Wine Quality Classification

In [1]:
```
import numpy as np
import seaborn as sns
from ucimlrepo import fetch_ucirepo
from sklearn.cluster import KMeans
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
```

In [2]:
```
wine_quality = fetch_ucirepo(id=186)
X = wine_quality.data.features
y = wine_quality.data.targets
```

In [3]:
```
print(X)
```

```
      fixed_acidity  volatile_acidity  citric_acid  residual_sugar  chlorides  \
0               7.4              0.70         0.00             1.9      0.076   
1               7.8              0.88         0.00             2.6      0.098   
2               7.8              0.76         0.04             2.3      0.092   
3              11.2              0.28         0.56             1.9      0.075   
4               7.4              0.70         0.00             1.9      0.076   
...             ...               ...          ...             ...        ...   
6492            6.2              0.21         0.29             1.6      0.039   
6493            6.6              0.32         0.36             8.0      0.047   
6494            6.5              0.24         0.19             1.2      0.041   
6495            5.5              0.29         0.30             1.1      0.022   
6496            6.0              0.21         0.38             0.8      0.020   

      free_sulfur_dioxide  total_sulfur_dioxide  density    pH  sulphates  \
0                    11.0                  34.0  0.99780  3.51       0.56   
1                    25.0                  67.0  0.99680  3.20       0.68   
2                    15.0                  54.0  0.99700  3.26       0.65   
3                    17.0                  60.0  0.99800  3.16       0.58   
4                    11.0                  34.0  0.99780  3.51       0.56   
...                   ...                   ...      ...   ...        ...   
6492                 24.0                  92.0  0.99114  3.27       0.50   
6493                 57.0                 168.0  0.99490  3.15       0.46   
6494                 30.0                 111.0  0.99254  2.99       0.46   
6495                 20.0                 110.0  0.98869  3.34       0.38   
6496                 22.0                  98.0  0.98941  3.26       0.32   

      alcohol  
0         9.4  
1         9.8  
2         9.8  
3         9.8  
4         9.4  
...       ...  
6492     11.2  
6493      9.6  
6494      9.4  
6495     12.8  
6496     11.8  

[6497 rows x 11 columns]
```

In [4]:
```
print(y)
```

```
      quality
0           5
1           5
2           5
3           6
4           5
...       ...
6492        6
6493        5
6494        6
6495        7
6496        6

[6497 rows x 1 columns]
```

Pre-processing of data

In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [6]:

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

y_train = y_train.values.flatten()
y_test = y_test.values.flatten()
```

Unsupervised K-Means Clustering Model

In [7]:

```
kmeans = KMeans()
kmeans_labels_train = kmeans.fit_predict(X_train_scaled)
```

In [8]:

```
cluster_to_label = {}
for cluster in np.unique(kmeans_labels_train):
    cluster_indices = np.where(kmeans_labels_train == cluster)[0]  # Find the indices of the current cluster
    most_common_label = np.bincount(y_train[cluster_indices]).argmax()  # Get the most common label
    cluster_to_label[cluster] = most_common_label # Label each cluster

kmeans_labels_train_mapped = np.array([cluster_to_label[label] for label in kmeans_labels_train])
```

Unsupervised K-Means Clustering Model

In [9]:

```
kmeans_labels_test = kmeans.predict(X_test_scaled)
kmeans_labels_test_mapped = np.array([cluster_to_label[label] for label in kmeans_labels_test])
```

In [10]:

```
correct_predictions = np.sum(kmeans_labels_test_mapped == y_test)
total_predictions = len(y_test)

model_accuracy = correct_predictions / total_predictions
print(f"Accuracy: {model_accuracy:.4f}")
```

```
Accuracy: 0.4746
```

In [11]:

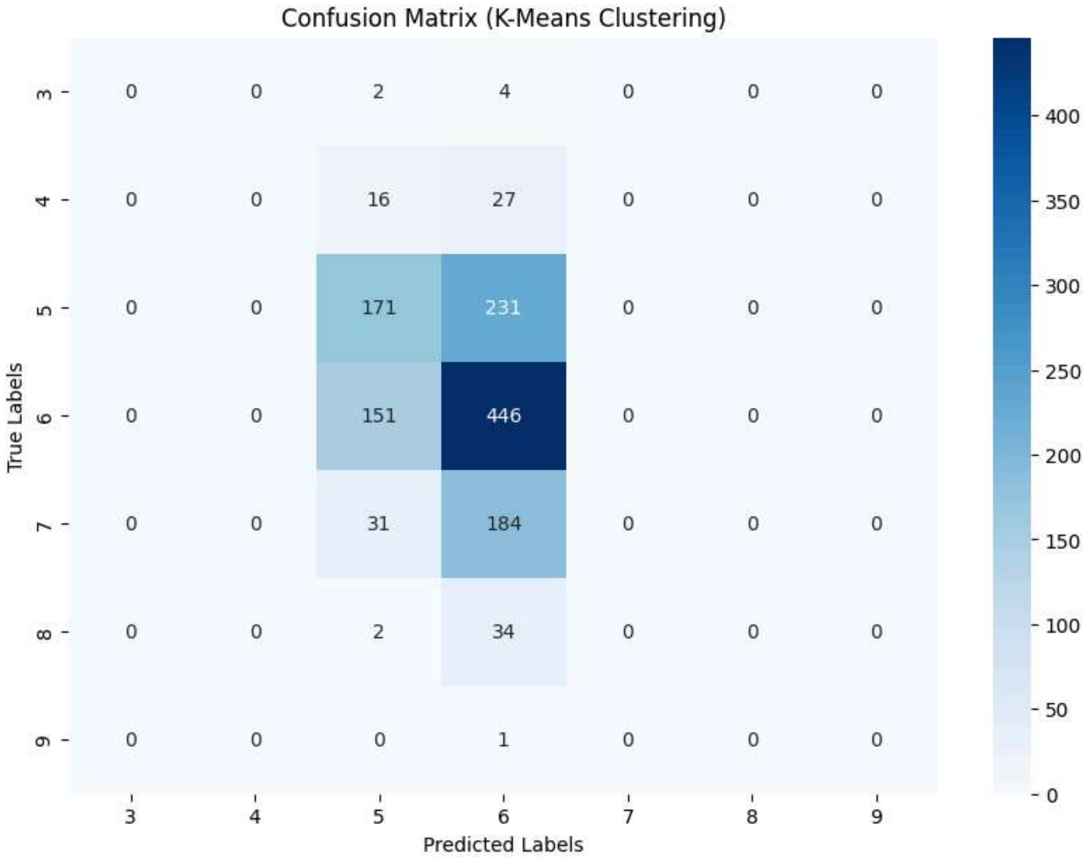
```
report = classification_report(y_test, kmeans_labels_test_mapped, target_names=[str(i) for i in np.unique(y_test)])
print("Classification Report (K-Means Clustering):\n", report)
```

```
Classification Report (K-Means Clustering):
                precision    recall  f1-score   support

           3       0.00      0.00      0.00         6
           4       0.00      0.00      0.00        43
           5       0.46      0.43      0.44       402
           6       0.48      0.75      0.59       597
           7       0.00      0.00      0.00       215
           8       0.00      0.00      0.00        36
           9       0.00      0.00      0.00         1

    accuracy                           0.47      1300
   macro avg       0.13      0.17      0.15      1300
weighted avg       0.36      0.47      0.41      1300
```

```
c:\Users\randa\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics\_classification.py:1565: Undefined
MetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\randa\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics\_classification.py:1565: Undefined
MetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\randa\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics\_classification.py:1565: Undefined
MetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

In [12]:

```
confusion_matrix1 = confusion_matrix(y_test, kmeans_labels_test_mapped)
plt.figure(figsize=(10, 7))
sns.heatmap(confusion_matrix1, annot=True, fmt='d', cmap='Blues', xticklabels=np.unique(y_test), yticklabels=np.unique(y_t
plt.title("Confusion Matrix (K-Means Clustering)")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```



Supervised Logistic Regression Classification Model

In [13]:

```
log_reg_model = LogisticRegression()
log_reg_model.fit(X_train_scaled, y_train)
```

Out[13]:

```
▾ LogisticRegression
LogisticRegression()
```

In [14]:

```
y_pred = log_reg_model.predict(X_test_scaled)
```

In [15]:

```
correct_predictions = np.sum(y_pred == y_test)
total_predictions = len(y_test)

model_accuracy = correct_predictions / total_predictions
print(f"Logistic Regression Accuracy: {model_accuracy:.4f}")
```

```
Logistic Regression Accuracy: 0.5362
```

Results

In [16]:

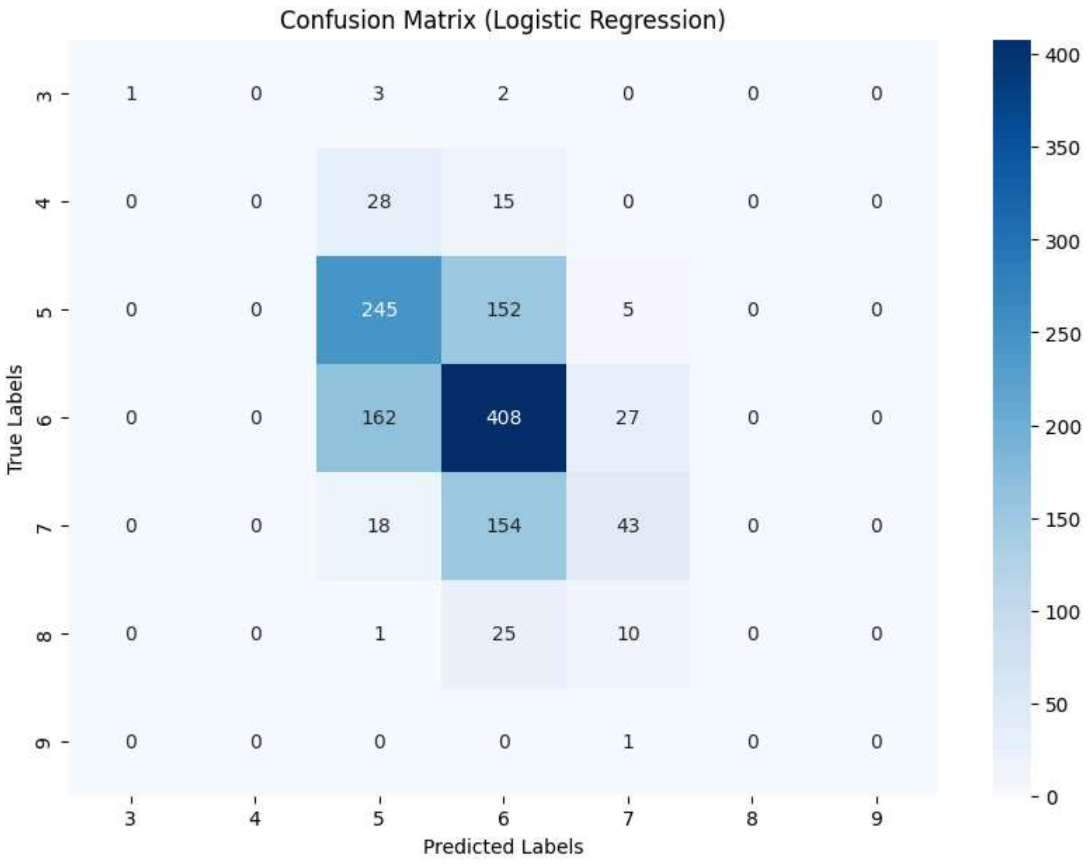
```
report = classification_report(y_test, y_pred, target_names=[str(i) for i in np.unique(y_test)])
print("Classification Report (Logistic Regression):\n", report)
```

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

           3       1.00      0.17      0.29         6
           4       0.00      0.00      0.00        43
           5       0.54      0.61      0.57       402
           6       0.54      0.68      0.60       597
           7       0.50      0.20      0.29       215
           8       0.00      0.00      0.00        36
           9       0.00      0.00      0.00         1

    accuracy                           0.54      1300
   macro avg       0.37      0.24      0.25      1300
weighted avg       0.50      0.54      0.50      1300
```

```
c:\Users\randa\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics\_classification.py:1565: Undefined
MetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\randa\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics\_classification.py:1565: Undefined
MetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\randa\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\metrics\_classification.py:1565: Undefined
MetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

In [17]:

```
confusion_matrix2 = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10, 7))
sns.heatmap(confusion_matrix2, annot=True, fmt='d', cmap='Blues', xticklabels=np.unique(y_test), yticklabels=np.unique(y_t
plt.title("Confusion Matrix (Logistic Regression)")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```


Confusion Matrix (Logistic Regression)
True Labels
Predicted Labels
3: 1 0 3 2 0 0 0
4: 0 0 28 15 0 0 0
5: 0 0 245 152 5 0 0
6: 0 0 162 408 27 0 0
7: 0 0 18 154 43 0 0
8: 0 0 1 25 10 0 0
9: 0 0 0 0 1 0 0