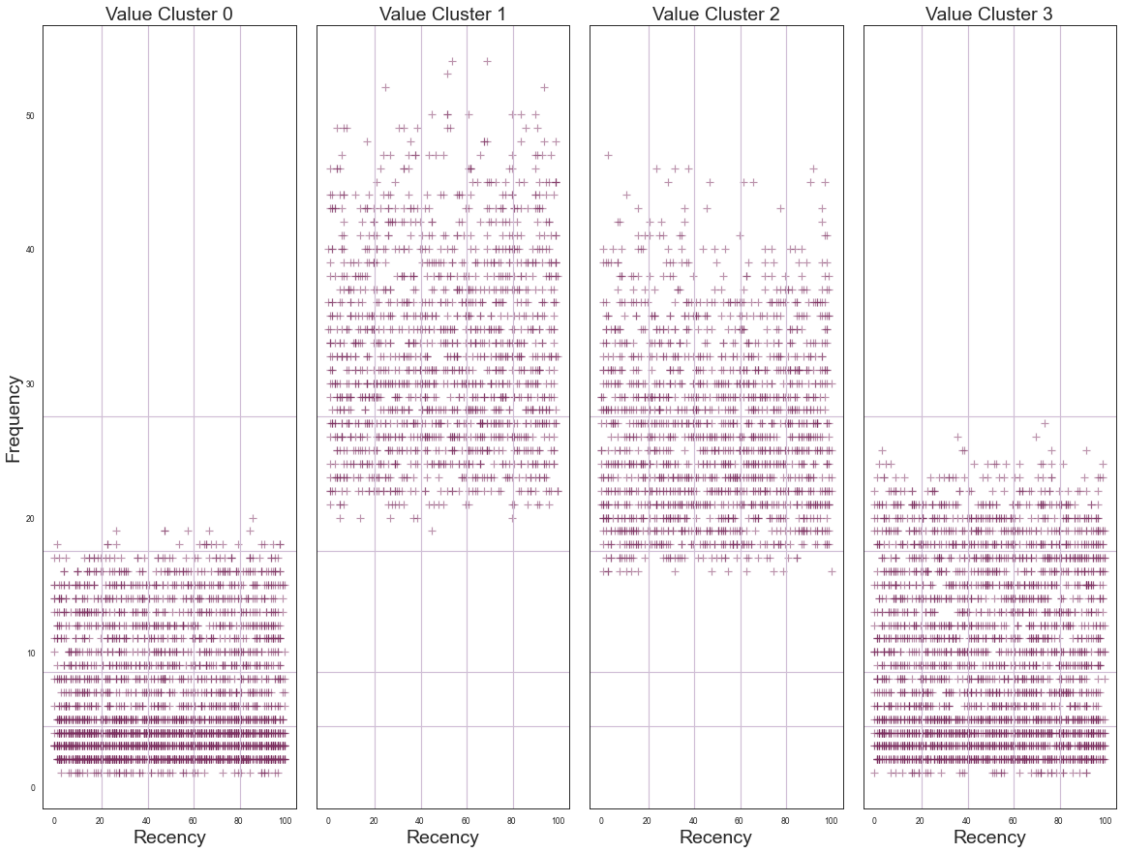
2.7 Merging Value and RFM

[117]:
```
df_value_rfm = pd.concat([df_value_kmeans,rfm_df],axis=1)
sorted(df_value_rfm['value_labels'].unique().tolist())
```

[117]: [0, 1, 2, 3]

[118]:
```
val_labels_list = sorted(df_value_rfm['value_labels'].unique().tolist())

fig, axes = plt.subplots(1,len(val_labels_list), figsize=(17,13), sharey=True)

val_markers = ['.','+','x']
for l, ax in zip(val_labels_list, axes.flatten()):
    for q in range(len(quantile_values)):
        ax.axvline(quantile_values[q]+.49,
               label=str(int(quantile_values[q]))+'th Percentile',
               color=COLORS[3])

        ax.axhline(freq_quints[q]+.5,
               label=str(int(quantile_values[q]))+'th Percentile',
```

```
                color=COLORS[3])

    ax.scatter(data=df_value_rfm.loc[df_value_rfm['value_labels']==l,:],␣
↪x='Recency', y='Freq', #hue='wine_labels',
                alpha=.5, s=80, marker=val_markers[1], cmap=CAT_CMAP,
                          )

    if l== 0:
        ax.set_ylabel('Frequency', fontsize=20)
    ax.set_xlabel('Recency', fontsize=20)

    ax.set_title('Value Cluster '+str(l), fontsize=20)
    #ax.legend([])
    #rfm_title='Recency vs Frequency Quintiles'

    #plt.suptitle(rfm_title, fontsize=20, y=.91)
    #plt.margins(0.025, 0.05)
    #if SAVE_PLOTS:
    #    save_fig(rfm_title, fig)
plt.tight_layout()
plt.show()
```



[ ]:

3 Characterizing final clusters

[119]:
```
## Function to plot histograms of numeric features for specified dataframe
def plot_final_histo_box(df, features, col, title = "Final Clusters: Relative␣
 ↪Distributions of Numeric Variables"):
    if show_plots:

        rows = sorted(df[col].unique().tolist())

        merged_labels_map = {}
        for i in range(len(rows)):
            merged_labels_map[rows[i]] = i

        cols = range(len(features))

        fig = plt.figure(figsize=(22,32), \
                         constrained_layout=True)



        subfigs = fig.subfigures(len(rows), len(features), facecolor='#fdfdfd')
        for c in cols: # feats
            for r in rows: # clusters
                ri = merged_labels_map[r]
                color = cm.viridis(ri / len(rows))

                df_ = df.loc[df[col]==r,[features[c]]]
                axs = subfigs[ri][c].subplots(2, 1, sharex='col', \
                                gridspec_kw={'height_ratios': [4,1]})
                axs[0].hist(df_, color=color)
                axs[0].set_xlim(0,1)
                axs[0].set_title(features[c], y=1, fontsize=20)

                sns.boxplot(x=df_[features[c]], ax=axs[1], color=color)
                axs[1].set_xlabel(None)

                if c==0:
                    csize = ' [ ' + str(len(df_)) + ' ]'
                    axs[0].set_ylabel('Cluster '+ str(r)+csize)

        plt.suptitle(title, fontsize=24)
        if SAVE_PLOTS:
            save_fig(title, fig)
```

```
        plt.show()
    else:
        print("show_plots is currently set to False")
```

[120]:
```
df_
```

[120]:

Custid	Dayswus_mm	Freq_mm	Recency_mm	LTV_mm	value_labels	Dryred_dec
5325	0.147143	0.358491	0.032787	0.348824	3	0.67
3956	0.701429	0.660377	0.060109	0.401456	2	0.49
3681	0.165714	0.056604	0.102004	0.095745	0	0.04
2829	0.712857	0.018868	0.083789	0.096305	1	0.86
8788	0.410000	0.018868	0.005464	0.101904	0	0.85
…	…	…	…	…	…	…
1383	0.831429	0.339623	0.107468	0.204367	1	0.78
4070	0.065714	0.320755	0.081967	0.318589	3	0.30
7909	0.098571	0.037736	0.118397	0.102464	0	0.06
4158	0.795714	0.000000	0.670310	0.100784	0	0.18
4914	0.612857	0.452830	0.051002	0.263718	2	0.63

Custid	Sweetred_dec	Drywh_dec	Sweetwh_dec	Dessert_dec	Exotic_dec	labels
5325	0.04	0.26	0.02	0.01	0.01	0
3956	0.00	0.46	0.01	0.03	0.00	4
3681	0.29	0.14	0.32	0.21	0.48	6
2829	0.01	0.11	0.01	0.01	0.55	7
8788	0.00	0.12	0.02	0.01	0.28	5
…	…	…	…	…	…	…
1383	0.00	0.20	0.01	0.01	0.11	5
4070	0.12	0.36	0.10	0.12	0.13	2
7909	0.24	0.10	0.38	0.22	0.41	6
4158	0.13	0.45	0.11	0.13	0.13	2
4914	0.10	0.13	0.11	0.03	0.04	0

Custid	wine_labels	wine_letters	merged_labels
5325	1	B	3B
3956	2	C	2C
3681	0	A	0A
2829	0	A	1A
8788	2	C	0C
…	…	…	…
1383	2	C	1C
4070	1	B	3B
7909	0	A	0A

```
4158              1            B            0B
4914              1            B            2B

[9966 rows x 15 columns]
```

[121]:
```
plot_final_histo_box(df_, wine_feats_dec, 'merged_labels', title='Final␣
↪Clusters, Wine Features')
plot_final_histo_box(df_, value_feats_mm, 'merged_labels', title='Final␣
↪Clusters, Value Features')
```


Final Clusters, Wine Features
Dryred_dec
Sweetred_dec
Drywh_dec
Sweetwh_dec
Dessert_dec
Exotic_dec


Final Clusters, Value Features
Dayswus_mm
Freq_mm
Recency_mm
LTV_mm

[122]:
```
df_['merged_labels']
```

[122]:
```
Custid
5325    3B
3956    2C
3681    0A
2829    1A
8788    0C
        ..
1383    1C
4070    3B
7909    0A
4158    0B
4914    2B
Name: merged_labels, Length: 9966, dtype: object
```

[ ]:
```

```

[123]:
```
#plot_final_histo_box(df, other_features, 'merged_labels', title='Final␣
↪Clusters, Other Features')
```

[124]:
```
#plot_final_histo_box(df_, wine_feats_dec, 'wine_labels', title='Wine␣
↪Segmentation, Wine Features')
#plot_final_histo_box(df_, value_feats_mm, 'value_labels', title='Value␣
↪Segmentation, Value Features')
```

[125]:
```
df_allfeats = pd.merge(df_original, df_, how='right',left_index=True,␣
↪right_index=True, suffixes=('', '_drop')) #.drop_duplicates()
```

[126]:
```
def hist_cluster_vs_all(df, k, col, feats, title='Mean Values, Clusters vs␣
↪Population'):

    clustermeans_ = df.groupby(col).mean()[feats].reset_index()

    fig, axes = plt.subplots(2,int(len(feats)/2), figsize=(15,7),␣
↪constrained_layout=True)
    i = 0

    colors = [ (cm.viridis(float(i) / len(clustermeans_['merged_labels']))) for␣
↪i in range(len(clustermeans_['merged_labels'])) ]

    for m, ax in zip(feats, axes.flatten()):

        ax.bar(height=clustermeans_.loc[:,m], x=clustermeans_['merged_labels'].
↪astype(str), color=colors)
        m_ = df[feats].mean()[m]
        ax.axhline(m_, label='Pop. mean', color='r',linestyle='dashed')
```

```
        ax.set_xlabel('Cluster Labels')
        if (m == wine_feats_dec[-1])|(m == value_feats_mm[-1])|(m ==␣
↪other_features[-1]):
            ax.legend(bbox_to_anchor=(1,1.02), loc="upper right", frameon=False)
        ax.set_title(m)

    fig.suptitle(title, fontsize=20)

    save_fig(title, fig)

    plt.show()
```

[127]:
```
hist_cluster_vs_all(df_allfeats,5,'merged_labels', value_feats_mm, title='Mean␣
↪Values of Value Features, Clusters vs Population')
```

[128]:
```
hist_cluster_vs_all(df_allfeats,5,'merged_labels', wine_feats_dec, title='Mean␣
↪Values of Wine Features, Clusters vs Population')
```

[129]:
```
hist_cluster_vs_all(df_allfeats,5,'merged_labels', other_features, title='Mean␣
↪Values of Other Features, Clusters vs Population')
```

4 Exploring the clusters

[130]:
```
df_[value_feats_mm]
df_[wine_feats_dec]
```

[130]:
```
         Dryred_dec  Sweetred_dec  Drywh_dec  Sweetwh_dec  Dessert_dec  \
Custid
```

```
5325          0.67          0.04       0.26          0.02          0.01
3956          0.49          0.00       0.46          0.01          0.03
3681          0.04          0.29       0.14          0.32          0.21
2829          0.86          0.01       0.11          0.01          0.01
8788          0.85          0.00       0.12          0.02          0.01
…             …             …          …             …             …
1383          0.78          0.00       0.20          0.01          0.01
4070          0.30          0.12       0.36          0.10          0.12
7909          0.06          0.24       0.10          0.38          0.22
4158          0.18          0.13       0.45          0.11          0.13
4914          0.63          0.10       0.13          0.11          0.03

        Exotic_dec
Custid
5325          0.01
3956          0.00
3681          0.48
2829          0.55
8788          0.28
…             …
1383          0.11
4070          0.13
7909          0.41
4158          0.13
4914          0.04

[9966 rows x 6 columns]
```

[131]:
```
def show_value_wine_scatter(df, row_feats, col_feats, title='Scatter Plot of␣
↪Features'):
    fig, axes = plt.subplots(len(row_feats), len(col_feats),
                         figsize=(15,11), sharex=True)

    for row in range(len(row_feats)):
        for col in range(len(col_feats)):
            x=df[row_feats[row]]
            y=df[col_feats[col]]
            axes[row][col].scatter(x,y, alpha=.25, s=1)

            if col == 0:
                axes[row][col].set_ylabel(row_feats[row])

            if row == 0:
                axes[row][col].set_title(col_feats[col], y=1)

    plt.suptitle(title)
    plt.tight_layout()
```

```
    plt.show()
```

[132]:
```
df_
```

[132]:

Custid	Dayswus_mm	Freq_mm	Recency_mm	LTV_mm	value_labels	Dryred_dec
5325	0.147143	0.358491	0.032787	0.348824	3	0.67
3956	0.701429	0.660377	0.060109	0.401456	2	0.49
3681	0.165714	0.056604	0.102004	0.095745	0	0.04
2829	0.712857	0.018868	0.083789	0.096305	1	0.86
8788	0.410000	0.018868	0.005464	0.101904	0	0.85
…	…	…	…	…	…	…
1383	0.831429	0.339623	0.107468	0.204367	1	0.78
4070	0.065714	0.320755	0.081967	0.318589	3	0.30
7909	0.098571	0.037736	0.118397	0.102464	0	0.06
4158	0.795714	0.000000	0.670310	0.100784	0	0.18
4914	0.612857	0.452830	0.051002	0.263718	2	0.63

Custid	Sweetred_dec	Drywh_dec	Sweetwh_dec	Dessert_dec	Exotic_dec	labels
5325	0.04	0.26	0.02	0.01	0.01	0
3956	0.00	0.46	0.01	0.03	0.00	4
3681	0.29	0.14	0.32	0.21	0.48	6
2829	0.01	0.11	0.01	0.01	0.55	7
8788	0.00	0.12	0.02	0.01	0.28	5
…	…	…	…	…	…	…
1383	0.00	0.20	0.01	0.01	0.11	5
4070	0.12	0.36	0.10	0.12	0.13	2
7909	0.24	0.10	0.38	0.22	0.41	6
4158	0.13	0.45	0.11	0.13	0.13	2
4914	0.10	0.13	0.11	0.03	0.04	0

Custid	wine_labels	wine_letters	merged_labels
5325	1	B	3B
3956	2	C	2C
3681	0	A	0A
2829	0	A	1A
8788	2	C	0C
…	…	…	…
1383	2	C	1C
4070	1	B	3B
7909	0	A	0A
4158	1	B	0B
4914	1	B	2B

[9966 rows x 15 columns]

[133]:
```
df_allfeats.loc[df_allfeats['Recency']<=100,:]
```

[133]:

Custid	Dayswus	Age	Edu	Income	Freq	Recency	Monetary	LTV	\
5325	653.0	55.0	20.0	78473.0	20.0	18.0	826.0	445.0	
3956	1041.0	75.0	18.0	105087.0	36.0	33.0	1852.0	539.0	
3681	666.0	18.0	12.0	27984.0	4.0	56.0	39.0	-7.0	
2829	1049.0	42.0	16.0	61748.0	2.0	46.0	37.0	-6.0	
8788	837.0	47.0	16.0	65789.0	2.0	3.0	36.0	4.0	
…	…	…	…	…	…	…	…	…	
7989	774.0	43.0	18.0	42853.0	4.0	33.0	59.0	-17.0	
1383	1132.0	57.0	20.0	81033.0	19.0	59.0	776.0	187.0	
4070	596.0	66.0	15.0	84714.0	18.0	45.0	720.0	391.0	
7909	619.0	18.0	12.0	40466.0	3.0	65.0	47.0	5.0	
4914	979.0	55.0	16.0	94926.0	25.0	28.0	1148.0	293.0	

Custid	Perdeal	Dryred	…	Dryred_dec	Sweetred_dec	Drywh_dec	\
5325	7.0	67.0	…	0.67	0.04	0.26	
3956	2.0	49.0	…	0.49	0.00	0.46	
3681	88.0	4.0	…	0.04	0.29	0.14	
2829	70.0	86.0	…	0.86	0.01	0.11	
8788	35.0	85.0	…	0.85	0.00	0.12	
…	…	…	…	…	…	…	
7989	90.0	73.0	…	0.73	0.01	0.25	
1383	22.0	78.0	…	0.78	0.00	0.20	
4070	5.0	30.0	…	0.30	0.12	0.36	
7909	23.0	6.0	…	0.06	0.24	0.10	
4914	7.0	63.0	…	0.63	0.10	0.13	

Custid	Sweetwh_dec	Dessert_dec	Exotic_dec	labels	wine_labels	\
5325	0.02	0.01	0.01	0	1	
3956	0.01	0.03	0.00	4	2	
3681	0.32	0.21	0.48	6	0	
2829	0.01	0.01	0.55	7	0	
8788	0.02	0.01	0.28	5	2	
…	…	…	…	…	…	
7989	0.01	0.00	0.07	0	1	
1383	0.01	0.01	0.11	5	2	
4070	0.10	0.12	0.13	2	1	
7909	0.38	0.22	0.41	6	0	
4914	0.11	0.03	0.04	0	1	

Custid	wine_letters	merged_labels
5325	B	3B

```
3956                C              2C
3681                A              0A
2829                A              1A
8788                C              0C
…                   …              …
7989                B              0B
1383                C              1C
4070                B              3B
7909                A              0A
4914                B              2B

[9547 rows x 32 columns]
```

[134]:
```
show_value_wine_scatter(df_, value_feats_mm, wine_feats_dec, title='Scatter␣
↪Plot of Wine Features against Value Features')
```

[135]:
```
#clusterlabels = sorted(df_['wine_labels'].unique().tolist())

#for c in clusterlabels:
#    plottitle ='Scatter Plot of Wine Features against Value Features: Cluster␣
↪'+ str(c)
```

```
#    show_value_wine_scatter(df_.loc[df_['wine_labels']==c,:], value_feats_mm,␣
↪wine_feats_dec, plottitle)
```

[ ]:

[136]:
```
df_original
df_
```

[136]:

Custid	Dayswus_mm	Freq_mm	Recency_mm	LTV_mm	value_labels	Dryred_dec
5325	0.147143	0.358491	0.032787	0.348824	3	0.67
3956	0.701429	0.660377	0.060109	0.401456	2	0.49
3681	0.165714	0.056604	0.102004	0.095745	0	0.04
2829	0.712857	0.018868	0.083789	0.096305	1	0.86
8788	0.410000	0.018868	0.005464	0.101904	0	0.85
…	…	…	…	…	…	…
1383	0.831429	0.339623	0.107468	0.204367	1	0.78
4070	0.065714	0.320755	0.081967	0.318589	3	0.30
7909	0.098571	0.037736	0.118397	0.102464	0	0.06
4158	0.795714	0.000000	0.670310	0.100784	0	0.18
4914	0.612857	0.452830	0.051002	0.263718	2	0.63

Custid	Sweetred_dec	Drywh_dec	Sweetwh_dec	Dessert_dec	Exotic_dec	labels
5325	0.04	0.26	0.02	0.01	0.01	0
3956	0.00	0.46	0.01	0.03	0.00	4
3681	0.29	0.14	0.32	0.21	0.48	6
2829	0.01	0.11	0.01	0.01	0.55	7
8788	0.00	0.12	0.02	0.01	0.28	5
…	…	…	…	…	…	…
1383	0.00	0.20	0.01	0.01	0.11	5
4070	0.12	0.36	0.10	0.12	0.13	2
7909	0.24	0.10	0.38	0.22	0.41	6
4158	0.13	0.45	0.11	0.13	0.13	2
4914	0.10	0.13	0.11	0.03	0.04	0

Custid	wine_labels	wine_letters	merged_labels
5325	1	B	3B
3956	2	C	2C
3681	0	A	0A
2829	0	A	1A
8788	2	C	0C
…	…	…	…
1383	2	C	1C
4070	1	B	3B
7909	0	A	0A

```
4158              1            B            0B
4914              1            B            2B

[9966 rows x 15 columns]
```

[137]:
```
df_full = pd.merge(df_original, df_, how='left',left_index=True,␣
↪right_index=True, suffixes=('', '_drop')) #.drop_duplicates()
df_full.drop([col for col in df_full.columns if 'drop' in col], axis=1,␣
↪inplace=True)
```

[ ]:
```

```

[138]:
```
wine_cluster_means2 = df_full.groupby(['merged_labels']).mean()[wine_features]
wine_cluster_means2
#if SAVE_PLOTS:
    #wine_cluster_means2.to_csv('../../out/wine_final_cluster_means_percent.
↪csv')
```

[138]:
```
                  Dryred   Sweetred      Drywh    Sweetwh    Dessert  \
merged_labels
0A              41.935743  11.572959  24.327979  11.333333  10.807229
0B              41.754783   8.724348  31.813043   8.801739   8.893043
0C              61.137842   4.112158  26.409247   4.085616   4.217466
1A              42.646277  11.206117  23.698138  11.263298  11.202128
1B              42.651344   8.758023  30.766696   9.167389   8.676496
1C              62.074773   3.879909  25.941843   3.920695   4.120846
2A              44.632184  10.505747  25.781609   9.735632   9.318008
2B              45.763321   6.736059  34.158612   6.809170   6.526642
2C              60.696884   3.839943  28.011331   3.677054   3.753541
3A              42.836364  10.436364  25.465455  10.701818  10.545455
3B              46.290974   6.918052  33.228029   7.103325   6.425178
3C              60.094750   4.071703  28.299616   3.696543   3.765685

                  Exotic
merged_labels
0A              35.991968
0B              14.144348
0C              18.775685
1A              36.466755
1B              14.271466
1C              17.958459
2A              15.206897
2B               7.214374
2C               7.494334
3A              14.069091
3B               7.153207
3C               7.558259
```

[139]:
```
value_cluster_means = df_full.groupby(['merged_labels']).mean()[value_features]


value_cluster_means_mm = df_full.groupby(['merged_labels']).
↪mean()[value_feats_mm]

if SAVE_PLOTS:
    value_cluster_means.to_csv('../../out/value_final_cluster_means.csv')
    value_cluster_means_mm.to_csv('../../out/value_final_cluster_means_scaled.
↪csv')
```

[140]:
```
other_cluster_means = df_full.groupby(['merged_labels']).
↪mean()[demog_features+other_features]

if SAVE_PLOTS:
    other_cluster_means.to_csv('../../out/othervars_final_cluster_means.csv')
```

[141]:
```
wine_means = df_full.groupby(['wine_labels']).mean()

value_means = df_full.groupby(['value_labels']).mean()

if SAVE_PLOTS:
    wine_means.to_csv('../../out/wine_means.csv')
    value_means.to_csv('../../out/value_means.csv')
```

[142]:
```
value_means.reset_index()
```

[142]:

	value_labels	Dayswus	Age	Edu	Income	Freq	\
0	0.0	724.407830	37.584339	16.619250	53128.786623	5.041436	
1	1.0	1060.005884	38.545060	16.592753	54515.348715	8.736451	
2	2.0	1082.701804	65.222661	17.016347	98096.652198	31.142616	
3	3.0	730.306112	64.777661	16.965227	97328.646997	24.857745	

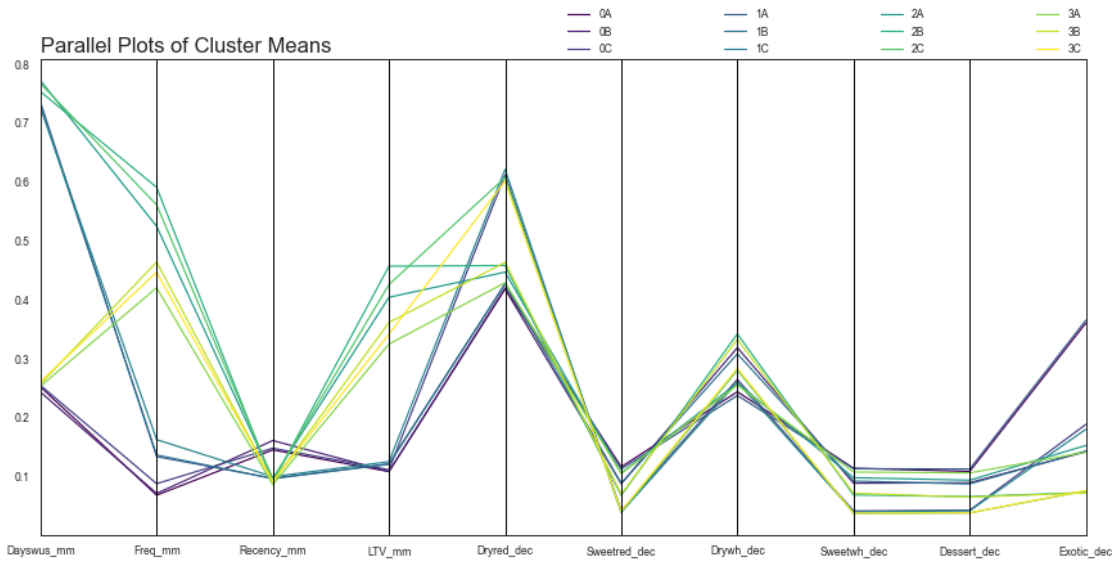
	Recency	Monetary	LTV	Perdeal	…	Recency_mm	LTV_mm	\
0	83.251223	125.668842	16.701142	47.786297	…	0.151642	0.109015	
1	53.413750	286.422731	40.061319	47.184887	…	0.097293	0.122095	
2	50.873732	1540.829200	601.095265	6.127959	…	0.092666	0.436224	
3	49.317176	1145.213383	442.435722	6.681770	…	0.089831	0.347388	

	Dryred_dec	Sweetred_dec	Drywh_dec	Sweetwh_dec	Dessert_dec	Exotic_dec	\
0	0.491853	0.076610	0.279295	0.076215	0.075778	0.212339	
1	0.506144	0.073280	0.271421	0.075042	0.073967	0.209523	
2	0.515400	0.061381	0.304797	0.059932	0.058337	0.085017	
3	0.514705	0.062566	0.300753	0.062229	0.059278	0.083219	

	labels	wine_labels
0	3.629364	1.137357

```
1  3.655311     1.177145
2  3.268884     1.250846
3  3.283983     1.266596

[4 rows x 30 columns]
```

[143]:
```
df_parallel = df_full.groupby(['merged_labels']).mean()

fig, ax = plt.subplots(figsize=(15,7))
pd.plotting.parallel_coordinates(
    df_parallel[value_feats_mm+wine_feats_dec].reset_index(), 'merged_labels',
    colormap='viridis', ax=ax
)

ax.legend(bbox_to_anchor=(.5,1,.5,1), loc="lower left", frameon=False,
          mode='expand', borderaxespad=0, ncol=4,
          )
ax.yaxis.grid(which="major", linewidth=0)

plt.title('Parallel Plots of Cluster Means', loc='left', fontsize=17)

if SAVE_PLOTS:
    save_fig('Parallel Plots of Cluster Means', fig)
plt.show()
```



5 Classify and predict noise and outlier rows

[144]:
```
Xv = df_full.loc[~df_full['merged_labels'].isna(),value_features]
yv = df_full.loc[~df_full['merged_labels'].isna(),'value_labels']
```

[145]:
```
X_train, X_test, y_train, y_test = train_test_split(
    Xv, yv, test_size=0.2, random_state=RANDOM_STATE
)

# Fitting the decision tree
dtv = DecisionTreeClassifier(random_state=RANDOM_STATE, max_depth=5)
dtv.fit(X_train, y_train)

print("On average, we are able to predict an estimated {0:.2f}% of the␣
 ↪customers' Value labels correctly".format(dtv.score(X_test, y_test)*100))
```

```
On average, we are able to predict an estimated 93.88% of the customers' Value
labels correctly
```

[146]:
```
pd.DataFrame(dtv.feature_importances_, index=X_train.columns).sort_values(by=0,␣
 ↪ascending=False)
```

[146]:

	0
Freq	0.563614
Dayswus	0.368439
LTV	0.042326
Recency	0.025620

[147]:
```
# Predicting the cluster labels of the noise and outliers

df_full.loc[df_full['merged_labels'].isna(),'value_labels'] = dtv.
 ↪predict(df_full.loc[df_full['merged_labels'].isna(),value_features])
```

[148]:
```
Xw = df_full.loc[~df_full['merged_labels'].isna(),wine_features]
yw = df_full.loc[~df_full['merged_labels'].isna(),'wine_labels']
```

[149]:
```
Xw_train, Xw_test, yw_train, yw_test = train_test_split(
    Xw, yw, test_size=0.2, random_state=RANDOM_STATE
)

# Fitting the decision tree
dtw = DecisionTreeClassifier(random_state=RANDOM_STATE, max_depth=5)
dtw.fit(Xw_train, yw_train)

print("On average, we are able to predict an estimated {0:.2f}% of the␣
 ↪customers' Wine labels correctly".format(dtw.score(Xw_test, yw_test)*100))
```

```
On average, we are able to predict an estimated 88.31% of the customers' Wine
labels correctly
```

[150]:
```
pd.DataFrame(dtw.feature_importances_, index=Xw_train.columns).
↪sort_values(by=0, ascending=False)
```

[150]:

	0
Dryred	0.639514
Exotic	0.231822
Drywh	0.128664
Sweetred	0.000000
Sweetwh	0.000000
Dessert	0.000000

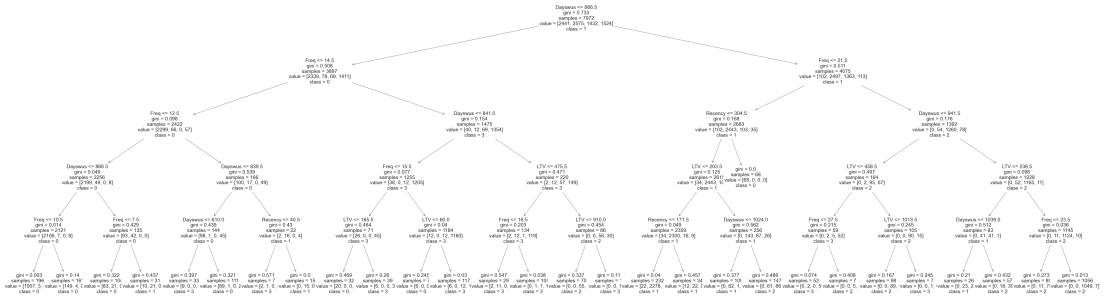
[151]:
```
df_full.loc[df_full['merged_labels'].isna(),'wine_labels'] = dtw.
↪predict(df_full.loc[df_full['merged_labels'].isna(),wine_features])
```

[152]:
```
val_labels_list
```

[152]: [0, 1, 2, 3]

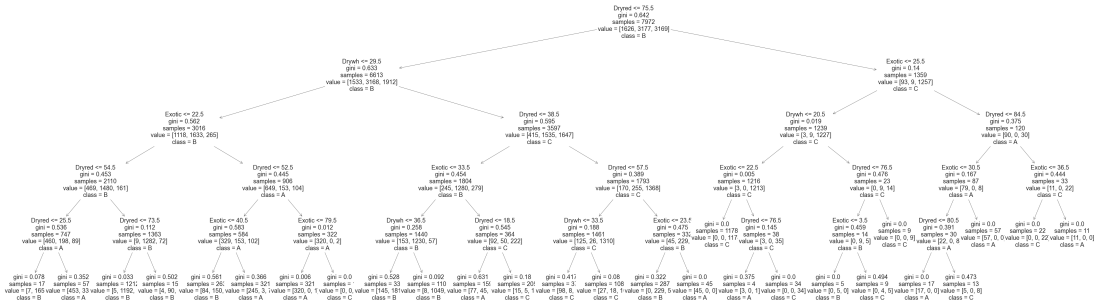
6 Visualize Decision Tree

[153]:
```
fig, axes = plt.subplots(1,1,figsize = (80,24))

tree.plot_tree(dtv,
               feature_names = Xv.columns.to_list(),
               class_names=['0','1','2','3'],
               filled = False,
               fontsize=20, ax=axes);
plt.show()
```



[154]:
```
fig, axes = plt.subplots(1,1,figsize = (80,24))

tree.plot_tree(dtw,
               feature_names = Xw.columns.to_list(),
               class_names=['A','B','C'],
```

```
                filled = False,
                fontsize=24, ax=axes);
plt.show()
```



7 Supporting Visualizations

[155]:
```
## Code based from sklearn documentation:
## https://scikit-learn.org/stable/auto_examples/cluster/
 ↪plot_kmeans_silhouette_analysis.html?highlight=silhouette

def plotSilhouettes(X, max_clust, plot_dim, title = "Silhouette analysis for␣
 ↪KMeans clustering with different K sizes"):
    range_n_clusters = range(2, max_clust+1)

    fig, axes = plt.subplots(plot_dim[0], plot_dim[1], sharex=True,␣
 ↪figsize=(19,11))
    #fig.set_size_inches(11, 11)

    for ax, nclust in zip(axes.flatten(), range_n_clusters):
        # The (n_clusters+1)*10 is for inserting blank space between silhouette
        # plots of individual clusters, to demarcate them clearly.
        ax.set_ylim([0, len(X) + (nclust + 1) * 10])

        # Initialize the clusterer with n_clusters value and a random generator
        # seed of 10 for reproducibility.
        clusterer = KMeans(n_clusters=nclust, random_state=10)
        cluster_labels = clusterer.fit_predict(X)

        # The silhouette_score gives the average value for all the samples.
        # This gives a perspective into the density and separation of the formed
        silhouette_avg = silhouette_score(X, cluster_labels)

        # Compute the silhouette scores for each sample
```

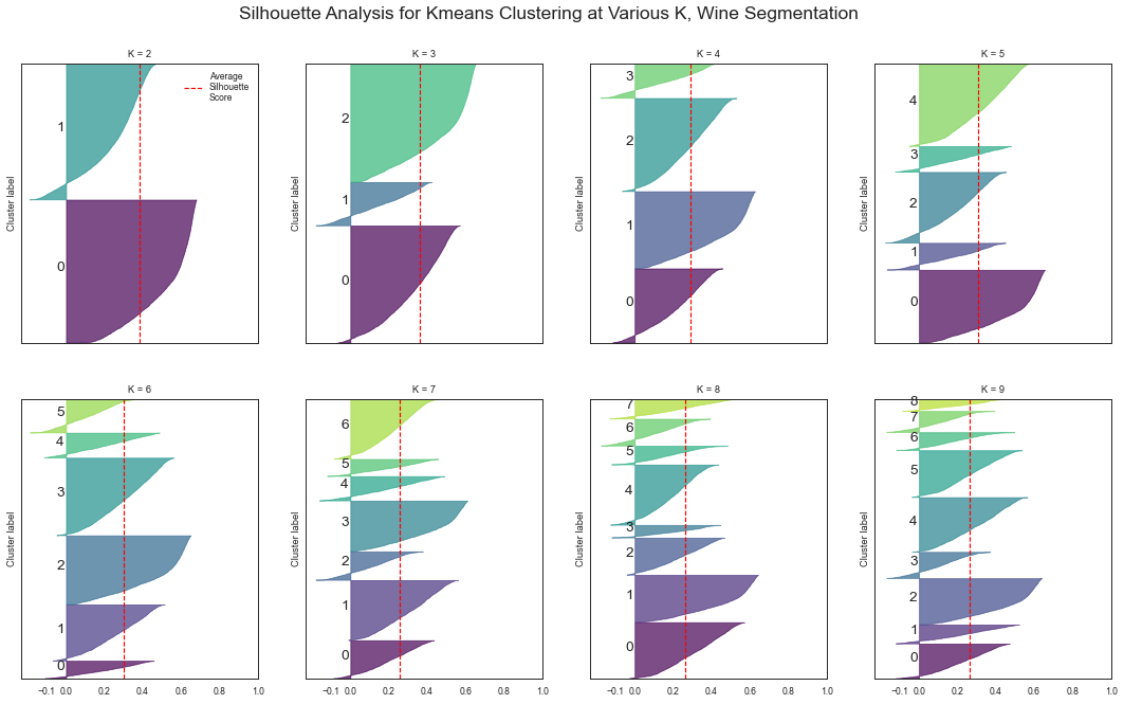
```
        sample_silhouette_values = silhouette_samples(X, cluster_labels)

        y_lower = 10
        for i in range(nclust):
            # Aggregate the silhouette scores for samples belonging to
            # cluster i, and sort them
            ith_cluster_silhouette_values =␣
↪sample_silhouette_values[cluster_labels == i]

            ith_cluster_silhouette_values.sort()

            size_cluster_i = ith_cluster_silhouette_values.shape[0]
            y_upper = y_lower + size_cluster_i

            color = cm.viridis(float(i) / nclust)
            #color = COLORS[i]
            ax.fill_betweenx(
                np.arange(y_lower, y_upper),
                0,
                ith_cluster_silhouette_values,
                facecolor=color,
                edgecolor=color,
                alpha=0.7,
            )

            # Label the silhouette plots with their cluster numbers at the␣
↪middle
            ax.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i), fontsize=14)

        # Compute the new y_lower for next plot
            y_lower = y_upper + 10  # 10 for the 0 samples


        ax.set_title("K = "+ str(nclust))
        ax.set_ylabel("Cluster label")

        ax.set_yticks([])  # Clear the yaxis labels / ticks
        ax.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

        # The vertical line for average silhouette score of all the values
        ax.axvline(x=silhouette_avg, color="red", linestyle="--",␣
↪label='Average\nSilhouette\nScore')
        if nclust == 2 :
            ax.legend(frameon=False)

        plt.suptitle(
            title,
```

```
            fontsize=18,
            y=.95
        )
    save_fig(title, fig)
    plt.show()
```

7.1 Ideal cluster sizes: silhouette plots

7.1.1 Wine Segmentation

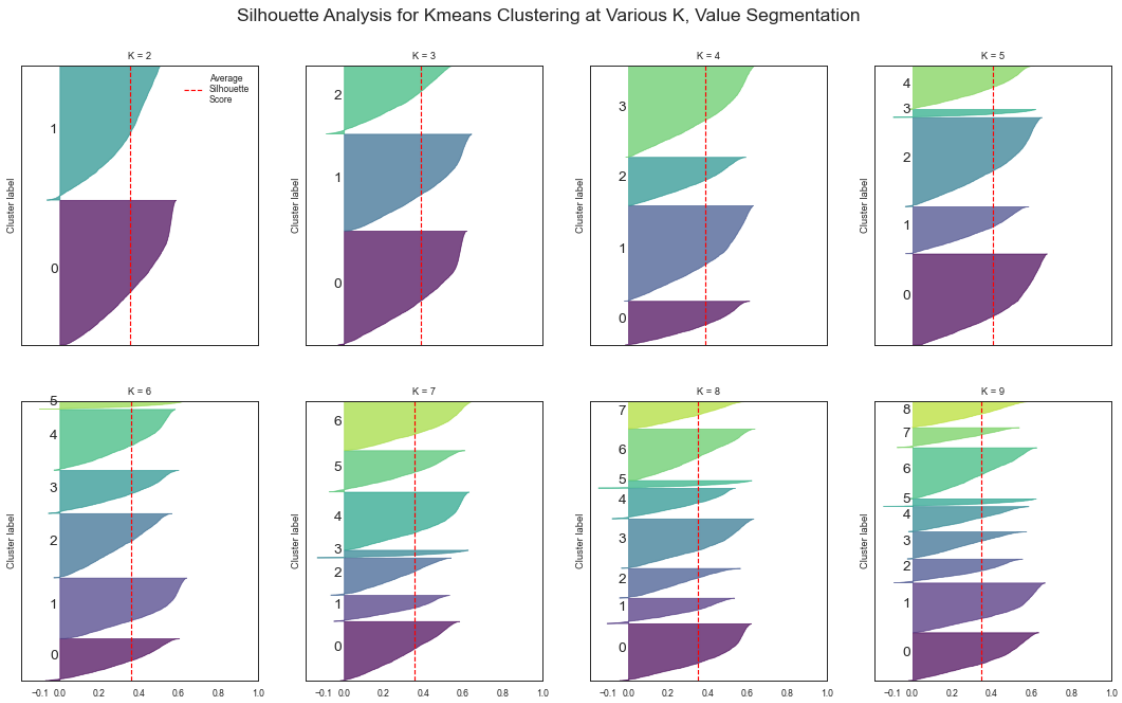
[156]:
```
plotSilhouettes(df_wine_kmeans[wine_feats_dec], 9, [2,4], title='Silhouette␣
↪Analysis for Kmeans Clustering at Various K, Wine Segmentation')
```



[ ]:
```

```

7.1.2 Value Segmentation

[157]:
```
plotSilhouettes(df_value_kmeans[value_feats_mm], 9, [2,4], title='Silhouette␣
↪Analysis for Kmeans Clustering at Various K, Value Segmentation')
```



[158]:
```
df_
```

[158]:

Custid	Dayswus_mm	Freq_mm	Recency_mm	LTV_mm	value_labels	Dryred_dec	\
5325	0.147143	0.358491	0.032787	0.348824	3	0.67	
3956	0.701429	0.660377	0.060109	0.401456	2	0.49	
3681	0.165714	0.056604	0.102004	0.095745	0	0.04	
2829	0.712857	0.018868	0.083789	0.096305	1	0.86	
8788	0.410000	0.018868	0.005464	0.101904	0	0.85	
…	…	…	…	…	…	…	
1383	0.831429	0.339623	0.107468	0.204367	1	0.78	
4070	0.065714	0.320755	0.081967	0.318589	3	0.30	
7909	0.098571	0.037736	0.118397	0.102464	0	0.06	
4158	0.795714	0.000000	0.670310	0.100784	0	0.18	
4914	0.612857	0.452830	0.051002	0.263718	2	0.63	

Custid	Sweetred_dec	Drywh_dec	Sweetwh_dec	Dessert_dec	Exotic_dec	labels	\
5325	0.04	0.26	0.02	0.01	0.01	0	
3956	0.00	0.46	0.01	0.03	0.00	4	
3681	0.29	0.14	0.32	0.21	0.48	6	
2829	0.01	0.11	0.01	0.01	0.55	7	
8788	0.00	0.12	0.02	0.01	0.28	5	
…	…	…	…	…	…	…	

```
1383            0.00       0.20         0.01         0.01        0.11       5
4070            0.12       0.36         0.10         0.12        0.13       2
7909            0.24       0.10         0.38         0.22        0.41       6
4158            0.13       0.45         0.11         0.13        0.13       2
4914            0.10       0.13         0.11         0.03        0.04       0

        wine_labels wine_letters merged_labels
Custid
5325              1            B            3B
3956              2            C            2C
3681              0            A            0A
2829              0            A            1A
8788              2            C            0C
…               …            …            …
1383              2            C            1C
4070              1            B            3B
7909              0            A            0A
4158              1            B            0B
4914              1            B            2B

[9966 rows x 15 columns]
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

8 Other things we tried

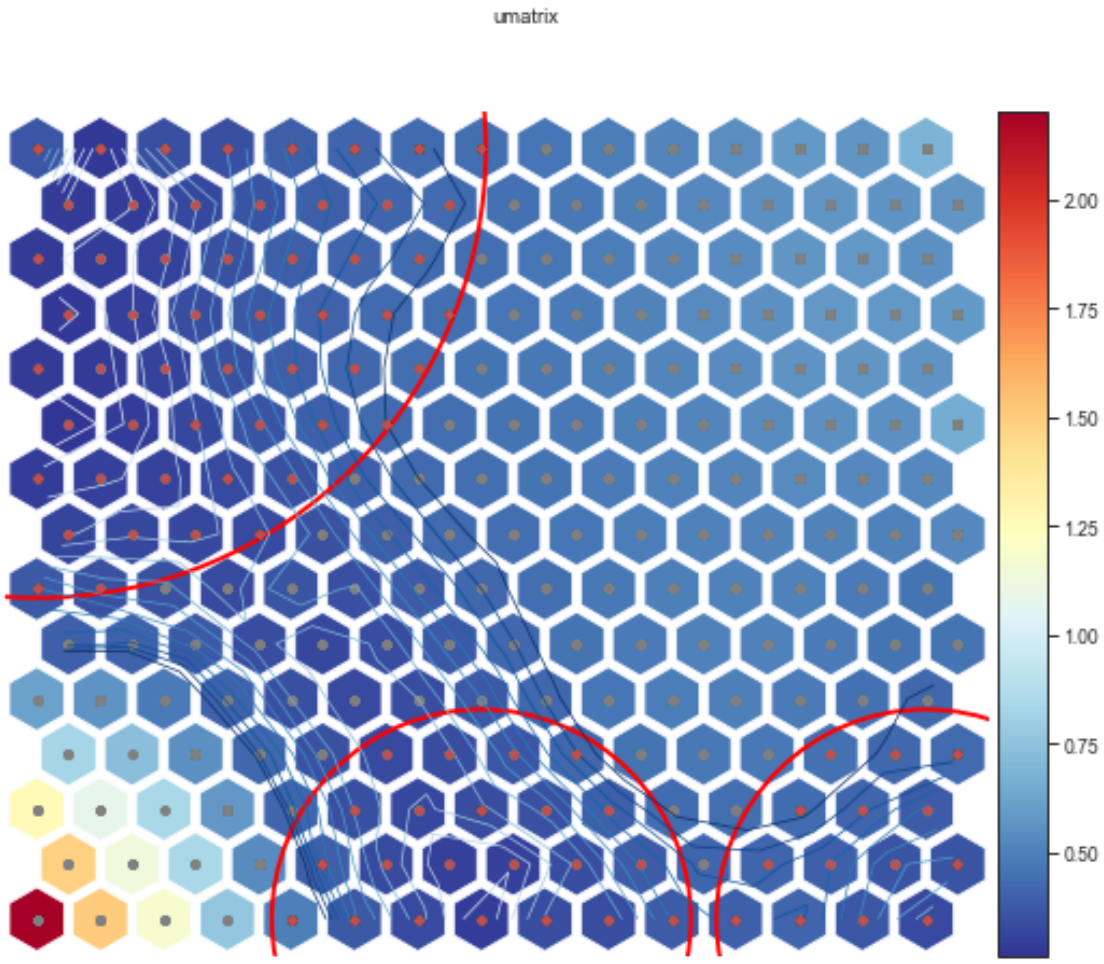
8.1 Self Organizing Maps

8.1.1 Value

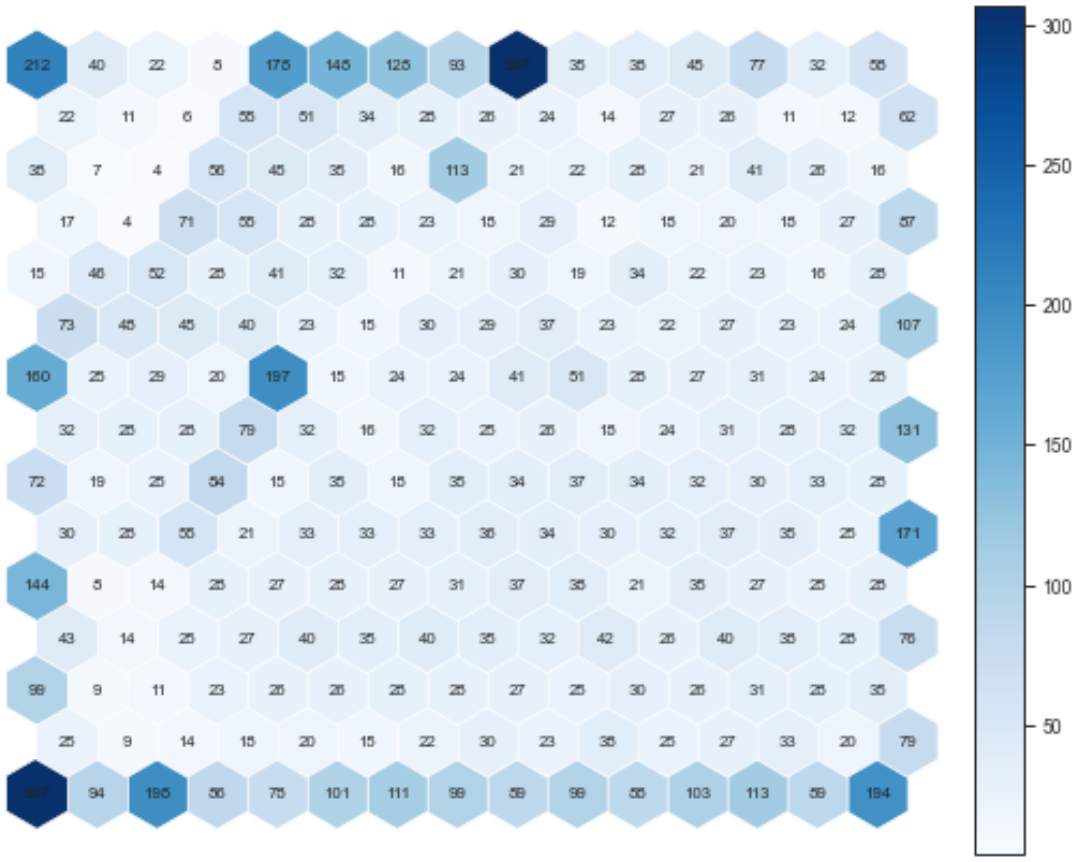
[159]:
```
som_vars = value_feats_mm
df_som = df_nonoise.copy()

## From Lab 11

np.random.seed(RANDOM_STATE)

smv = sompy.SOMFactory().build(
    df_som[som_vars].values,
    mapsize=[15,15],
```

```
    initialization='random',
    neighborhood='gaussian',
    training='batch',
    lattice='hexa',
    component_names=som_vars
)
smv.train(n_job=4, train_rough_len=50, train_finetune_len=50)
```

[160]:
```
u = sompy.umatrix.UMatrixView(8, 8, 'umatrix', show_axis=True, text_size=8,␣
 ↪show_text=True)

UMAT = u.show(
    smv,
    distance=4,
    row_normalized=False,
    show_data=True,
    contour=True, # Visualize isomorphic curves
    blob=True
)
```



[161]:
```
vhts  = BmuHitsView(9,9,"Hits Map")
vhts.show(smv, anotate=True, onlyzeros=False, labelsize=8, cmap="Blues")
plt.show()
```



[162]:
```
def cluster_on_som(sm, k, df_, som_vars):
    df = df_.copy()
    kmeans = KMeans(n_clusters=k, init='k-means++', n_init=20,␣
 ↪random_state=random_state)
    nodeclus_labels = kmeans.fit_predict(sm.codebook.matrix)
    sm.cluster_labels = nodeclus_labels

    # Check the nodes and and respective clusters
    nodes = sm.codebook.matrix

    somlabelk = 'som_label_'+str(k)

    df_nodes = pd.DataFrame(nodes, columns=som_vars)
    df_nodes[somlabelk] = nodeclus_labels
```

```
    # Obtaining SOM's BMUs labels
    bmus_map = sm.find_bmu(df[som_vars])[0]  # get bmus for each observation in␣
↪df
    bmuk = 'BMU'+str(k)
    df[bmuk] = bmus_map
    df_bmus = df.copy()

    # Get cluster labels for each observation
    df = df_bmus.merge(df_nodes[somlabelk], 'left', left_on=bmuk,␣
↪right_index=True)

    s_ = df.groupby([somlabelk]).count().iloc[:,0].tolist()
    r_ = r2_(df[som_vars + [somlabelk]],somlabelk)

    return ({'k':k,'r':r_,'sizes':s_}, df[somlabelk])
```

[163]:
```
kval = 10
som_k_sizes_v = pd.DataFrame(columns=['k','r','sizes'])

for k in range(2,kval+2):
    cluster_on_som_res_d = cluster_on_som(smv,k, df_som, som_vars)
    som_k_sizes_v = som_k_sizes_v.
↪append(cluster_on_som_res_d[0],ignore_index=True)

som_k_sizes_v
```

[163]:

	k	r	sizes
0	2	0.708591	[4863, 5103]
1	3	0.883568	[6684, 274, 3008]
2	4	0.883568	[3008, 274, 6684]
3	5	0.876198	[1939, 1390, 6637]
4	6	0.801357	[9691, 1, 274]
5	7	0.198242	[4, 274, 9662, 26]
6	8	0.872431	[9444, 274, 248]
7	9	0.801305	[274, 9691, 1]
8	10	0.975582	[7417, 2275, 274]
9	11	0.987858	[1748, 2992, 5226]

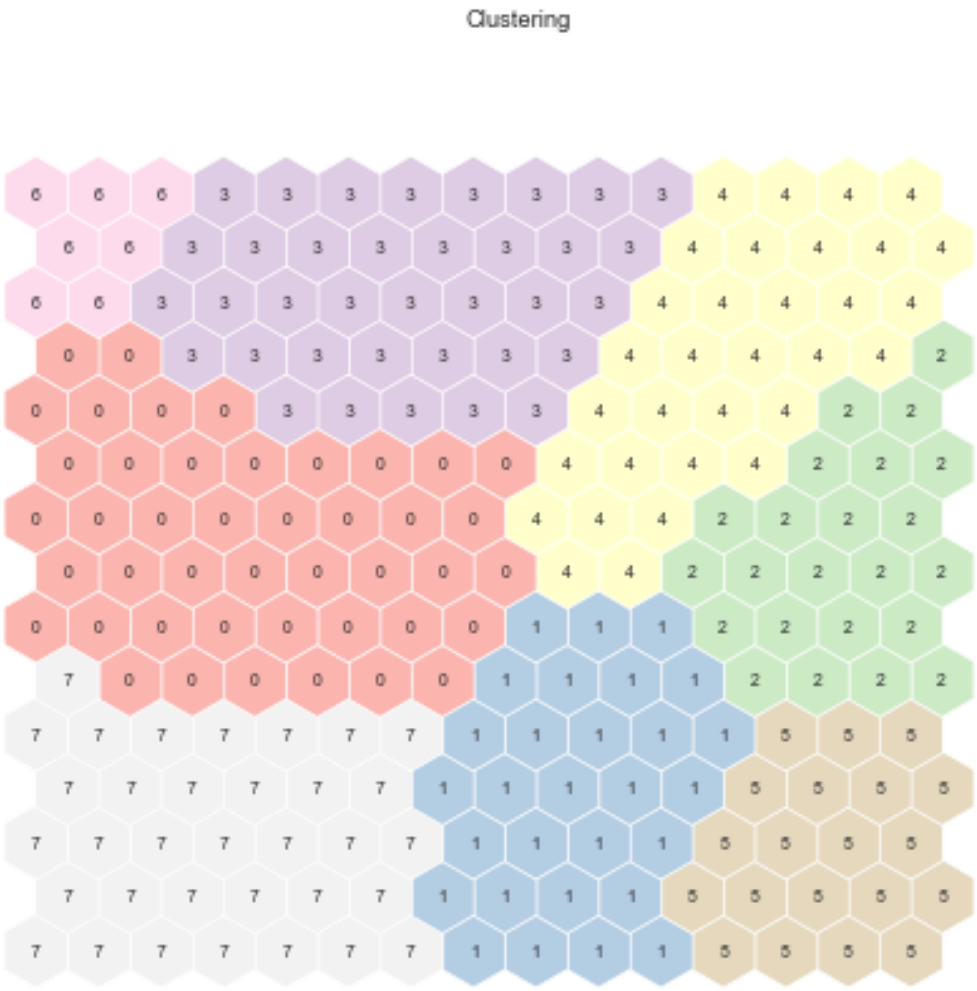
[164]:
```
som_k_sizes_v.sort_values(by=['r'], ascending=False)
```

[164]:

	k	r	sizes
9	11	0.987858	[1748, 2992, 5226]
8	10	0.975582	[7417, 2275, 274]
1	3	0.883568	[6684, 274, 3008]
2	4	0.883568	[3008, 274, 6684]
3	5	0.876198	[1939, 1390, 6637]

```
6   8  0.872431    [9444, 274, 248]
4   6  0.801357      [9691, 1, 274]
7   9  0.801305      [274, 9691, 1]
0   2  0.708591        [4863, 5103]
5   7  0.198242  [4, 274, 9662, 26]
```

[165]:
```
k = 8
cluster_on_som_res = cluster_on_som(smv,k, df_som, som_vars)
df_som['som_value']  = cluster_on_som_res[1]

# Perform K-Means clustering with selected K
kmeans = KMeans(n_clusters=k, init='k-means++', n_init=20,␣
 ↪random_state=random_state)
nodeclus_labels = kmeans.fit_predict(smv.codebook.matrix)
smv.cluster_labels = nodeclus_labels  # setting the cluster labels of sompy

hits = HitMapView(8, 8,"Clustering", text_size=10)
hits.show(smv, anotate=True, onlyzeros=False, labelsize=7, cmap="Pastel1")

plt.show()
```

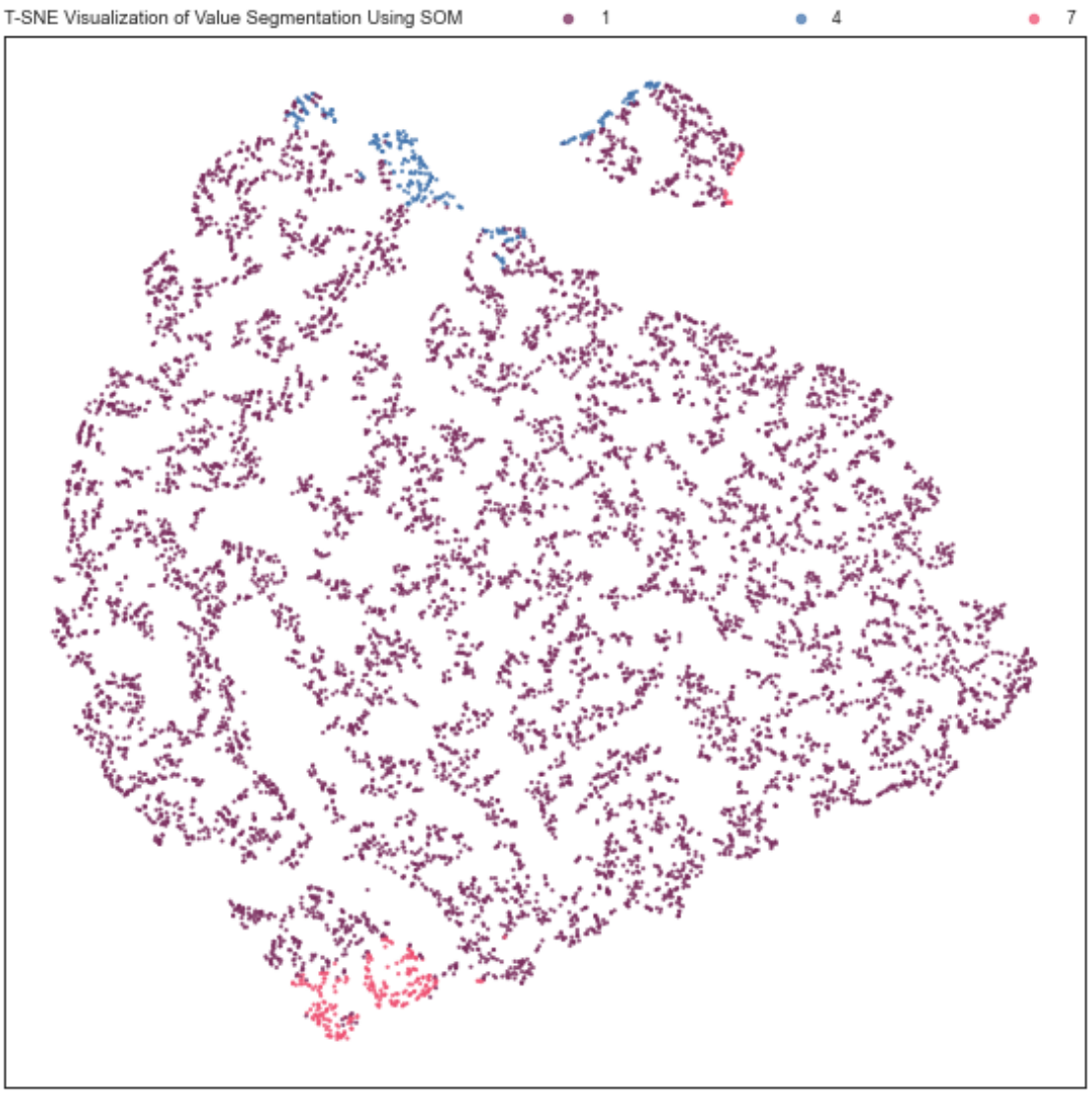


Visualize

[ ]:

[166]:
```
tsnemodel_somv = TSNE(random_state=RANDOM_STATE).
↪fit_transform(df_som[value_feats_mm])

tsne_value_somv = pd.DataFrame(tsnemodel_somv)
```

[167]:
```
plot_clusters(tsne_value_somv, df_som['som_value'], 'T-SNE Visualization of␣
↪Value Segmentation Using SOM')
```

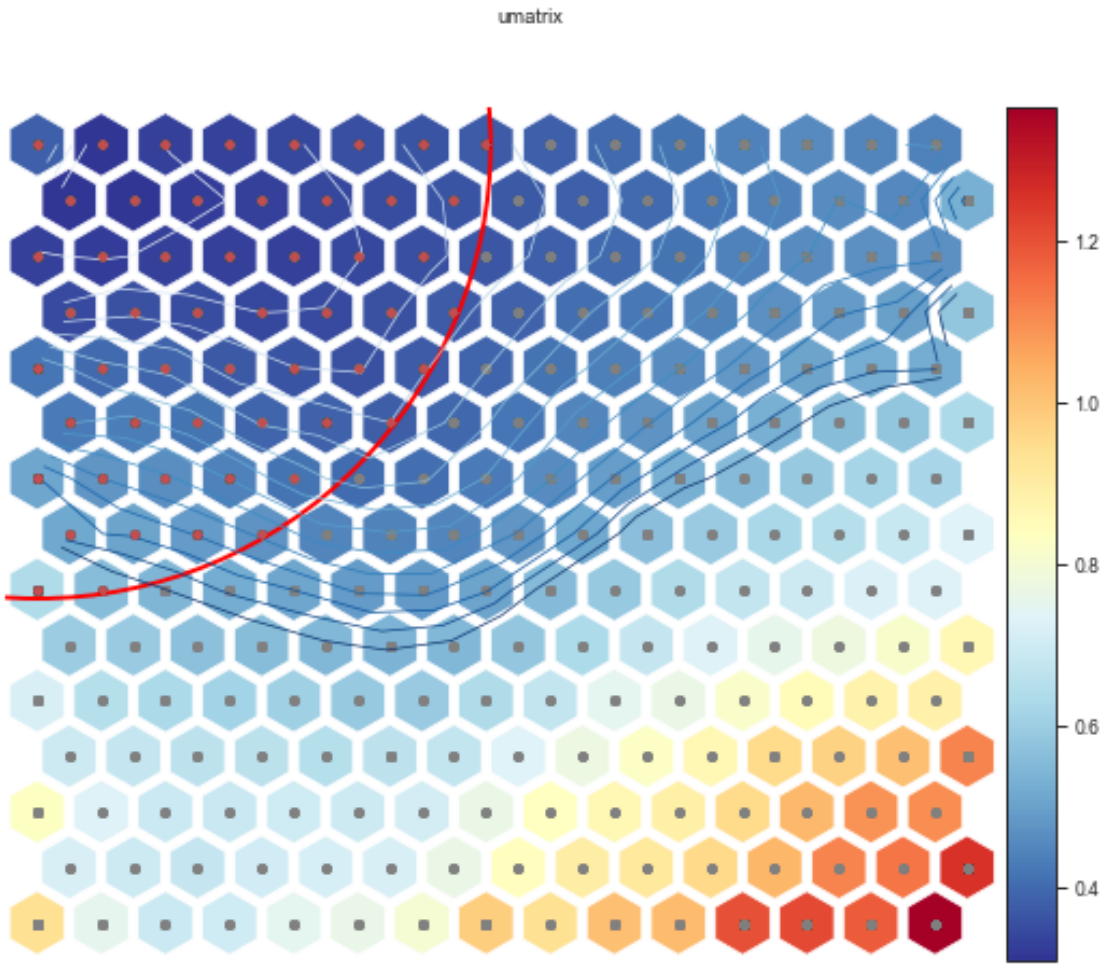


8.1.2 Wine

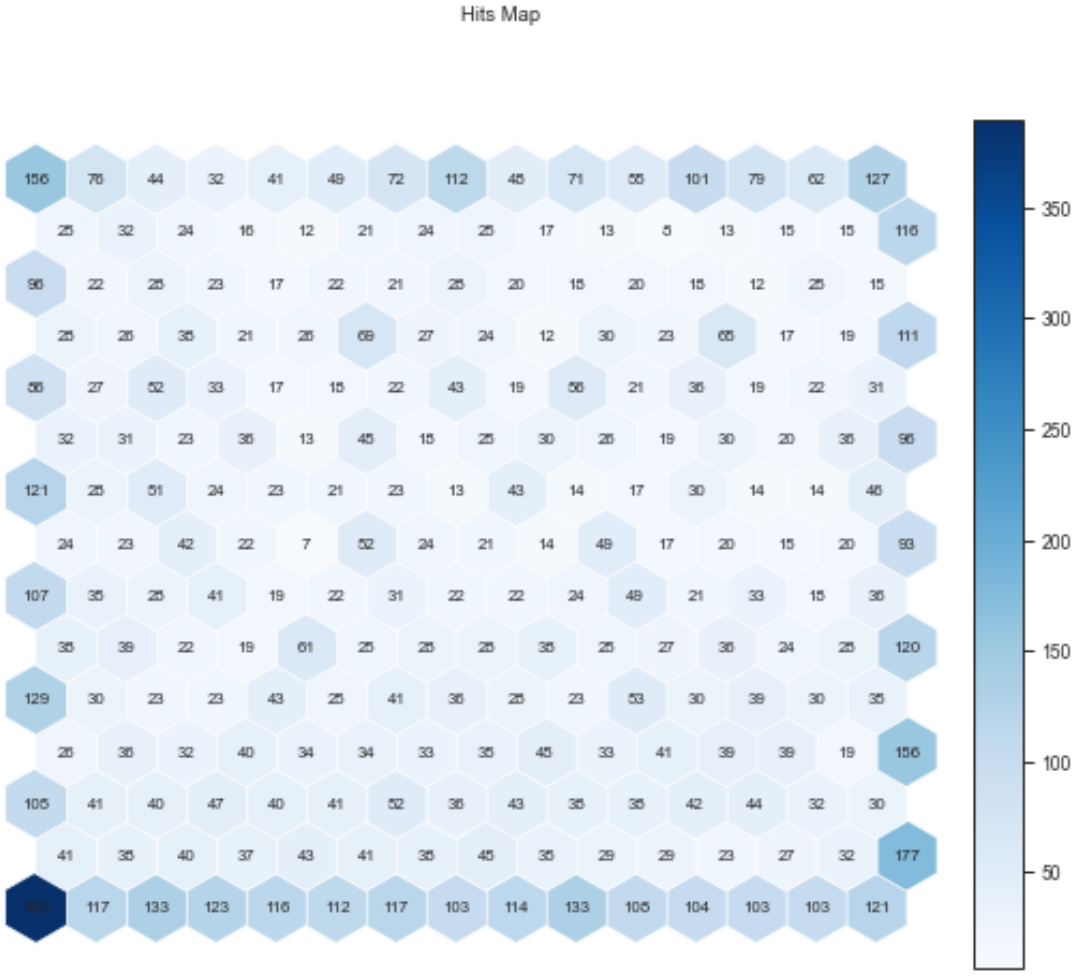
[168]:
```
df_wine = df[wine_feats_dec].copy()

np.random.seed(RANDOM_STATE)
som_vars_w = wine_feats_dec

# Training the SOM
smw = sompy.SOMFactory().build(
    df_wine.values,
    mapsize=[15, 15],
    initialization='random',
```

```
    neighborhood='gaussian',
    training='batch',
    lattice='hexa',
    component_names=wine_feats_dec
)
smw.train(n_job=-1, verbose='info', train_rough_len=100, train_finetune_len=100)
```

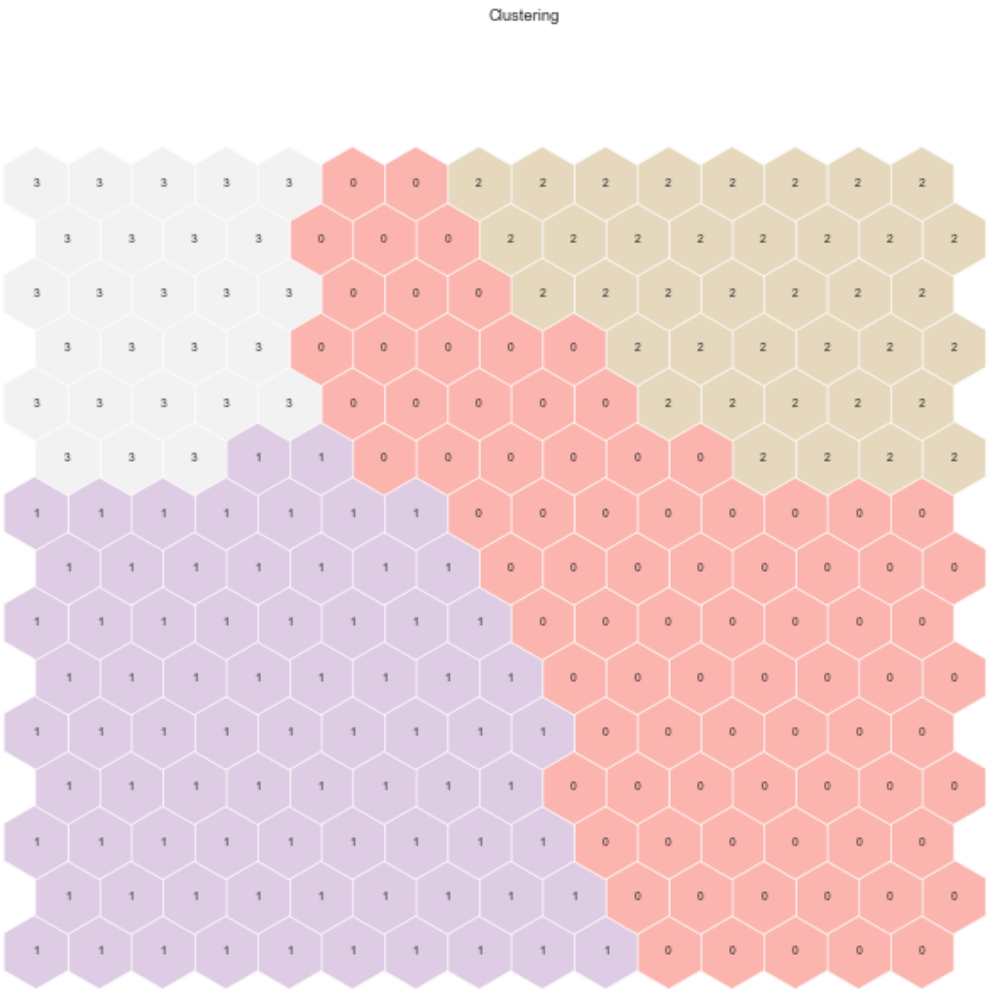
[169]:
```
u = sompy.umatrix.UMatrixView(8, 8, 'umatrix', show_axis=True, text_size=8,␣
↪show_text=True)

UMAT = u.show(
    smw,
    distance=4,
    row_normalized=False,
    show_data=True,
    contour=True, # Visualize isomorphic curves
    blob=True
)
```



[170]:
```
whts  = BmuHitsView(9,9,"Hits Map")
whts.show(smw, anotate=True, onlyzeros=False, labelsize=8, cmap="Blues")
plt.show()
```



[171]:
```
kmeans_somw = KMeans(n_clusters=4, init='k-means++', n_init=20,␣
↪random_state=RANDOM_STATE)
nodeclus_labels = kmeans_somw.fit_predict(smw.codebook.matrix)
smw.cluster_labels = nodeclus_labels  # setting the cluster labels of sompy

hits = HitMapView(12, 12,"Clustering", text_size=10)
hits.show(smw, anotate=True, onlyzeros=False, labelsize=7, cmap="Pastel1")

plt.show()
```



[172]:
```
# Check the nodes and and respective clusters
wine_som_nodes = smw.codebook.matrix

df_wine_som_nodes = pd.DataFrame(wine_som_nodes, columns=wine_feats_dec)
df_wine_som_nodes['wine_nodes'] = nodeclus_labels
#df_wine_som_nodes
```

[173]:
```
# Obtaining SOM's BMUs labels
bmus_map = smw.find_bmu(df_wine)[0]  # get bmus for each observation in df

df_bmus = pd.DataFrame(
    np.concatenate((df, np.expand_dims(bmus_map,1)), axis=1),
    index=df.index, columns=np.append(df.columns,"BMU")
```

```
)
#df_bmus
```

[174]:
```
# Get cluster labels for each observation
dfw_ = df_bmus.merge(df_wine_som_nodes['wine_nodes'], 'left', left_on="BMU",␣
↪right_index=True)

# Characterizing the final clusters
dfw_.drop(columns='BMU').groupby('wine_nodes').mean()
```

[174]:

wine_nodes	Dayswus	Age	Edu	Income	Freq	\
0	897.411321	43.701887	15.173585	63919.154717	14.773585	
1	897.206771	50.997452	17.702949	74437.594467	14.965417	
3	898.398332	46.971527	16.429681	68479.202905	14.533218	

wine_nodes	Recency	Monetary	LTV	Perdeal	Dryred	…	\
						…	
0	69.569811	656.471698	250.381132	31.316981	14.837736	…	
1	56.622861	613.264652	176.955224	30.573717	78.640335	…	
3	62.915301	626.758269	220.619212	33.087863	40.746764	…	

wine_nodes	Dryred_dec	Sweetred_dec	Drywh_dec	Sweetwh_dec	Dessert_dec	\
0	0.148377	0.177057	0.312528	0.186000	0.176226	
1	0.786403	0.019385	0.155952	0.019752	0.017856	
3	0.407468	0.086123	0.335459	0.085820	0.085037	

wine_nodes	Exotic_dec	Recency_Rank	Frequency_Rank	Monetary_Rank	RFM_Ave
0	0.103472	2.849057	2.656604	2.713208	2.739623
1	0.088970	2.941755	2.986167	3.071351	2.999757
3	0.196192	2.867990	2.710670	2.796664	2.791775

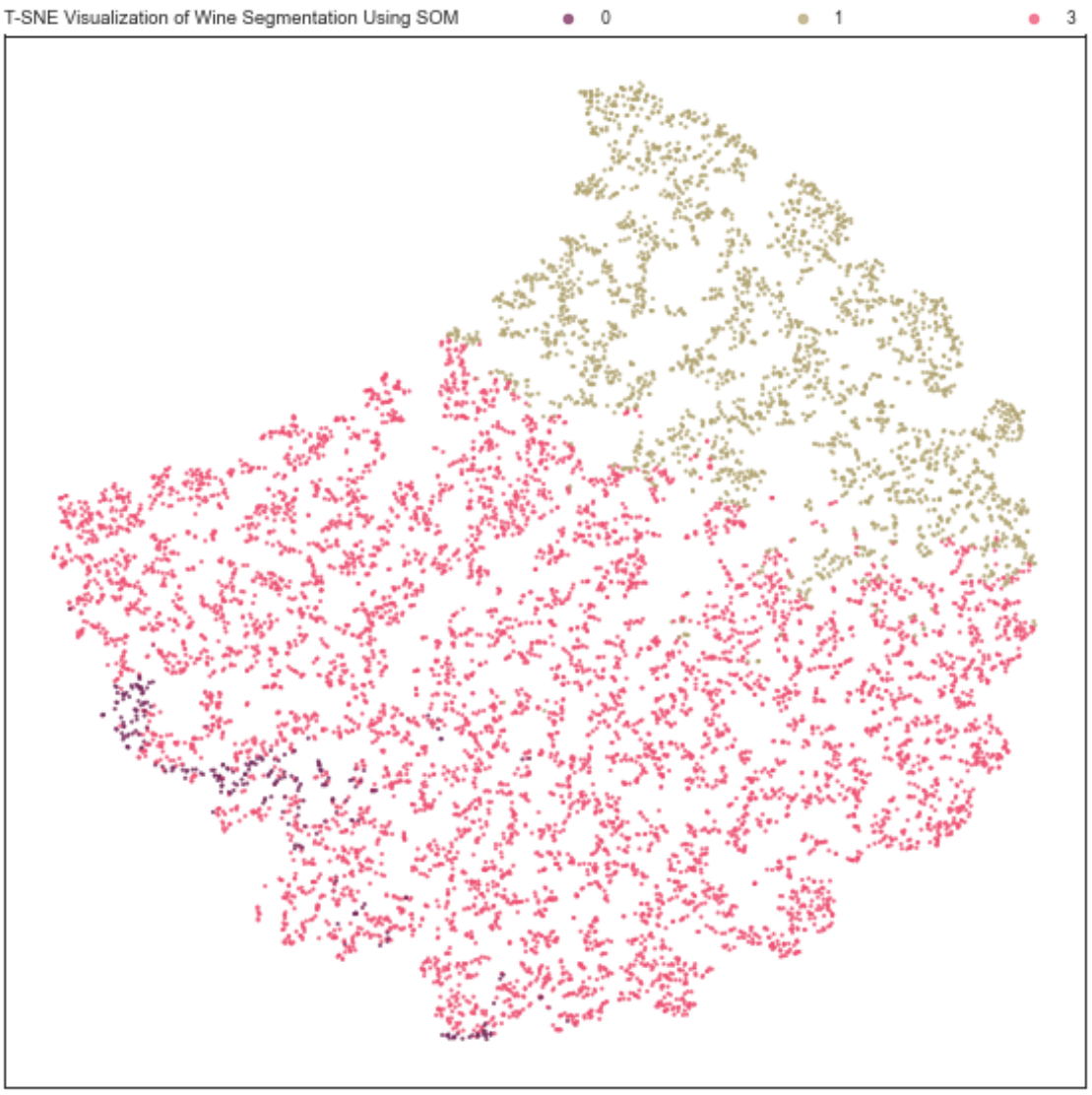
[3 rows x 32 columns]

[175]:
```
dfw_.drop('BMU', axis=1, inplace=True)
```

Visualize

[176]:
```
#tsnemodel_somw = TSNE(random_state=42).fit_transform(dfw_[wine_feats_dec])
tsnemodel_somw = TSNE(random_state=RANDOM_STATE).
↪fit_transform(df_som[wine_feats_dec])

tsne_value_somw = pd.DataFrame(tsnemodel_somw)
```

[177]:
```
plot_clusters(tsne_value_somw, dfw_['wine_nodes'], 'T-SNE Visualization of Wine␣
↪Segmentation Using SOM')
```



[ ]:

[ ]:

8.2 Mean Shift Clustering

8.2.1 Value

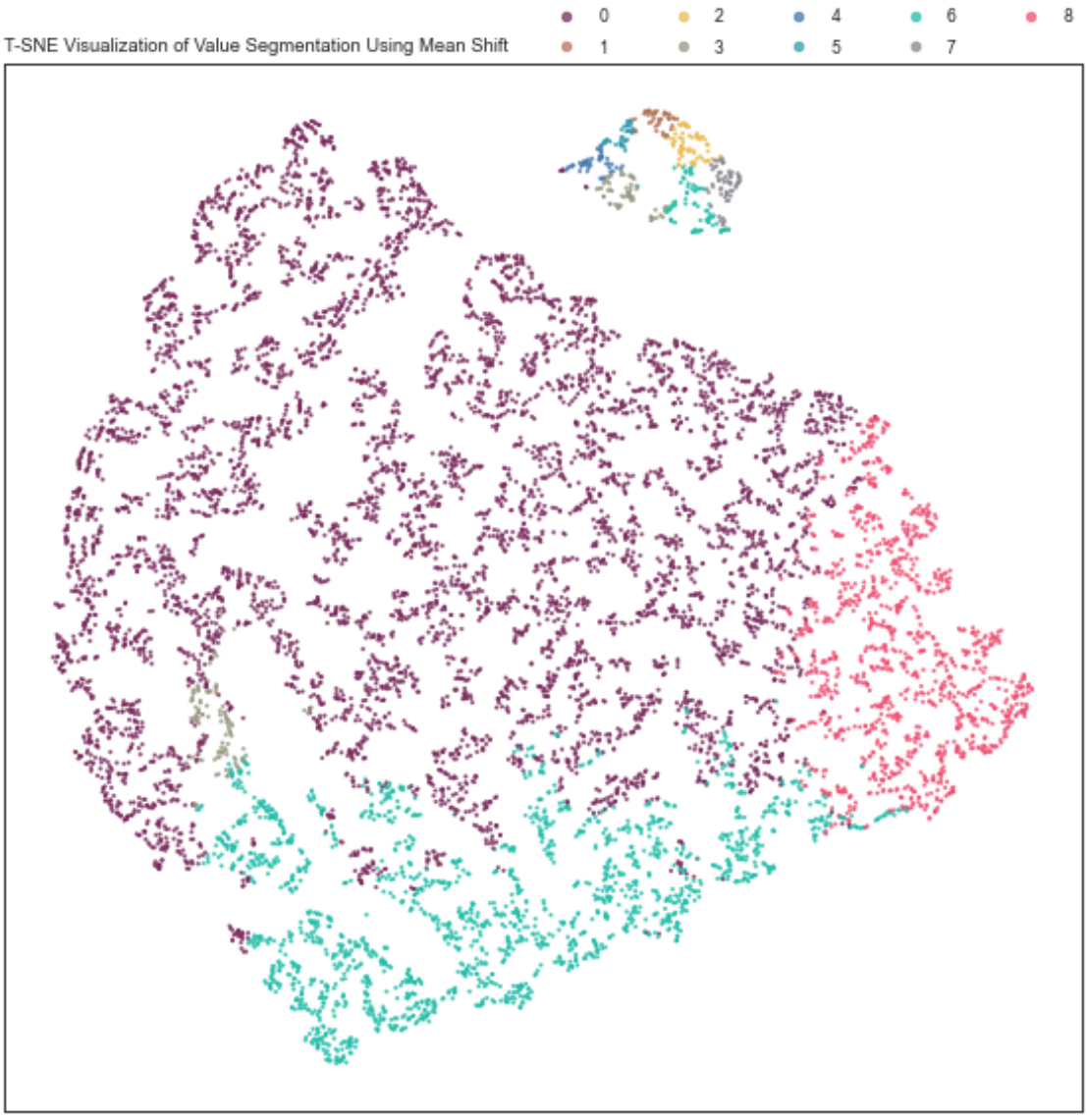
[178]:
```
df_msv = df_nonoise[value_feats_mm].copy()

bandwidth = estimate_bandwidth(df_msv, quantile=.05, random_state=RANDOM_STATE,␣
↪n_jobs=-1)
bandwidth
```

[178]: 0.1717970434977587

[179]:
```
msv = MeanShift(bandwidth=bandwidth, bin_seeding=True, n_jobs=4)
ms_labels = msv.fit_predict(df_msv)

ms_n_clusters = len(np.unique(ms_labels))
print("Number of estimated clusters : %d" % ms_n_clusters)
df_msv['ms_value'] = ms_labels
```

```
Number of estimated clusters : 9
```

[180]:
```
plot_clusters(tsne_value_somv, df_msv['ms_value'], 'T-SNE Visualization of␣
↪Value Segmentation Using Mean Shift')
```



8.2.2 Wine

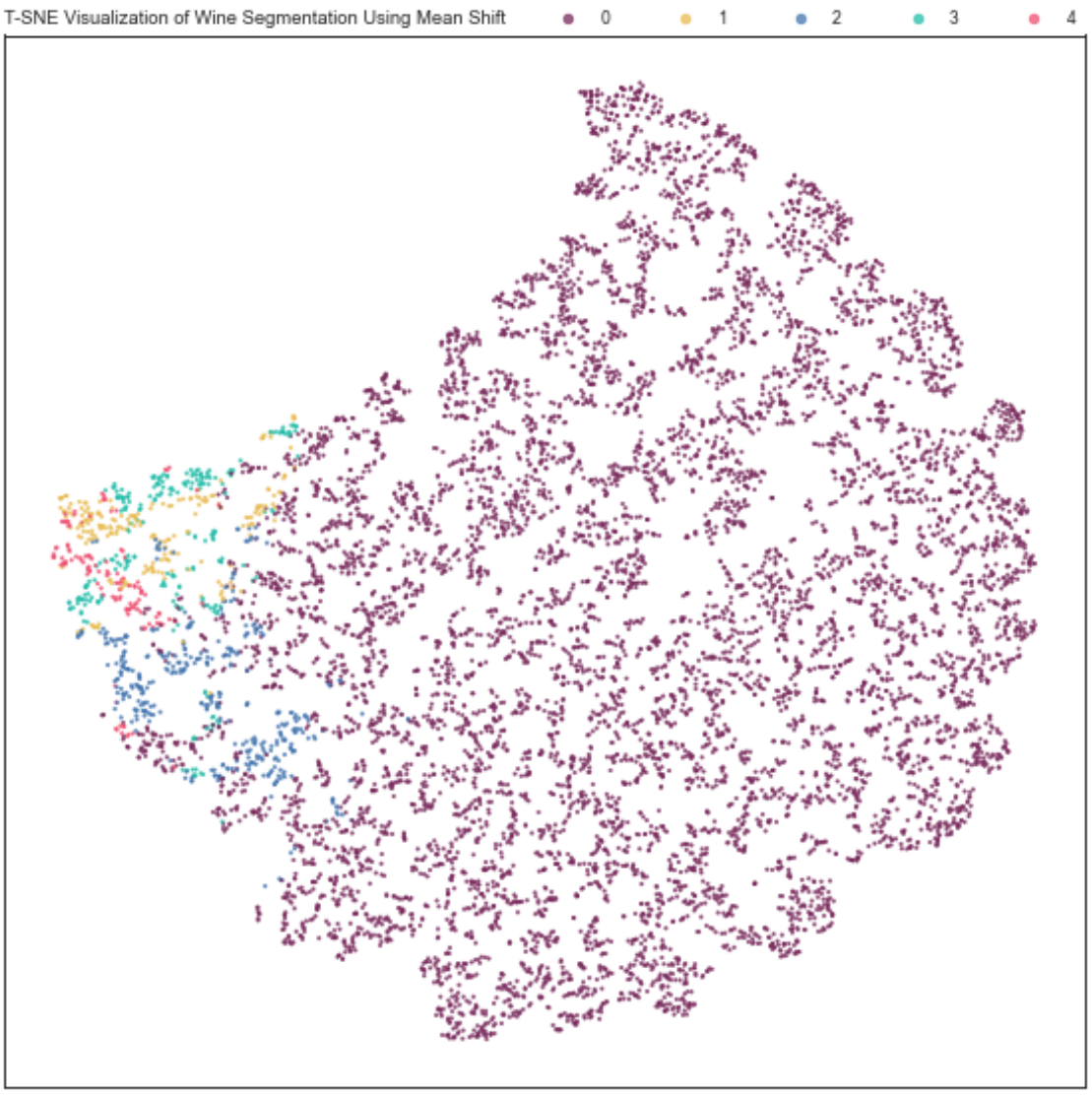
[181]:
```
df_msw = df_nonoise[wine_feats_dec].copy()

bandwidth = estimate_bandwidth(df_msw, quantile=.1, random_state=RANDOM_STATE,␣
↪n_jobs=-1)
bandwidth
```

[181]:
```
0.2029944194592844
```

[182]:
```
msw = MeanShift(bandwidth=bandwidth, bin_seeding=True, n_jobs=4)
msw_labels = msw.fit_predict(df_msw)

ms_n_clusters = len(np.unique(msw_labels))
print("Number of estimated clusters : %d" % ms_n_clusters)
df_msw['ms_wine'] = msw_labels
```

```
Number of estimated clusters : 5
```

[183]:
```
plot_clusters(tsne_value_somw, df_msw['ms_wine'], 'T-SNE Visualization of Wine␣
↪Segmentation Using Mean Shift')
```



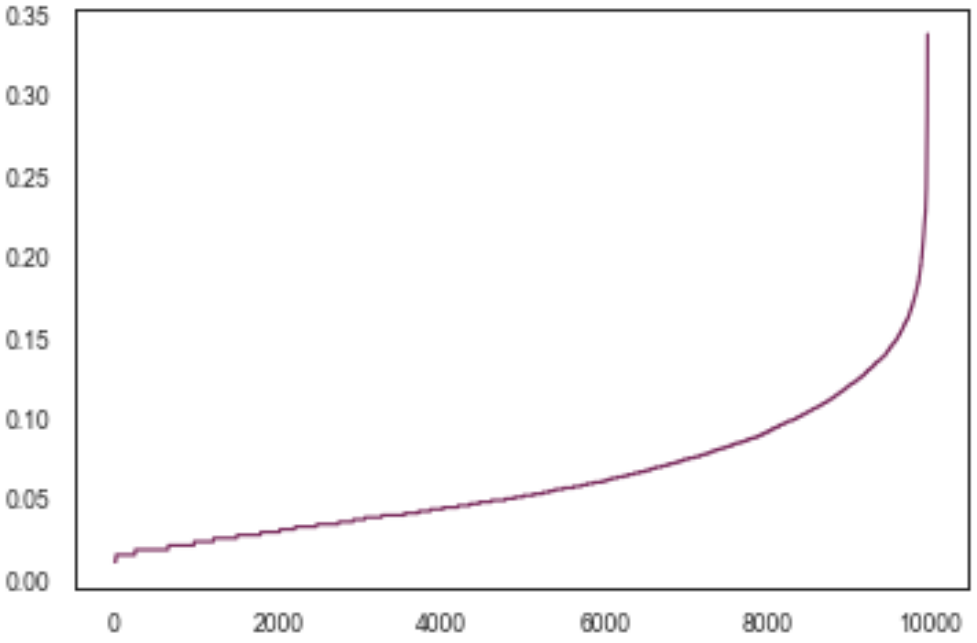
[ ]:

8.3 DBSCAN

8.3.1 Value

[184]:
```
df_dbv = df_nonoise[wine_feats_dec].copy()

# K-distance graph to find out the right eps value
neigh = NearestNeighbors(n_neighbors=10)
neigh.fit(df_dbv)
distances, _ = neigh.kneighbors(df_dbv)
distances = np.sort(distances[:, -1])
```

```
fig = plt.figure()
plt.plot(distances)
save_fig('Epsilon Plot for DBScan', fig)
plt.show()
```



[185]:
```
# Perform DBSCAN clustering
dbscanv = DBSCAN(eps=0.12, min_samples=10, n_jobs=4)
dbscan_labels = dbscanv.fit_predict(df_dbv)

dbscan_n_clusters = len(np.unique(dbscan_labels))
print("Number of estimated clusters : %d" % dbscan_n_clusters)

df_dbv['db_value'] = dbscan_labels
```

```
Number of estimated clusters : 2
```

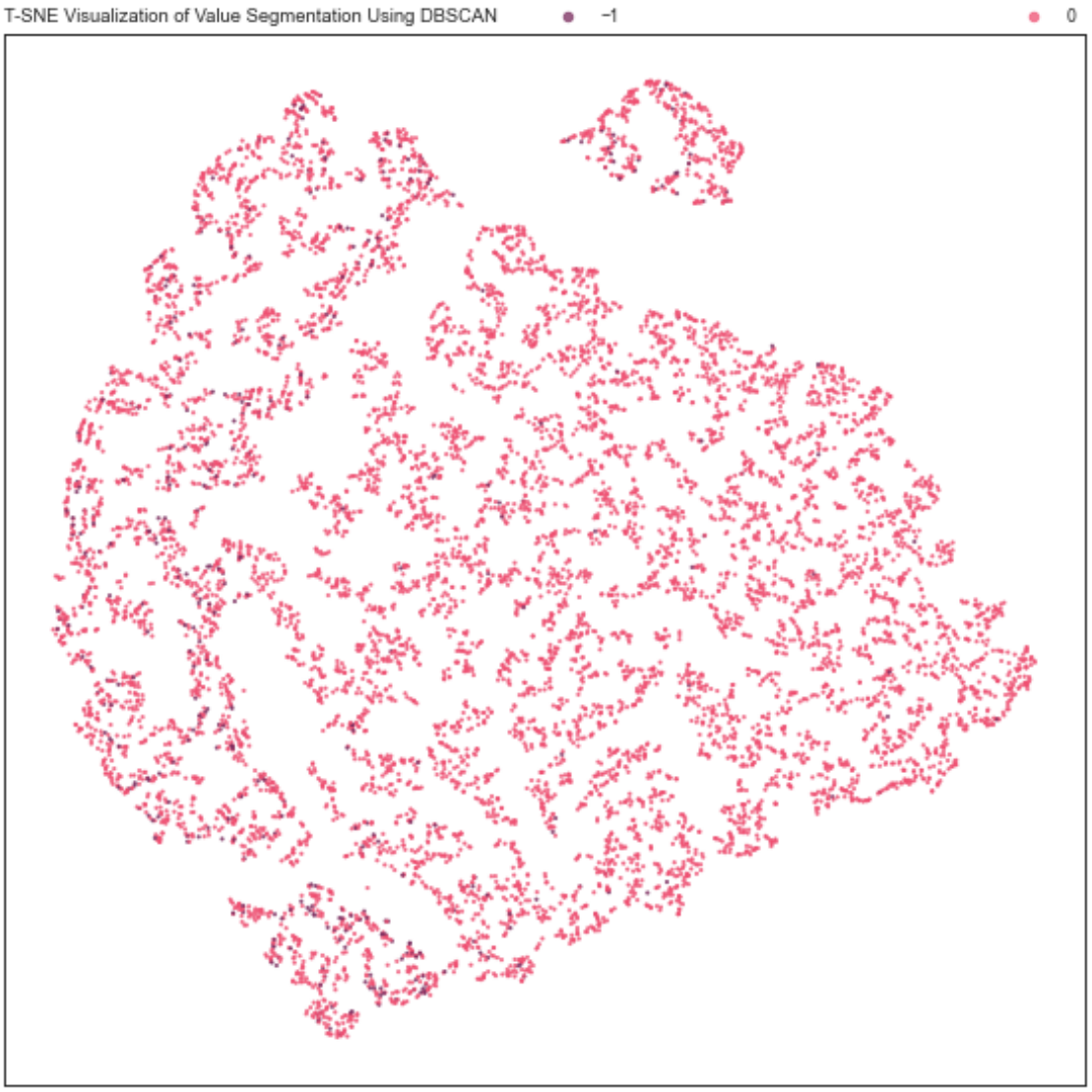
[186]:
```
df_dbv.groupby(['db_value']).count()
```

[186]:

db_value	Dryred_dec	Sweetred_dec	Drywh_dec	Sweetwh_dec	Dessert_dec	Exotic_dec
-1	440	440	440	440	440	440
0	9526	9526	9526	9526	9526	

0 9526

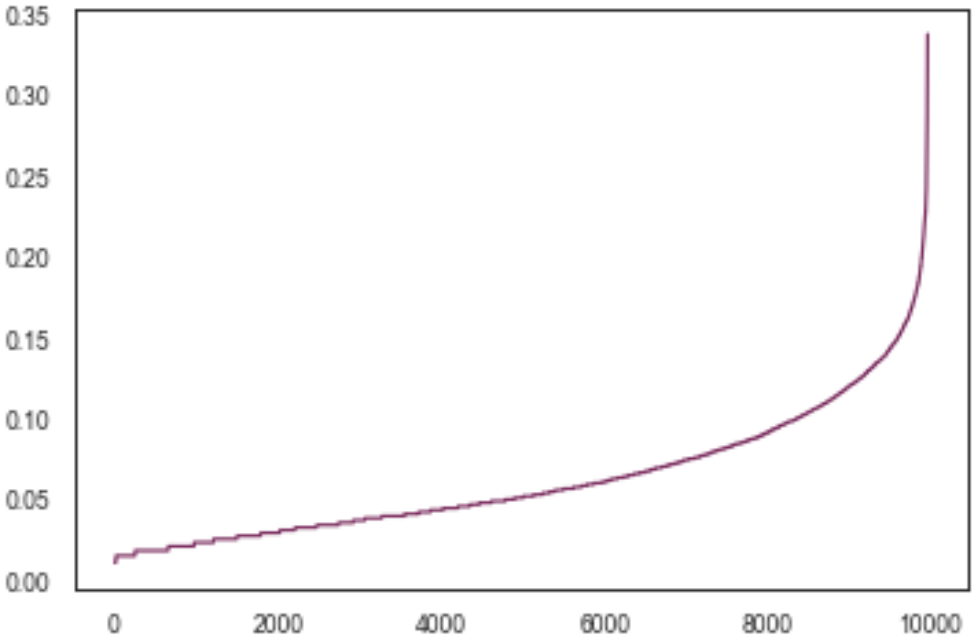
[187]:
```
plot_clusters(tsne_value_somv, df_dbv['db_value'], 'T-SNE Visualization of␣
↪Value Segmentation Using DBSCAN')
```



8.3.2 Wine

[188]:
```
df_db = df_nonoise[wine_feats_dec].copy()

# K-distance graph to find out the right eps value
neigh = NearestNeighbors(n_neighbors=10)
neigh.fit(df_db)
distances, _ = neigh.kneighbors(df_db)
```

```
distances = np.sort(distances[:, -1])
fig = plt.figure()
plt.plot(distances)
save_fig('Epsilon Plot for DBScan', fig)
plt.show()
```



[189]:
```
# Perform DBSCAN clustering
dbscan = DBSCAN(eps=0.13, min_samples=10, n_jobs=4)
dbscan_labels = dbscan.fit_predict(df_db)

dbscan_n_clusters = len(np.unique(dbscan_labels))
print("Number of estimated clusters : %d" % dbscan_n_clusters)

df_db['db_wine'] = dbscan_labels
```

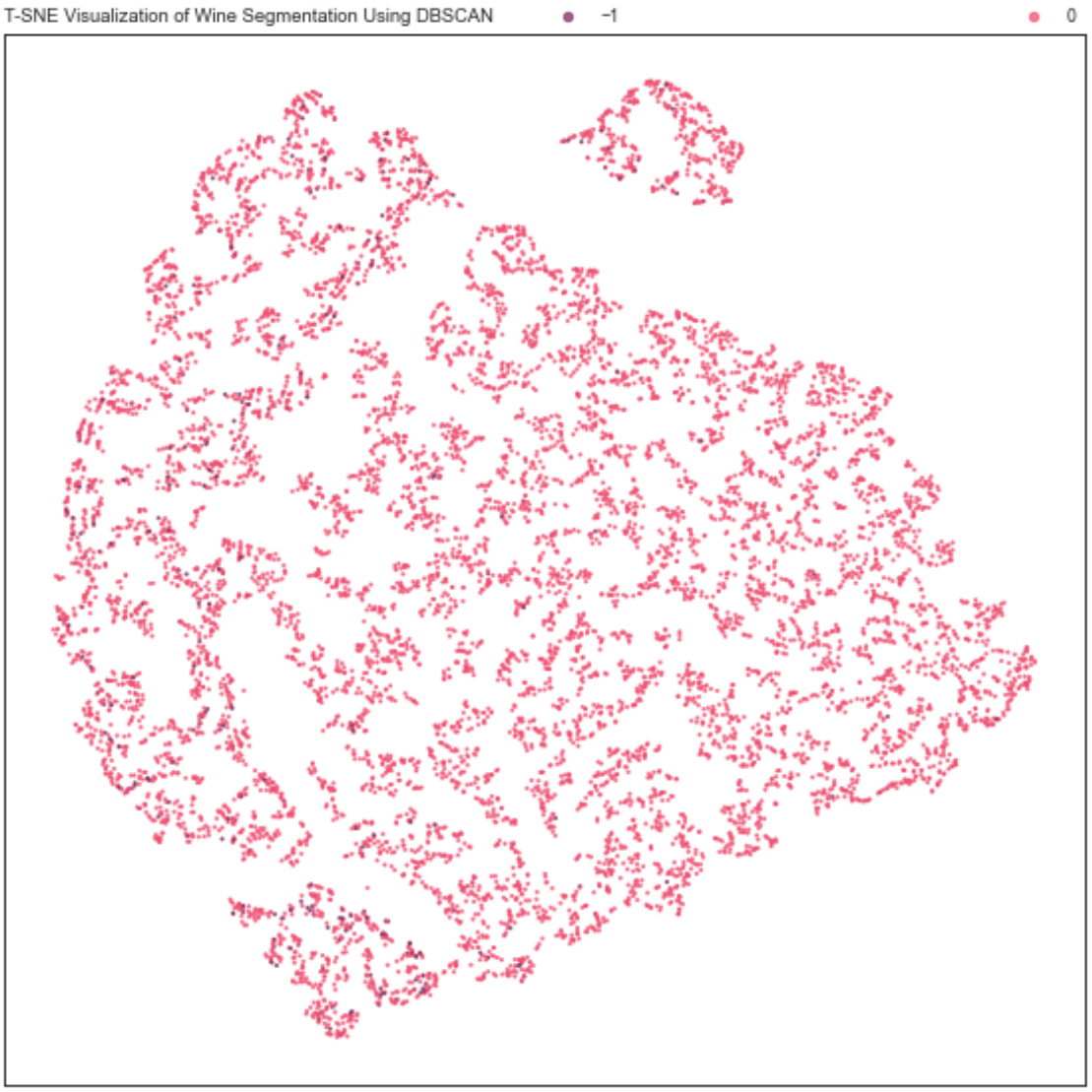
```
Number of estimated clusters : 2
```

[190]:
```
df_db.groupby(['db_wine']).count()
```

[190]:
```
          Dryred_dec  Sweetred_dec  Drywh_dec  Sweetwh_dec  Dessert_dec  \
db_wine
-1               284           284        284          284          284
 0              9682          9682       9682         9682         9682

          Exotic_dec
db_wine
```

```
-1          284
 0         9682
```

[191]:
```
plot_clusters(tsne_value_somv, df_db['db_wine'], 'T-SNE Visualization of Wine␣
↪Segmentation Using DBSCAN')
```



[ ]: