Connected to Python 3.13.5

In [ ]:

```
# LIBRARIES

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from rich import print
from sklearn.preprocessing import PowerTransformer, StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKF
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from sklearn.svm import SVC
from imblearn.over_sampling import SMOTE
from collections import Counter
from sklearn.linear_model import LogisticRegression
import pickle
from IPython.display import display
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier, St
import xgboost as xgb
```

In [ ]:

```
# PREDICTIVE SYSTEM

# Load the trained stacking model
with open(r'stacked_model.pkl', 'rb') as f:
    stacking_clf = pickle.load(f)

# Load the new test data
test_data = pd.read_csv(r'TestWineData.csv')

# Drop 'quality' and 'quality_binned' columns
# Not used for prediction but will be shown in the output
test_data_features = test_data.drop(columns=['quality', 'quality_binned'], errors='

# Make predictions using the trained stacking model
predictions = stacking_clf.predict(test_data_features)

# Map the numeric predictions back to quality categories
quality_map_rev = {0: 'Low Quality', 1: 'Medium Quality', 2: 'High Quality'}
predicted_labels = [quality_map_rev[p] for p in predictions]

# Add the predictions as a new column 'Predicted_Quality' in the new data
test_data['Predicted_Quality'] = predicted_labels

# Display the test data with 'quality', 'quality_binned', and 'Predicted_Quality'
# (if 'quality' and 'quality_binned' exist in the test data)
print(test_data[['quality', 'quality_binned', 'Predicted_Quality']])

# Save the new data with predictions to a new CSV file
test_data.to_csv(r'predicted_wines.csv', index=False)
```

```
      quality  quality_binned Predicted_Quality
0           6  Medium Quality    Medium Quality
1           5  Medium Quality      High Quality
2           5  Medium Quality    Medium Quality
3           6  Medium Quality      High Quality
4           6  Medium Quality    Medium Quality
...       ...             ...               ...
5741        7  Medium Quality    Medium Quality
5742        5  Medium Quality    Medium Quality
5743        9    High Quality    Medium Quality
5744        6  Medium Quality    Medium Quality
5745        6  Medium Quality    Medium Quality

[5746 rows x 3 columns]
```

In [ ]:

```
# CHECKING ACCURACY OF PREDICTED WINES

df = pd.read_csv(r'predicted_wines.csv')

accuracy = (df['Predicted_Quality'] == df['quality_binned']).mean()

# Print the accuracy in percentage
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 83.55%
```

In [ ]:

```
# CHECK TOTAL CLASSIFIED AND MISCLASSIFIED INSTANCES

# Total number of instances in the dataset
total_instances = df.shape[0]

# Total classified instances (correctly classified)
classified_instances = (df['Predicted_Quality'] == df['quality_binned']).sum()

# Misclassified instances (incorrectly classified)
misclassified_instances = total_instances - classified_instances

# Print the results
print(f"Total Instances: {total_instances}")
print(f"Classified Instances (Correctly Classified): {classified_instances}")
print(f"Misclassified Instances (Incorrectly Classified): {misclassified_instances}
```

```
Total Instances: 5746
Classified Instances (Correctly Classified): 4801
Misclassified Instances (Incorrectly Classified): 945
```