In [14]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, classification_report, confusion_
from sklearn.ensemble import IsolationForest
from imblearn.over_sampling import SMOTE
from sklearn.pipeline import Pipeline
```

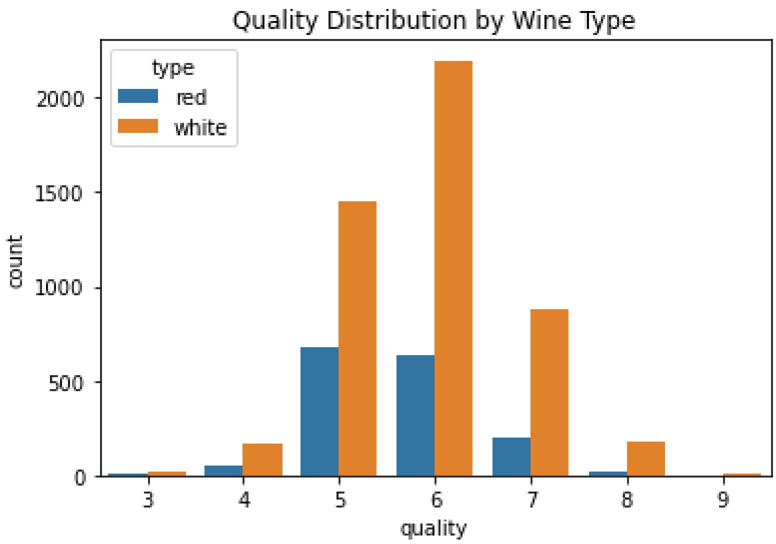
In [16]:

```
# Step 1: Load the datasets
red_wine = pd.read_csv("D:/Downloads/wine+quality/winequality-red.csv", sep=';')
white_wine = pd.read_csv("D:/Downloads/wine+quality/winequality-white.csv", sep=';')

# Combine datasets with an additional column for wine type
red_wine['type'] = 'red'
white_wine['type'] = 'white'
data = pd.concat([red_wine, white_wine], ignore_index=True)
```

In [3]:

```
# Step 2: Data Exploration

print(data.info())
print(data.describe())
sns.countplot(x='quality', hue='type', data=data)
plt.title('Quality Distribution by Wine Type')
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         6497 non-null   float64
 1   volatile acidity      6497 non-null   float64
 2   citric acid           6497 non-null   float64
 3   residual sugar        6497 non-null   float64
 4   chlorides             6497 non-null   float64
 5   free sulfur dioxide   6497 non-null   float64
 6   total sulfur dioxide  6497 non-null   float64
 7   density               6497 non-null   float64
 8   pH                    6497 non-null   float64
 9   sulphates             6497 non-null   float64
 10  alcohol               6497 non-null   float64
 11  quality               6497 non-null   int64  
 12  type                  6497 non-null   object 
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
None
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    6497.000000       6497.000000  6497.000000     6497.000000   
mean        7.215307          0.339666     0.318633        5.443235   
std         1.296434          0.164636     0.145318        4.757804   
min         3.800000          0.080000     0.000000        0.600000   
25%         6.400000          0.230000     0.250000        1.800000   
50%         7.000000          0.290000     0.310000        3.000000   
75%         7.700000          0.400000     0.390000        8.100000   
max        15.900000          1.580000     1.660000       65.800000   

         chlorides  free sulfur dioxide  total sulfur dioxide      density  
\
count  6497.000000          6497.000000           6497.000000  6497.000000   
mean      0.056034            30.525319            115.744574     0.994697   
std       0.035034            17.749400             56.521855     0.002999   
min       0.009000             1.000000              6.000000     0.987110   
25%       0.038000            17.000000             77.000000     0.992340   
50%       0.047000            29.000000            118.000000     0.994890   
75%       0.065000            41.000000            156.000000     0.996990   
max       0.611000           289.000000            440.000000     1.038980   

                pH    sulphates      alcohol      quality  
count  6497.000000  6497.000000  6497.000000  6497.000000  
mean      3.218501     0.531268    10.491801     5.818378  
std       0.160787     0.148806     1.192712     0.873255  
min       2.720000     0.220000     8.000000     3.000000  
25%       3.110000     0.430000     9.500000     5.000000  
50%       3.210000     0.510000    10.300000     6.000000  
75%       3.320000     0.600000    11.300000     6.000000  
max       4.010000     2.000000    14.900000     9.000000  
```



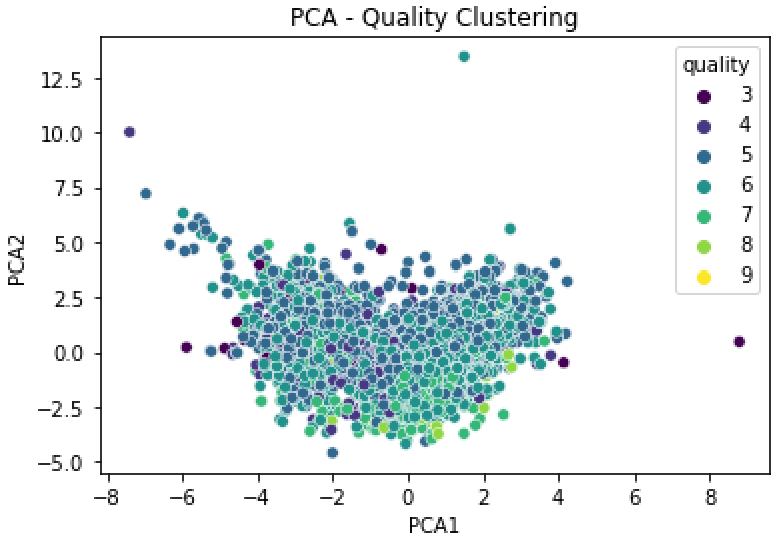
In [17]:

```
# Step 3: Data Preprocessing
features = data.drop(columns=['quality', 'type'])
labels = data['quality']
data['binary_quality'] = (data['quality'] >= 6).astype(int)  # Binary classification

# Standardize features
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Check for class imbalance
print("Class distribution in binary quality:")
print(data['binary_quality'].value_counts())
```

```
Class distribution in binary quality:
1    4113
0    2384
Name: binary_quality, dtype: int64
```

In [18]:

```
# Step 4: PCA
pca = PCA(n_components=2)
features_pca = pca.fit_transform(features_scaled)
data['PCA1'] = features_pca[:, 0]
data['PCA2'] = features_pca[:, 1]

sns.scatterplot(x='PCA1', y='PCA2', hue='quality', data=data, palette='viridis')
plt.title('PCA - Quality Clustering')
plt.show()
```



In [19]:

```
# Step 5: Regression Models with SMOTE to handle class imbalance
X_train, X_test, y_train, y_test = train_test_split(features_scaled, labels, test_size=0.2,
X_train_bin, X_test_bin, y_train_bin, y_test_bin = train_test_split(features_scaled, data['
```

In [20]:

```
# Apply SMOTE to handle class imbalance
smote = SMOTE(random_state=42)
X_train_bin_smote, y_train_bin_smote = smote.fit_resample(X_train_bin, y_train_bin)

# Linear Regression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print("Linear Regression RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
print("Linear Regression R^2:", r2_score(y_test, y_pred))
```

```
Linear Regression RMSE: 0.7393892357602038
Linear Regression R^2: 0.25976731297901656
```

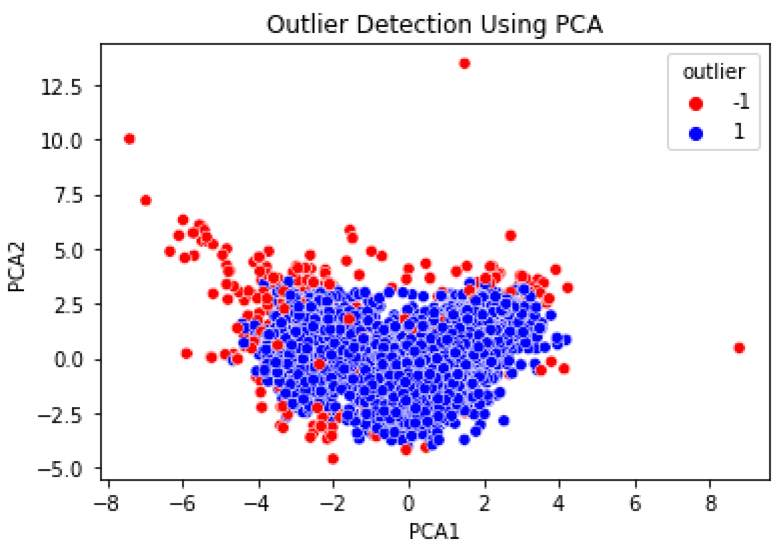
In [21]:

```
# Logistic Regression (binary classification with SMOTE)
logreg = LogisticRegression()
logreg.fit(X_train_bin_smote, y_train_bin_smote)
y_pred_bin = logreg.predict(X_test_bin)
print("Logistic Regression Classification Report:\n", classification_report(y_test_bin, y_p
```

```
Logistic Regression Classification Report:
               precision    recall  f1-score   support

           0       0.57      0.73      0.64       451
           1       0.83      0.70      0.76       849

    accuracy                           0.71      1300
   macro avg       0.70      0.72      0.70      1300
weighted avg       0.74      0.71      0.72      1300
```

In [22]:

```
# Step 6: Outlier Detection
iso_forest = IsolationForest(contamination=0.05, random_state=42)
data['outlier'] = iso_forest.fit_predict(features_scaled)

sns.scatterplot(x='PCA1', y='PCA2', hue='outlier', data=data, palette={1: 'blue', -1: 'red'
plt.title('Outlier Detection Using PCA')
plt.show()
```



In [23]:

```
# Step 7: Feature Selection
from sklearn.linear_model import LassoCV

lasso = LassoCV(cv=5)
lasso.fit(features_scaled, labels)
feature_importance = pd.Series(lasso.coef_, index=features.columns)
important_features = feature_importance[feature_importance != 0].sort_values()

print("Important Features:\n", important_features)

# Visualization of feature importance
important_features.plot(kind='barh', title='Feature Importance (LASSO)')
plt.show()
```

```
Important Features:
 volatile acidity       -0.212084
total sulfur dioxide   -0.072016
chlorides              -0.013085
pH                      0.010393
free sulfur dioxide     0.064013
residual sugar          0.075143
sulphates               0.078423
alcohol                 0.381366
dtype: float64
```