In [1]:
```
%%html
<style>
    .gray {
        background-color: #dfe0e8;
    }
</style>
```

Clustering and Dimensionality Reduction

The data in wine.csv (../data/wine.csv) contains information on 11 chemical properties of 6500 different bottles of vinho verde wine from northern Portugal. In addition, two other variables about each wine are recorded:

- whether the wine is red or white
- the quality of the wine, as judged on a 1-10 scale by a panel of certified wine snobs.

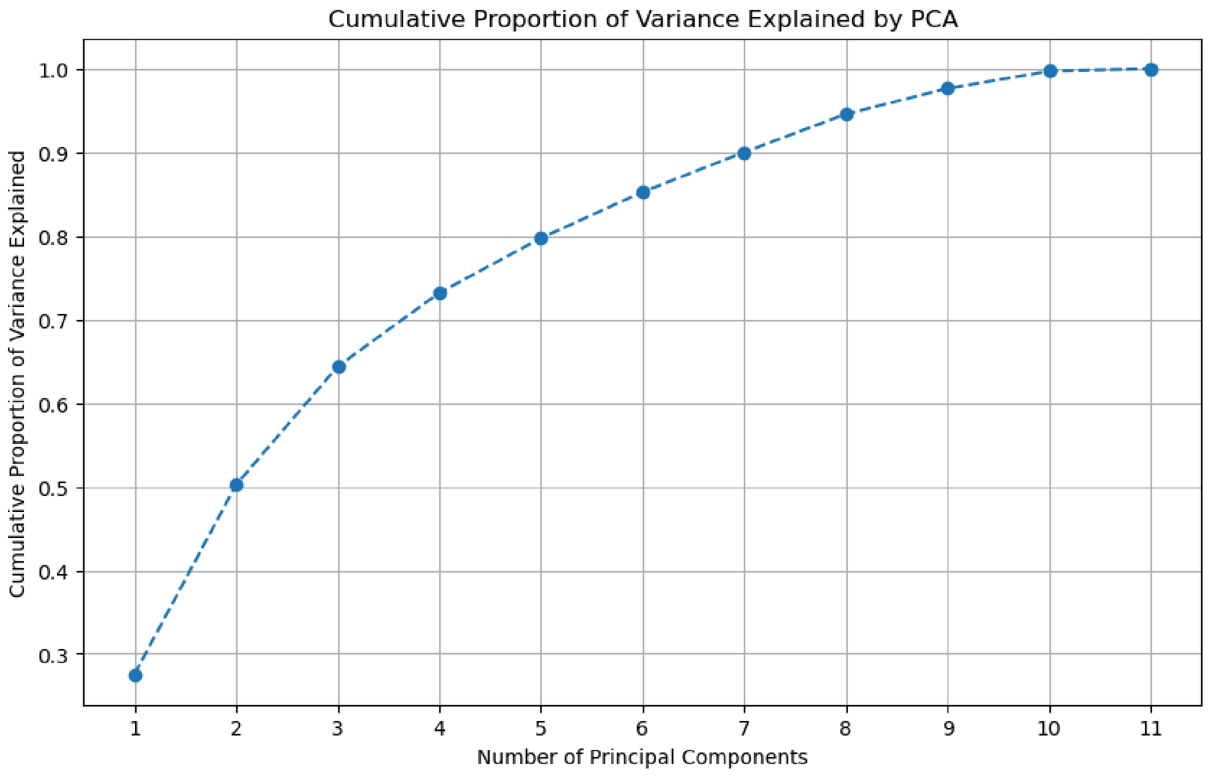
Run PCA, tSNE, and any clustering algorithm of your choice on the 11 chemical properties (or suitable transformations thereof) and summarize your results. Which dimensionality reduction technique makes the most sense to you for this data? Convince yourself (and me) that your chosen approach is easily capable of distinguishing the reds from the whites, using only the “unsupervised” information contained in the data on chemical properties. Does your unsupervised technique also seem capable of distinguishing the higher from the lower quality wines? Present appropriate numerical and/or visual evidence to support your conclusions. To clarify: I’m not asking you to run a supervised learning algorithms. Rather, I’m asking you to see whether the differences in the labels (red/white and quality score) emerge naturally from applying an unsupervised technique to the chemical properties. This should be straightforward to assess using plots

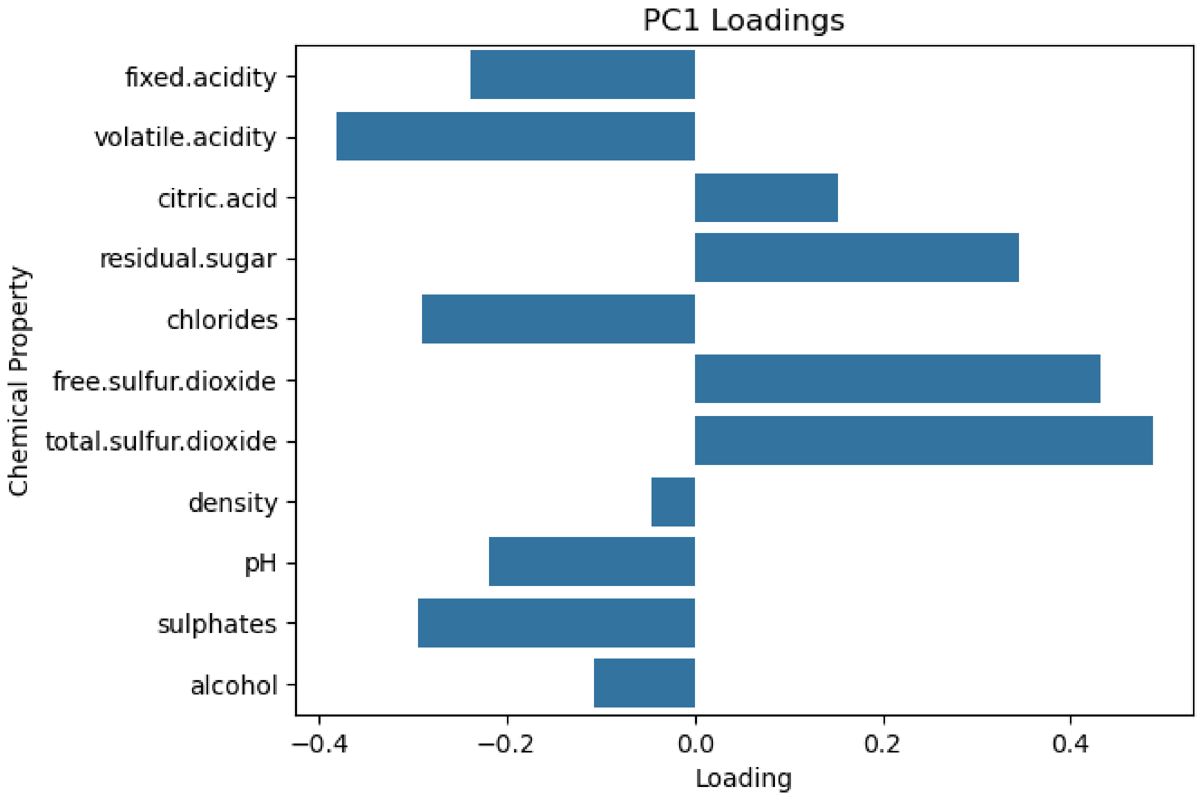
In [ ]:
```
# import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans
```

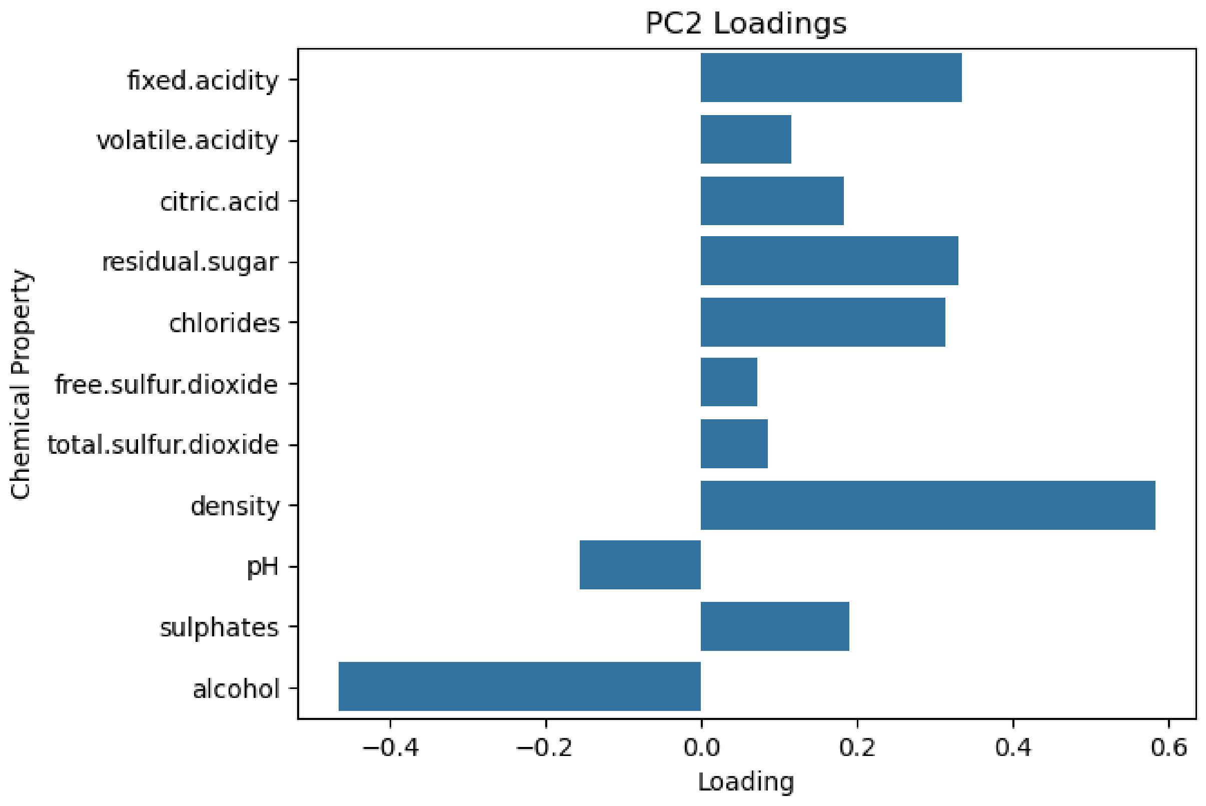
In [20]:
```
# load in the data
wine = pd.read_csv('../data/wine.csv')
chem_props = wine.drop(columns = ['quality', 'color'])
```

In [21]:
```
# scale
scaler = StandardScaler()
scaled_chem_props = scaler.fit_transform(chem_props)
```

```
# get principal components
pca = PCA()
pca_chem_props = pca.fit_transform(scaled_chem_props)
pca_chem_props_df = pd.DataFrame(data = pca_chem_props, columns = ['PCA1','P
                                                                   'PCA5', '
                                                                   'PCA9', '

# determine best number of PCs
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_explained_variance = np.cumsum(explained_variance_ratio)
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(cumulative_explained_variance) + 1), cumulative_explai
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Proportion of Variance Explained')
plt.title('Cumulative Proportion of Variance Explained by PCA')
plt.grid(True)
plt.xticks(range(1, len(cumulative_explained_variance) + 1))
plt.show()

# keep 6 PCs since they account for > 80% of the original variance
pca_chem_props_df = pca_chem_props_df[['PCA1','PCA2', 'PCA3', 'PCA4','PCA5',

# visualize loadings
PC1_loadings = pca.components_[0]
PC1_loadings_df = pd.DataFrame({'Chemical Property':chem_props.columns, 'Loa
sns.barplot(data = PC1_loadings_df, x = 'Loading', y = 'Chemical Property')
plt.title('PC1 Loadings')
plt.show()

PC2_loadings = pca.components_[1]
PC2_loadings_df = pd.DataFrame({'Chemical Property':chem_props.columns, 'Loa
sns.barplot(data = PC2_loadings_df, x = 'Loading', y = 'Chemical Property')
plt.title('PC2 Loadings')
plt.show()

# tSNE
tsne = TSNE(n_components=2, verbose=0, perplexity=40, n_iter=1000, learning_
tsne_results = tsne.fit_transform(scaled_chem_props) # run on standardized c
```


Cumulative Proportion of Variance Explained by PCA
Cumulative Proportion of Variance Explained
Number of Principal Components
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1 2 3 4 5 6 7 8 9 10 11


PC1 Loadings
Chemical Property
fixed.acidity
volatile.acidity
citric.acid
residual.sugar
chlorides
free.sulfur.dioxide
total.sulfur.dioxide
density
pH
sulphates
alcohol
Loading
−0.4
−0.2
0.0
0.2
0.4



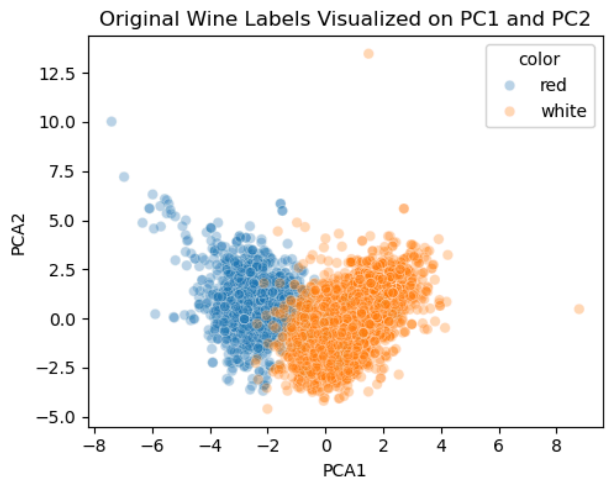
I decided to use 6 principal components to represent the original data because this accounts for more than 805 of the original variance in the data while reducing its dimensionality. I use these principal components to perform clustering.

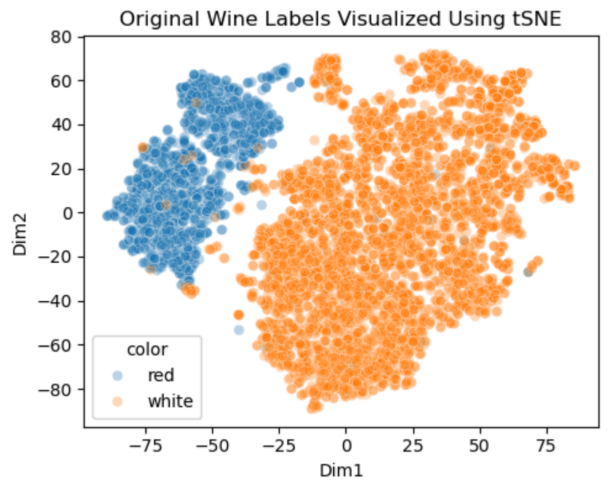
Wine Type Clusters

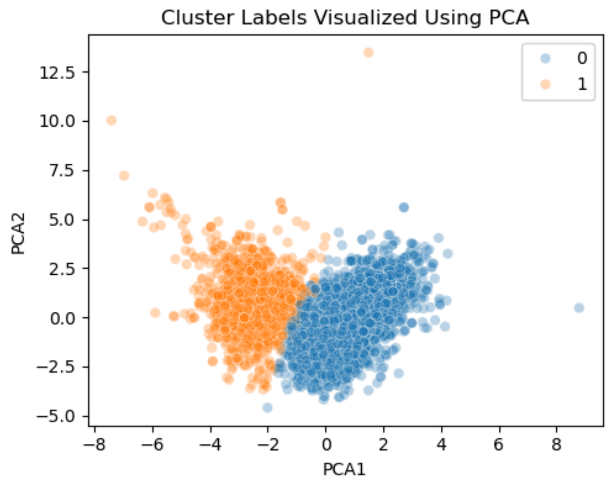
In [22]:
```
# cluster on PCs
kmeans = KMeans(n_clusters = 2, init = 'k-means++', n_init = 10, random_stat
kmeans.fit(pca_chem_props_df)
labels = kmeans.labels_
```

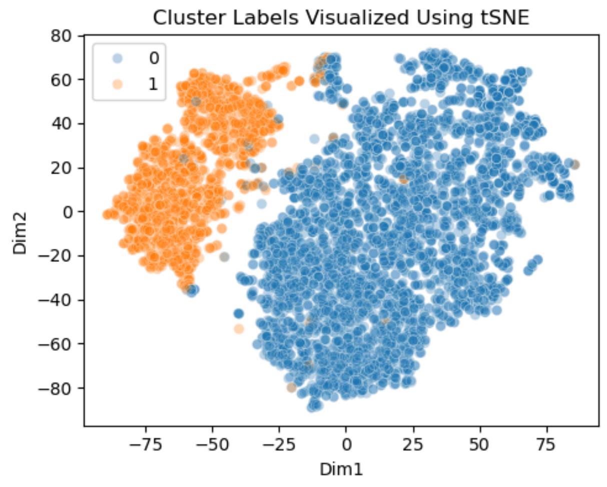
In [23]:
```
fig, axes = plt.subplots(nrows = 3, ncols = 2, figsize=(10, 12))
axes = axes.flatten()

# visualize original labels using PCA
sns.scatterplot(x = pca_chem_props_df['PCA1'], y = pca_chem_props_df['PCA2']
axes[0].set_title('Original Wine Labels Visualized on PC1 and PC2')

# visualize original labels using tSNE
sns.scatterplot(x = tsne_results[:, 0], y = tsne_results[:, 1], alpha = 0.3,
axes[1].set_title('Original Wine Labels Visualized Using tSNE')
axes[1].set_xlabel('Dim1')
axes[1].set_ylabel('Dim2')

# visualize cluster results using PCA
sns.scatterplot(x = pca_chem_props_df['PCA1'], y = pca_chem_props_df['PCA2']
axes[2].set_title('Cluster Labels Visualized Using PCA')
```

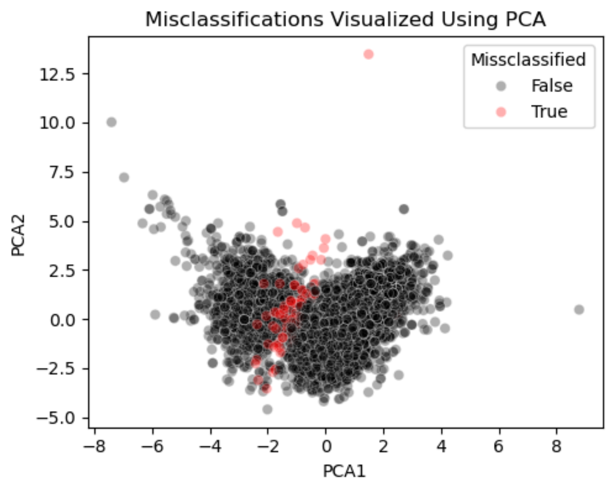
```
# visualize cluster results using tSNE
sns.scatterplot(x = tsne_results[:, 0], y = tsne_results[:, 1], alpha = 0.3,
axes[3].set_title('Cluster Labels Visualized Using tSNE')
axes[3].set_xlabel('Dim1')
axes[3].set_ylabel('Dim2')

# get missclassifications
converted_labels = pd.Series(labels).replace({0:'white', 1:'red'})
misclassifications = converted_labels != wine['color']

# visualize difference in results using PCA
custom_palette = {True: 'red', False: 'black'}
sns.scatterplot(x = pca_chem_props_df['PCA1'], y = pca_chem_props_df['PCA2']
                hue = misclassifications, palette = custom_palette, ax = axe
axes[4].legend(title='Missclassified')
axes[4].set_title('Misclassifications Visualized Using PCA')


# visualize difference in results using tSNE
sns.scatterplot(x = tsne_results[:, 0], y = tsne_results[:, 1], alpha = 0.3,
                hue = misclassifications, palette = custom_palette, ax = axe
axes[5].set_title('Misclassifications Visualized Using tSNE')
axes[5].set_xlabel('Dim1')
axes[5].set_ylabel('Dim2')
axes[5].legend(title='Missclassified')

plt.tight_layout()
plt.show()

# Accuracy
acc = (converted_labels == wine['color']).mean() * 100
print(f'The unsupervised cluster labels mapped onto the wine type {acc:.2f}%
```

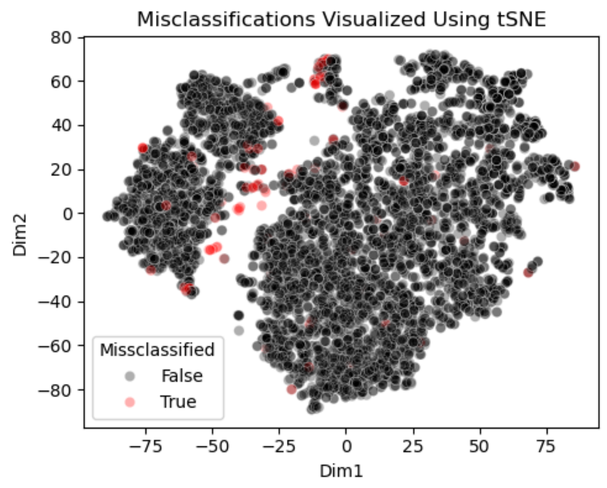










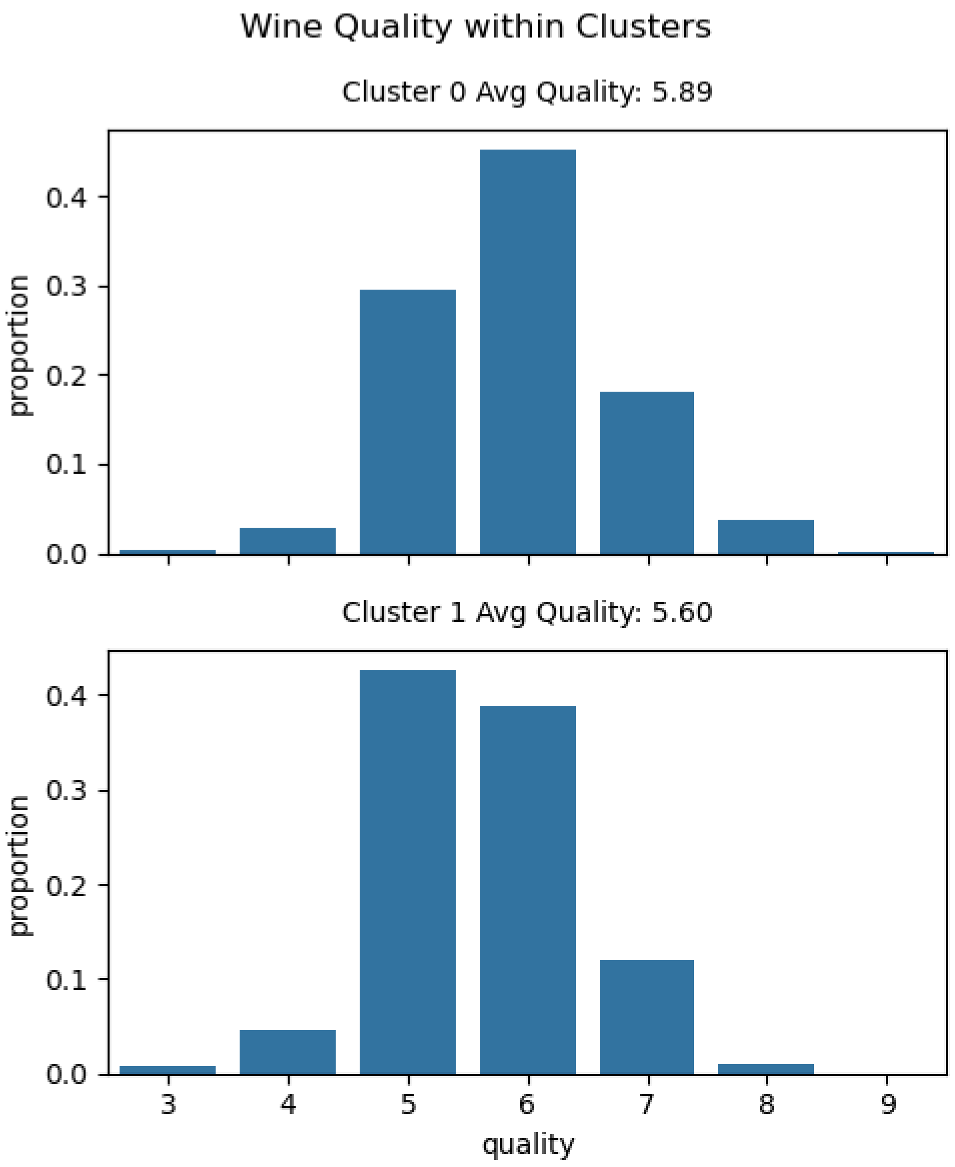


```
The unsupervised cluster labels mapped onto the wine type 98.45% of the time
```

When examining the cluster labels defined by PCA, I found that they correctly mapped onto the wine types 98.45% of the time. PCA was used to capture a majority of the variance in the data. Although t-SNE revealed more of the original data structure and was effective for visualization, we chose to use PCA for clustering to retain more variance while still reducing dimensionality.

Wine Quality Clusters

In [82]:
```python
# cluster on PCs
k = 2
kmeans = KMeans(n_clusters = k, init = 'k-means++', n_init = 10, random_stat
kmeans.fit(pca_chem_props_df)
labels = kmeans.labels_

# histogram of wine qualities
fig, axes = plt.subplots(nrows = k, ncols = 1, figsize = (5,6), sharex = Tru
axes.flatten()
avg_rating = []
for i in range(k):
    cluster_data = wine[labels==i]
    counts = cluster_data['quality'].value_counts()
    props = counts / counts.sum()
    sns.barplot(data = props, ax = axes[i])
    axes[i].set_ylabel('proportion')
    avg_quality = cluster_data['quality'].mean()
    axes[i].text(0.5, 1.05, f'Cluster {i} Avg Quality: {avg_quality:.2f}',
            ha='center', va='bottom', transform=axes[i].transAxes)
    # avg_rating.append(wine.loc[labels == i, 'quality'].mean())
plt.suptitle('Wine Quality within Clusters')
plt.tight_layout()
plt.show()
```

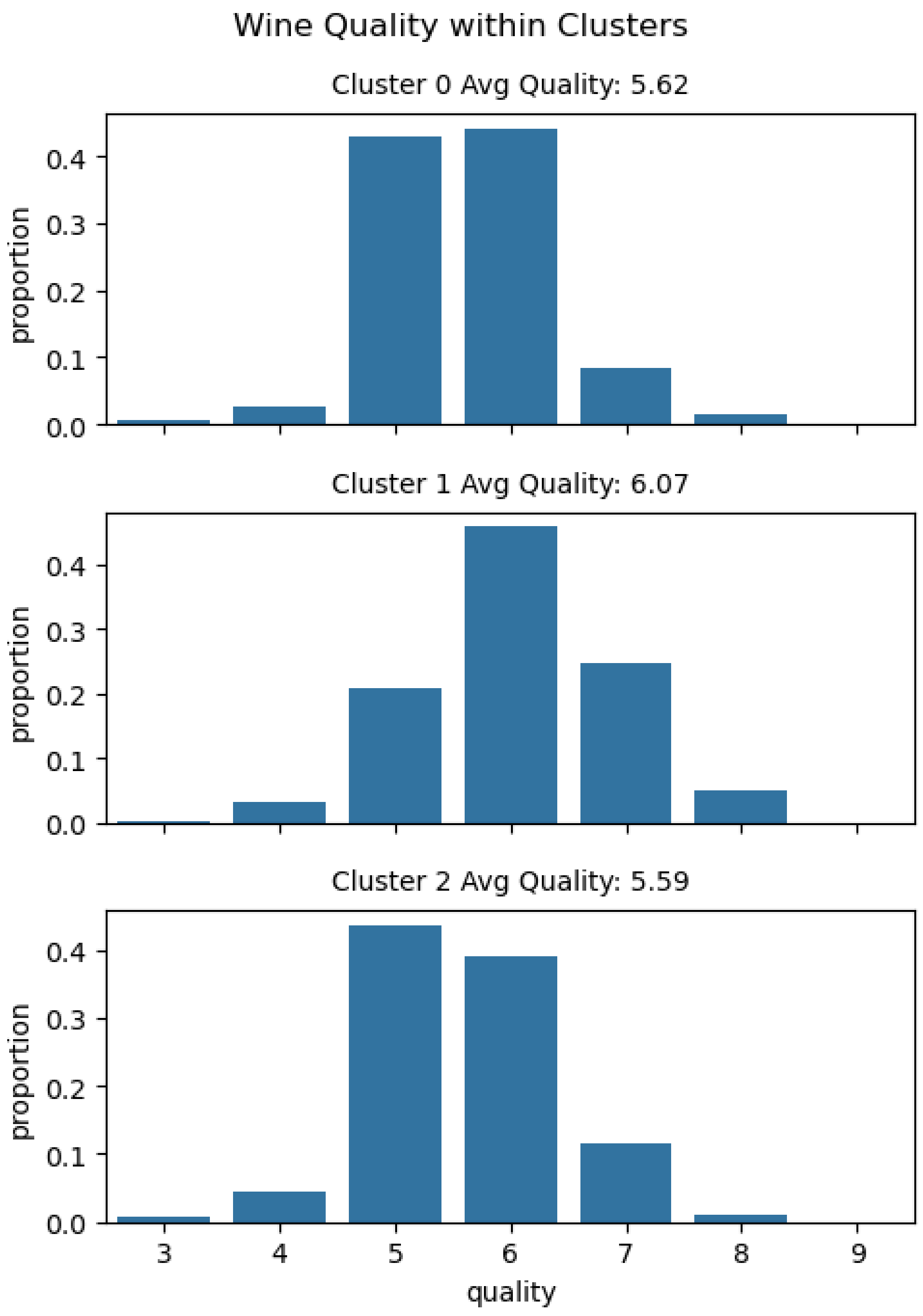


We experimented with various cluster configurations (detailed in the appendix), and ultimately found that two clusters were slightly more optimal. Analyzing the proportions, it appears that Cluster 1 contains a higher proportion of high-quality wines compared to Cluster 0, though both clusters have about average wine quality. Overall, clustering did not offer significant insights into the quality of wines.

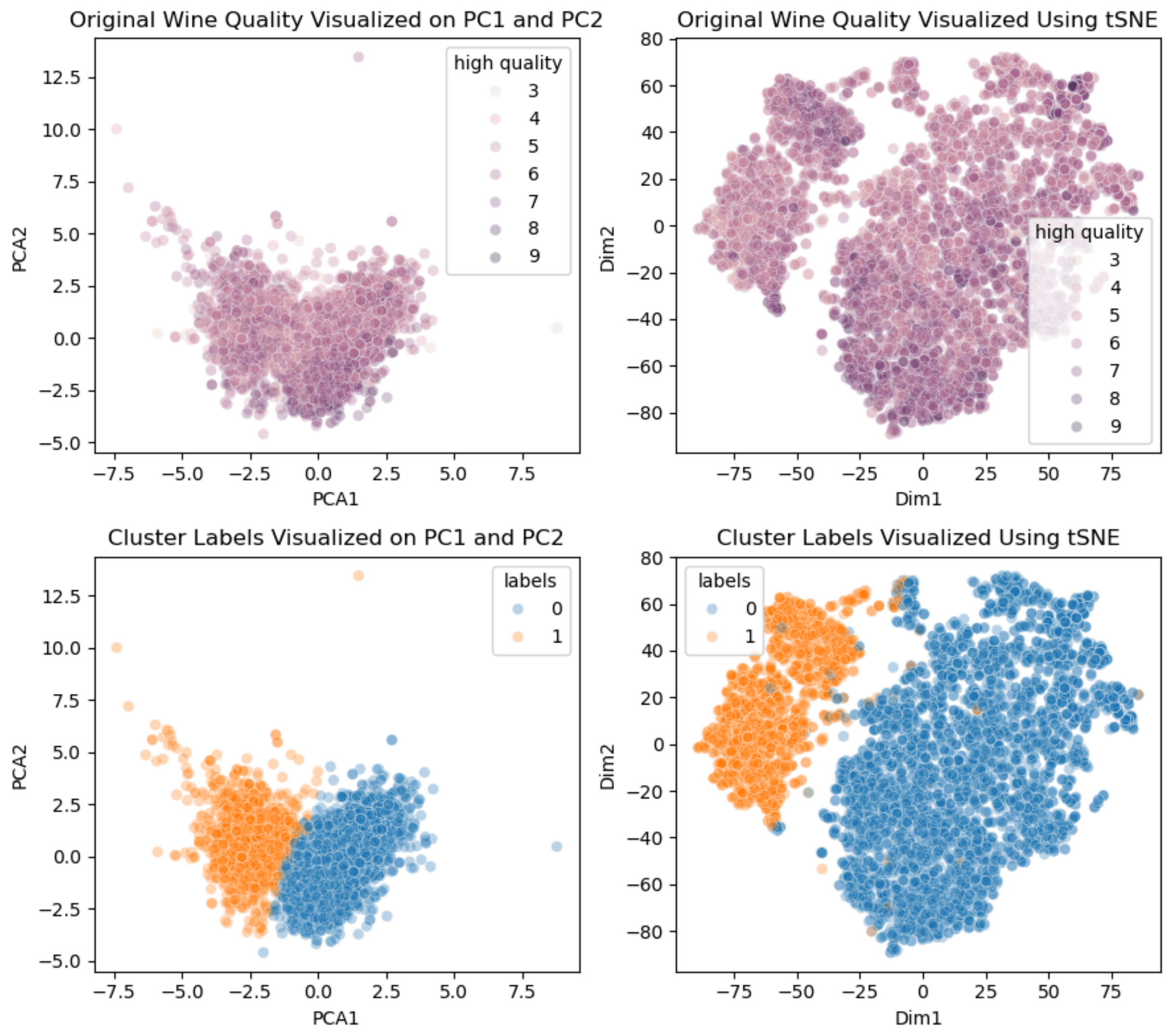
Appendix

In [80]:
```
# cluster on PCs
k = 3
kmeans = KMeans(n_clusters = k, init = 'k-means++', n_init = 10, random_stat
kmeans.fit(pca_chem_props_df)
labels = kmeans.labels_

# histogram of wine qualities
fig, axes = plt.subplots(nrows = k, ncols = 1, figsize = (5,7), sharex = Tru
axes.flatten()
```

```
avg_rating = []
for i in range(k):
    cluster_data = wine[labels==i]
    counts = cluster_data['quality'].value_counts()
    props = counts / counts.sum()
    sns.barplot(data = props, ax = axes[i])
    axes[i].set_ylabel('proportion')
    avg_quality = cluster_data['quality'].mean()
    axes[i].text(0.5, 1.05, f'Cluster {i} Avg Quality: {avg_quality:.2f}',
            ha='center', va='bottom', transform=axes[i].transAxes)
    # avg_rating.append(wine.loc[labels == i, 'quality'].mean())
plt.suptitle('Wine Quality within Clusters')
plt.tight_layout()
plt.show()
```



In [83]:
```python
fig, axes = plt.subplots(nrows = 2, ncols = 2, figsize=(9, 8))
axes = axes.flatten()

high_quality = wine['quality'] > wine['quality'].min()

# visualize original labels using PCA
sns.scatterplot(x = pca_chem_props_df['PCA1'], y = pca_chem_props_df['PCA2']
axes[0].set_title('Original Wine Quality Visualized on PC1 and PC2')
axes[0].legend(title = 'high quality')

# visualize original labels using tSNE
sns.scatterplot(x = tsne_results[:, 0], y = tsne_results[:, 1], alpha = 0.3,
axes[1].set_title('Original Wine Quality Visualized Using tSNE')
axes[1].set_xlabel('Dim1')
axes[1].set_ylabel('Dim2')
axes[1].legend(title = 'high quality')

# visualize cluster labels using PCA
sns.scatterplot(x = pca_chem_props_df['PCA1'], y = pca_chem_props_df['PCA2']
axes[2].set_title('Cluster Labels Visualized on PC1 and PC2')
axes[2].legend(title = 'labels')

# visualize cluster labels using tSNE
sns.scatterplot(x = tsne_results[:, 0], y = tsne_results[:, 1], alpha = 0.3,
axes[3].set_title('Cluster Labels Visualized Using tSNE')
axes[3].set_xlabel('Dim1')
axes[3].set_ylabel('Dim2')
axes[3].legend(title = 'labels')

plt.tight_layout()
plt.show()
```


Original Wine Quality Visualized on PC1 and PC2
PCA1
PCA2
high quality
3
4
5
6
7
8
9
Original Wine Quality Visualized Using tSNE
Dim1
Dim2
Cluster Labels Visualized on PC1 and PC2
labels
0
1
Cluster Labels Visualized Using tSNE