In [49]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, f1_score
from sklearn.metrics import classification_report

# Load the dataset
url = "https://raw.githubusercontent.com/tejavenkat473/Machine-Learning/main/winequality-white.csv"
data = pd.read_csv(url)
data
```

Out[49]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.00100	3.00	0.45	8.8	6
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.99400	3.30	0.49	9.5	6
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.99510	3.26	0.44	10.1	6
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	9.9	6
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	9.9	6
...	...	...	...	...	...	...	...	...	...	...	...	...
4893	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	6
4894	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	5
4895	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	6
4896	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	7
4897	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	6

4898 rows × 12 columns

In [50]:
```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898 entries, 0 to 4897
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         4898 non-null   float64
 1   volatile acidity      4898 non-null   float64
 2   citric acid           4898 non-null   float64
 3   residual sugar        4898 non-null   float64
 4   chlorides             4898 non-null   float64
 5   free sulfur dioxide   4898 non-null   float64
 6   total sulfur dioxide  4898 non-null   float64
 7   density               4898 non-null   float64
 8   pH                    4898 non-null   float64
 9   sulphates             4898 non-null   float64
 10  alcohol               4898 non-null   float64
 11  quality               4898 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 459.3 KB
None
```

In [51]:
```
print(data.isnull().sum())
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

In [52]:
```
# Total number of missing values
print("Total number of missing values:", data.isnull().sum().sum())
```

```
Total number of missing values: 0
```

In [53]:
```
print(data.shape)
```

```
(4898, 12)
```

In [54]:
```
duplicate = data.duplicated()
print(duplicate.sum())
```

```
937
```

In [55]:
```
data.drop_duplicates(inplace=True)
```

In [56]:
```
duplicate = data.duplicated()
print(duplicate.sum())
```

```
0
```

In [57]:
```
print(data.shape)
```

```
(3961, 12)
```

In [58]:
```
data.describe()
```

Out[58]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
count	3961.000000	3961.000000	3961.000000	3961.000000	3961.000000	3961.000000	3961.000000	3961.000000	3961.000000	3961.000000
mean	6.839346	0.280538	0.334332	5.914819	0.045905	34.889169	137.193512	0.993790	3.195458	0.490351
std	0.866860	0.103437	0.122446	4.861646	0.023103	17.210021	43.129065	0.002905	0.151546	0.113523
min	3.800000	0.080000	0.000000	0.600000	0.009000	2.000000	9.000000	0.987110	2.720000	0.220000
25%	6.300000	0.210000	0.270000	1.600000	0.035000	23.000000	106.000000	0.991620	3.090000	0.410000
50%	6.800000	0.260000	0.320000	4.700000	0.042000	33.000000	133.000000	0.993500	3.180000	0.480000
75%	7.300000	0.330000	0.390000	8.900000	0.050000	45.000000	166.000000	0.995710	3.290000	0.550000
max	14.200000	1.100000	1.660000	65.800000	0.346000	289.000000	440.000000	1.038980	3.820000	1.080000

In [59]:
```
data.corr()
```

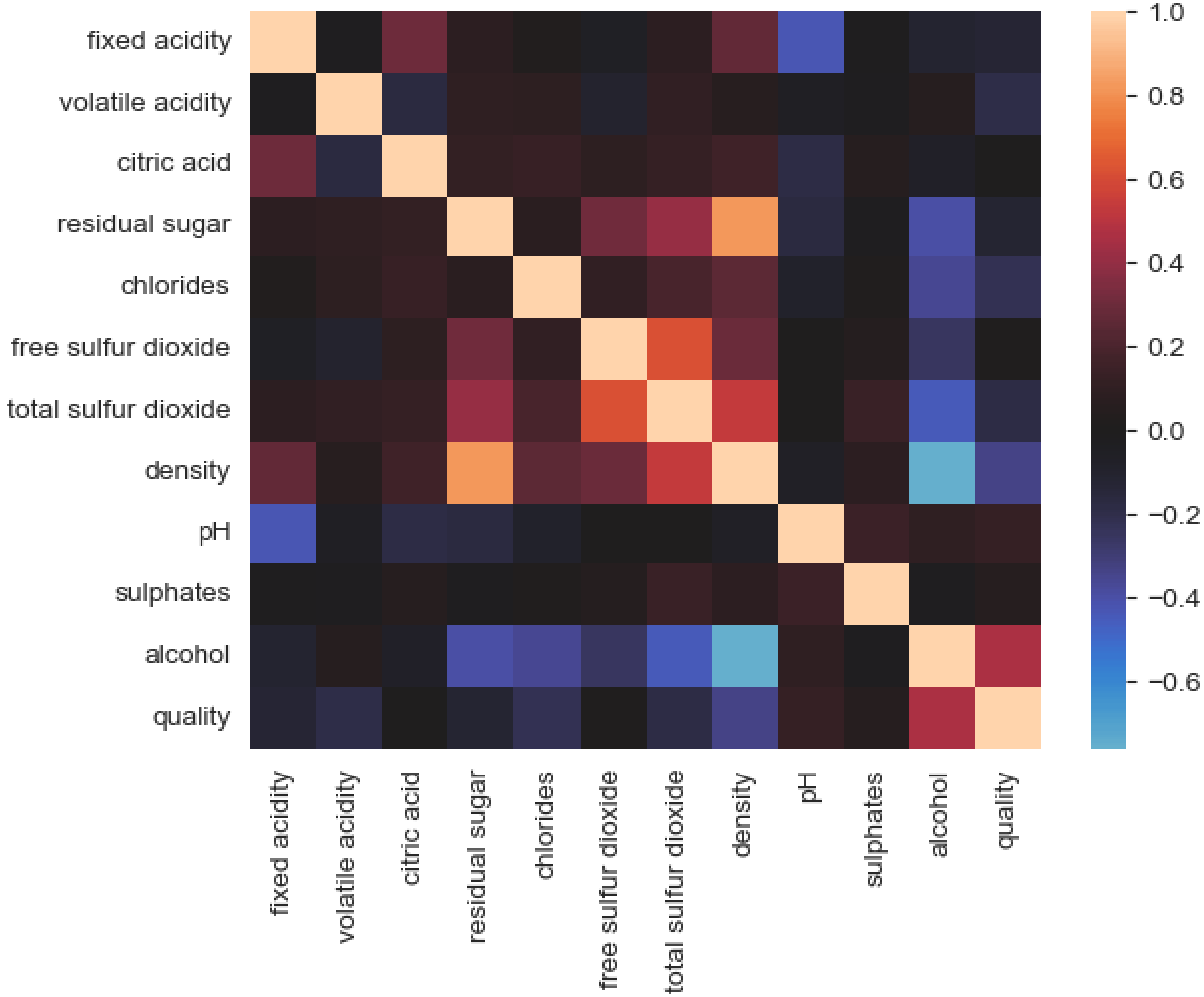
Out[59]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	-0.019214	0.298959	0.083620	0.024036	-0.058396	0.082425	0.266091	-0.431274	-0.017453	-0.110788	-0.124636
volatile acidity	-0.019214	1.000000	-0.163228	0.098340	0.086287	-0.102471	0.102315	0.060603	-0.046954	-0.021150	0.046815	-0.190678
citric acid	0.298959	-0.163228	1.000000	0.106269	0.132590	0.091681	0.122845	0.160076	-0.183015	0.049442	-0.076514	0.007065
residual sugar	0.083620	0.098340	0.106269	1.000000	0.076091	0.306835	0.409583	0.820498	-0.165997	-0.020503	-0.398167	-0.117339
chlorides	0.024036	0.086287	0.132590	0.076091	1.000000	0.101272	0.191145	0.253088	-0.090573	0.017871	-0.356928	-0.217739
free sulfur dioxide	-0.058396	-0.102471	0.091681	0.306835	0.101272	1.000000	0.619437	0.294638	-0.007750	0.037932	-0.251768	0.010507
total sulfur dioxide	0.082425	0.102315	0.122845	0.409583	0.191145	0.619437	1.000000	0.536868	0.008239	0.136544	-0.446643	-0.183356
density	0.266091	0.060603	0.160076	0.820498	0.253088	0.294638	0.536868	1.000000	-0.063734	0.082048	-0.760162	-0.337805
pH	-0.431274	-0.046954	-0.183015	-0.165997	-0.090573	-0.007750	0.008239	-0.063734	1.000000	0.142353	0.093095	0.123829
sulphates	-0.017453	-0.021150	0.049442	-0.020503	0.017871	0.037932	0.136544	0.082048	0.142353	1.000000	-0.022850	0.053200
alcohol	-0.110788	0.046815	-0.076514	-0.398167	-0.356928	-0.251768	-0.446643	-0.760162	0.093095	-0.022850	1.000000	0.462869
quality	-0.124636	-0.190678	0.007065	-0.117339	-0.217739	0.010507	-0.183356	-0.337805	0.123829	0.053200	0.462869	1.000000

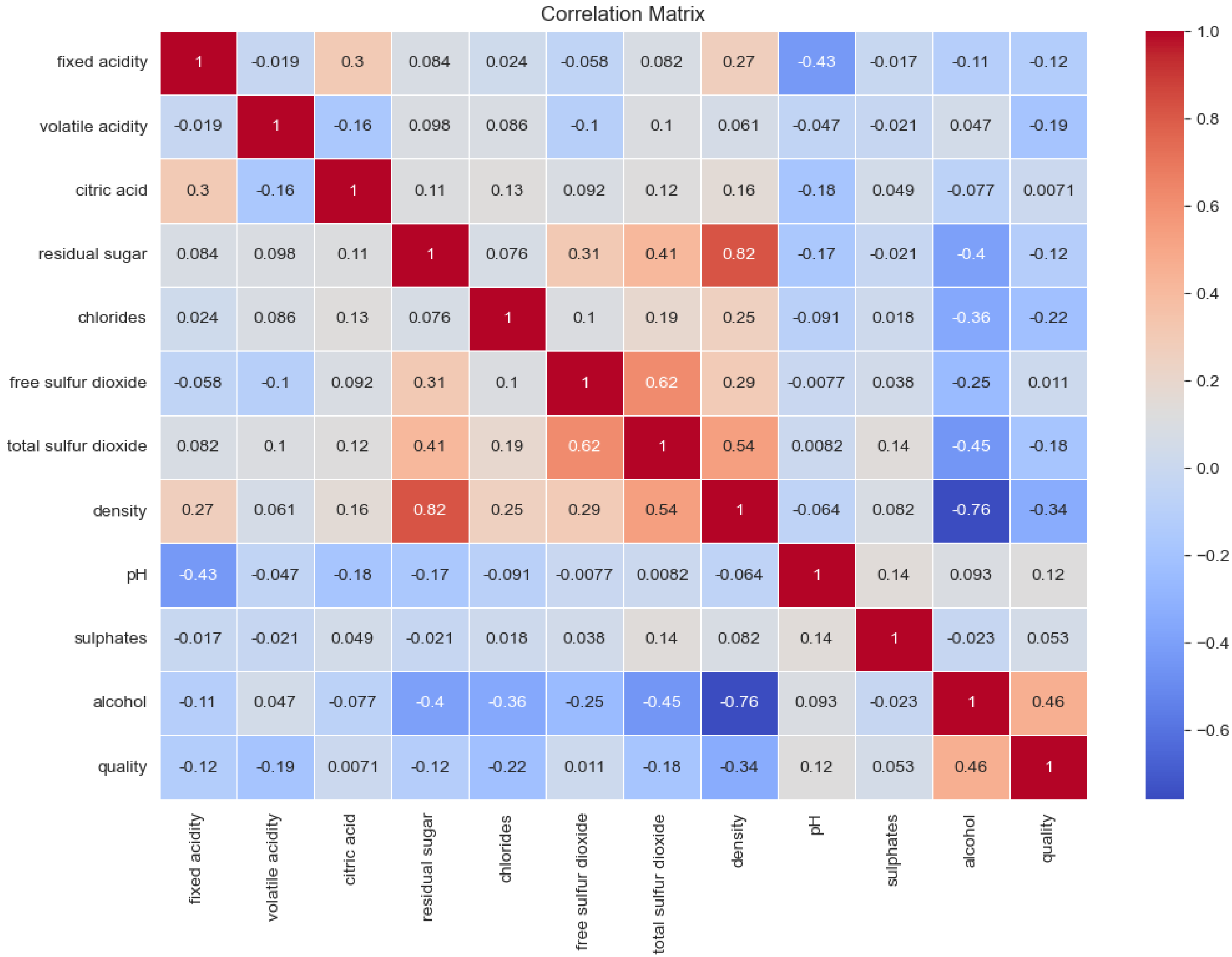
In [60]:

```
sns.heatmap(data.corr(), center=0)
```

Out[60]: <AxesSubplot:>

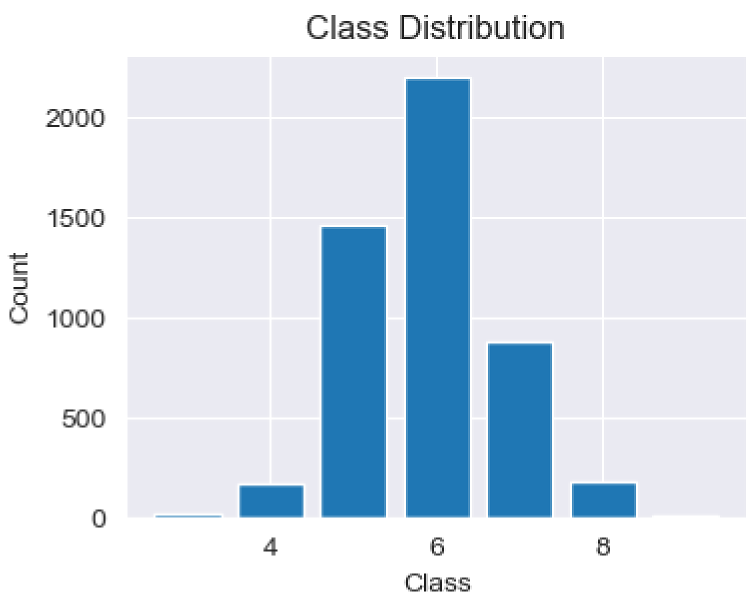


In [61]:
```python
# Calculate the correlation matrix
corr_matrix = data.corr()

# Display the correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```



In [95]:
```python
import pandas as pd
import matplotlib.pyplot as plt
```

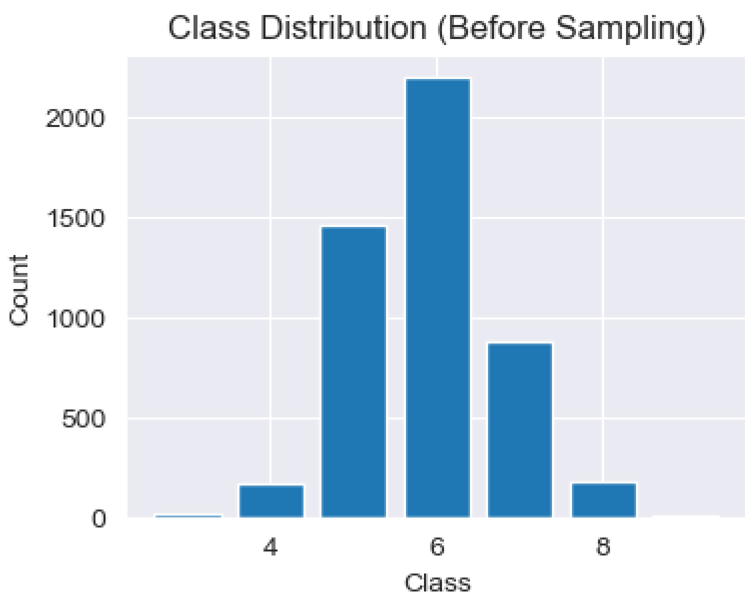
```
# Import the dataset
url = "https://raw.githubusercontent.com/tejavenkat473/Machine-Learning/main/winequality-white.csv"
data = pd.read_csv(url)

# Specify the target variable
target_variable = 'quality'  # Assuming 'quality' is the column to predict

# Count the occurrences of each class
class_counts = data[target_variable].value_counts()

# Plot the class distribution
plt.figure(figsize=(4, 3))
plt.bar(class_counts.index, class_counts.values)
plt.xlabel('Class')
plt.ylabel('Count')
plt.title('Class Distribution')
plt.show()

# Check if class imbalance exists
is_imbalanced = any(class_counts.values != class_counts.values[0])
if is_imbalanced:
    print("Class imbalance exists.")
else:
    print("Class imbalance does not exist.")
```

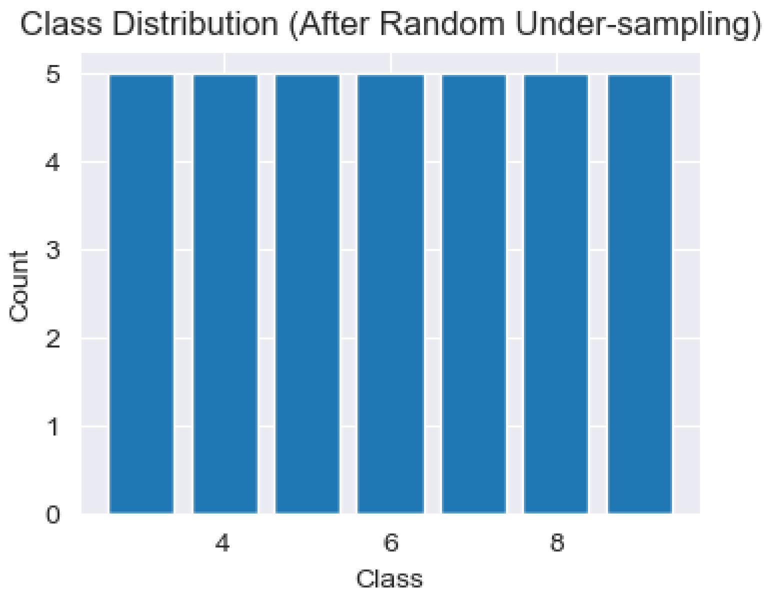



```
Class imbalance exists.
```

In [96]:
```
import pandas as pd
import matplotlib.pyplot as plt
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import RandomOverSampler, SMOTE

# Import the dataset
url = "https://raw.githubusercontent.com/tejavenkat473/Machine-Learning/main/winequality-white.csv"
data = pd.read_csv(url)

# Specify the target variable
target_variable = 'quality'  # Assuming 'quality' is the column to predict

# Count the occurrences of each class before sampling
class_counts_before = data[target_variable].value_counts()

# Plot the class distribution before sampling
plt.figure(figsize=(4, 3))
plt.bar(class_counts_before.index, class_counts_before.values)
plt.xlabel('Class')
plt.ylabel('Count')
plt.title('Class Distribution (Before Sampling)')
plt.show()

# Perform Random Under-sampling
rus = RandomUnderSampler(random_state=42)
X_resampled_under, y_resampled_under = rus.fit_resample(data.drop(target_variable, axis=1), data[target_variable])

# Count the occurrences of each class after Random Under-sampling
class_counts_under = pd.Series(y_resampled_under).value_counts()

# Plot the class distribution after Random Under-sampling
plt.figure(figsize=(4, 3))
plt.bar(class_counts_under.index, class_counts_under.values)
plt.xlabel('Class')
plt.ylabel('Count')
plt.title('Class Distribution (After Random Under-sampling)')
plt.show()

# Perform Random Over-sampling
ros = RandomOverSampler(random_state=42)
X_resampled_over, y_resampled_over = ros.fit_resample(data.drop(target_variable, axis=1), data[target_variable])

# Count the occurrences of each class after Random Over-sampling
class_counts_over = pd.Series(y_resampled_over).value_counts()
```

```
# Plot the class distribution after Random Over-sampling
plt.figure(figsize=(4, 3))
plt.bar(class_counts_over.index, class_counts_over.values)
plt.xlabel('Class')
plt.ylabel('Count')
plt.title('Class Distribution (After SMOTE Over-sampling)')
plt.show()
```





In [97]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

```
url = "https://raw.githubusercontent.com/tejavenkat473/Machine-Learning/main/winequality-white.csv"
data = pd.read_csv(url)

# Preprocessing
X = data.drop('quality', axis=1)  # Assuming 'target_variable' is the column to predict
y = data['quality']

# Convert categorical variables to numerical using label encoding
label_encoder = LabelEncoder()
X_encoded = X.apply(label_encoder.fit_transform)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)


dt_model = DecisionTreeClassifier()
dt_model.fit(X_train_scaled, y_train)
dt_predictions = dt_model.predict(X_test_scaled)
dt_accuracy = accuracy_score(y_test, dt_predictions)

print("Decision Tree Accuracy:", dt_accuracy)
```

```
Decision Tree Accuracy: 0.6051020408163266
```

In [98]:

```
# Calculate confusion matrix
dt_cm = confusion_matrix(y_test, dt_predictions)

# Calculate F1 score
dt_f1 = f1_score(y_test, dt_predictions, average='weighted')

print("Decision Tree Confusion Matrix:")
print(dt_cm)
print("Decision Tree F1 Score:", dt_f1)
```

```
Decision Tree Confusion Matrix:
[[  0   1   2   2   0   0   0]
 [  1   7   7   7   2   1   0]
 [  0  14 185  81   9   2   0]
 [  1  10  80 277  51  12   1]
 [  3   2   5  56 108  17   1]
 [  0   1   0   7  11  16   0]
 [  0   0   0   0   0   0   0]]
Decision Tree F1 Score: 0.6092344123362659
```

In [99]:
```
# Generate classification report
dt_classification_report = classification_report(y_test, dt_predictions)

print("Decision Tree Classification Report:")
print(dt_classification_report)
```

```
Decision Tree Classification Report:
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         5
           4       0.20      0.28      0.23        25
           5       0.66      0.64      0.65       291
           6       0.64      0.64      0.64       432
           7       0.60      0.56      0.58       192
           8       0.33      0.46      0.39        35
           9       0.00      0.00      0.00         0

    accuracy                           0.61       980
   macro avg       0.35      0.37      0.36       980
weighted avg       0.61      0.61      0.61       980
```

```
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Rec
all and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to c
ontrol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Rec
all and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to c
ontrol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Rec
all and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to c
ontrol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [7]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

url = "https://raw.githubusercontent.com/tejavenkat473/Machine-Learning/main/winequality-white.csv"
data = pd.read_csv(url)
```

```
# Preprocessing
X = data.drop('quality', axis=1) # Assuming 'target_variable' is the column to predict
y = data['quality']

# Convert categorical variables to numerical using label encoding
label_encoder = LabelEncoder()
X_encoded = X.apply(label_encoder.fit_transform)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

# Create instances of the classification algorithms
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)
lr_predictions = lr_model.predict(X_test)
lr_accuracy = accuracy_score(y_test, lr_predictions)
print("Logistic Regression Accuracy:", lr_accuracy)
```

```
Logistic Regression Accuracy: 0.523469387755102
```

```
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs fai
led to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [104…

```
# Calculate confusion matrix
lr_cm = confusion_matrix(y_test, lr_predictions)

# Calculate F1 score
lr_f1 = f1_score(y_test, lr_predictions, average='weighted')

print("Logistic Regression Confusion Matrix:")
print(lr_cm)
print("Logistic Regression F1 Score:", lr_f1)
```

```
Logistic Regression Confusion Matrix:
[[  0   0   2   2   0   1   0]
 [  0   1  12  12   0   0   0]
 [  0   1 151 137   1   1   0]
 [  0   0  80 322  29   0   1]
 [  0   0  13 133  46   0   0]
 [  0   0   2  25   8   0   0]
 [  0   0   0   0   0   0   0]]
Logistic Regression F1 Score: 0.4970071183496042
```

In [102…

```
# Generate classification report
lr_classification_report = classification_report(y_test, lr_predictions)

print("Logistic Regression Classification Report:")
print(lr_classification_report)
```

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

           3       0.00      0.00      0.00         5
           4       0.50      0.04      0.07        25
           5       0.58      0.52      0.55       291
           6       0.51      0.75      0.61       432
           7       0.55      0.24      0.33       192
           8       0.00      0.00      0.00        35
           9       0.00      0.00      0.00         0

    accuracy                           0.53       980
   macro avg       0.31      0.22      0.22       980
weighted avg       0.52      0.53      0.50       980
```

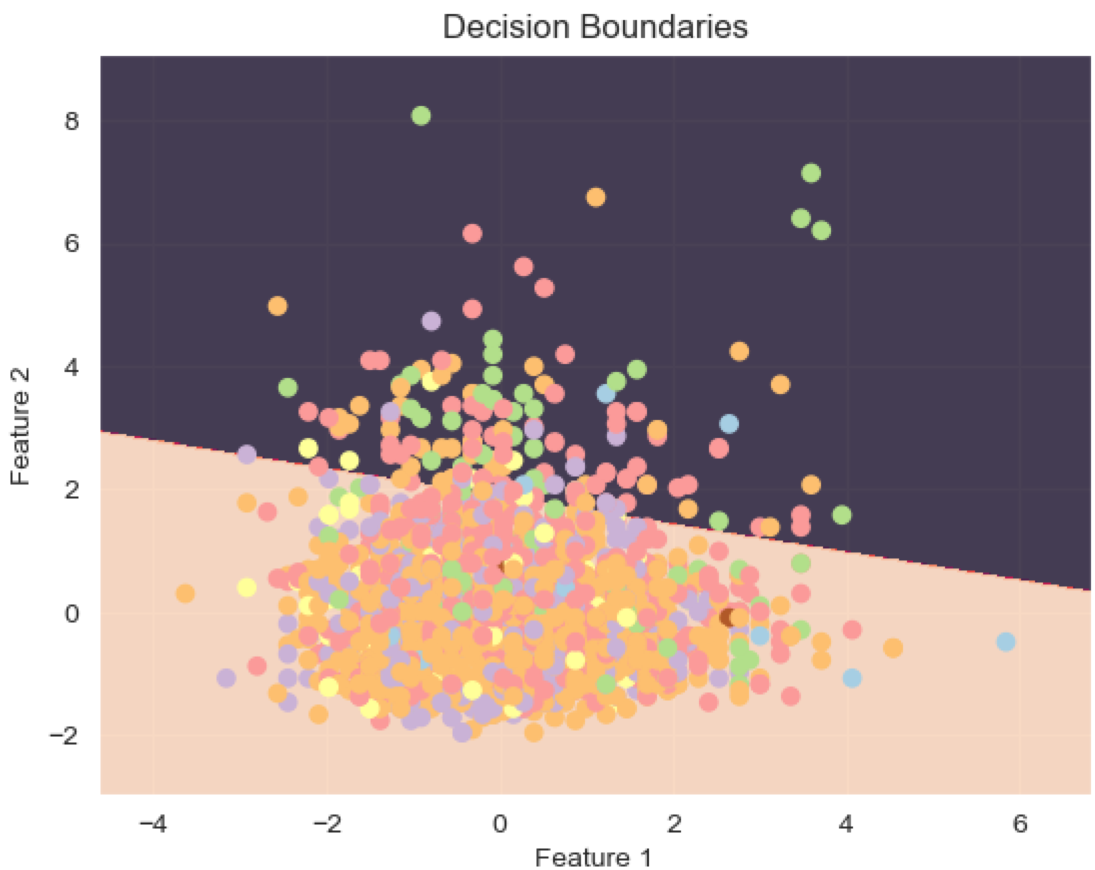
```
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parame
ter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Rec
all and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to c
ontrol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parame
ter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Rec
all and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to c
ontrol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parame
ter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Rec
all and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to c
ontrol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

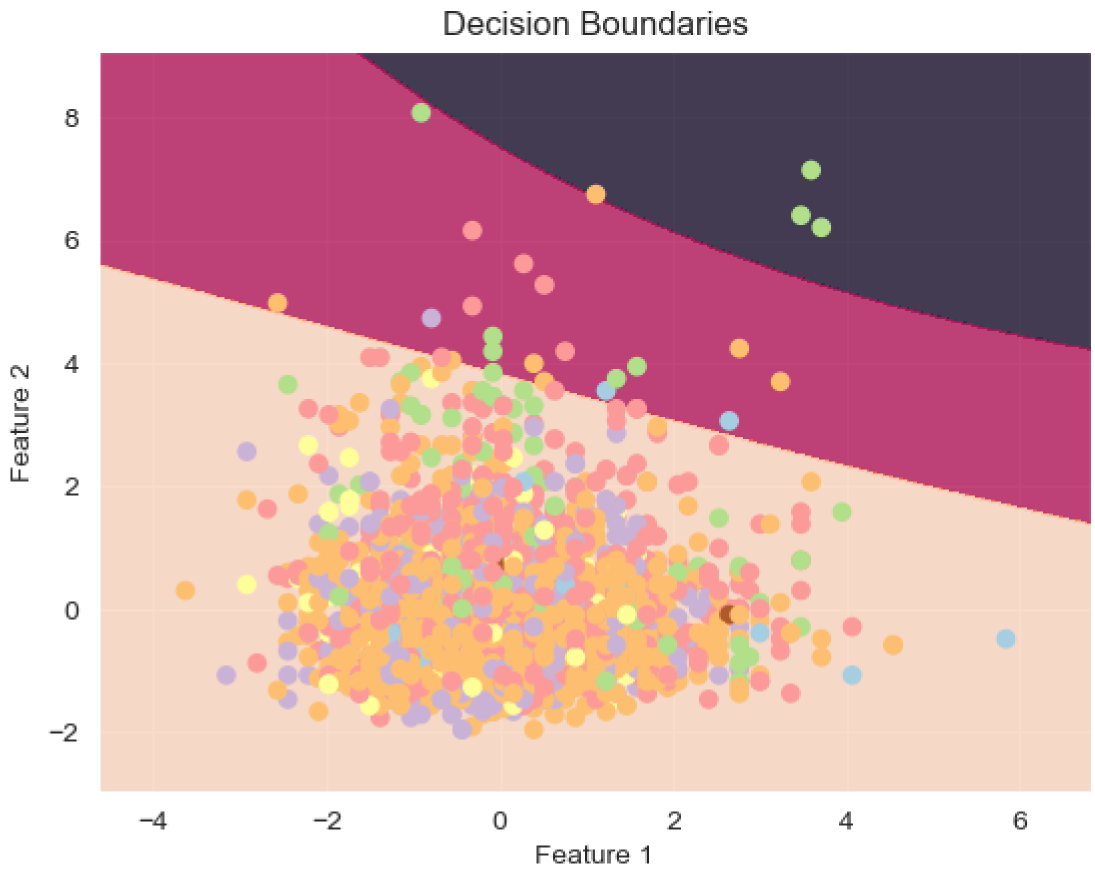
In [119…

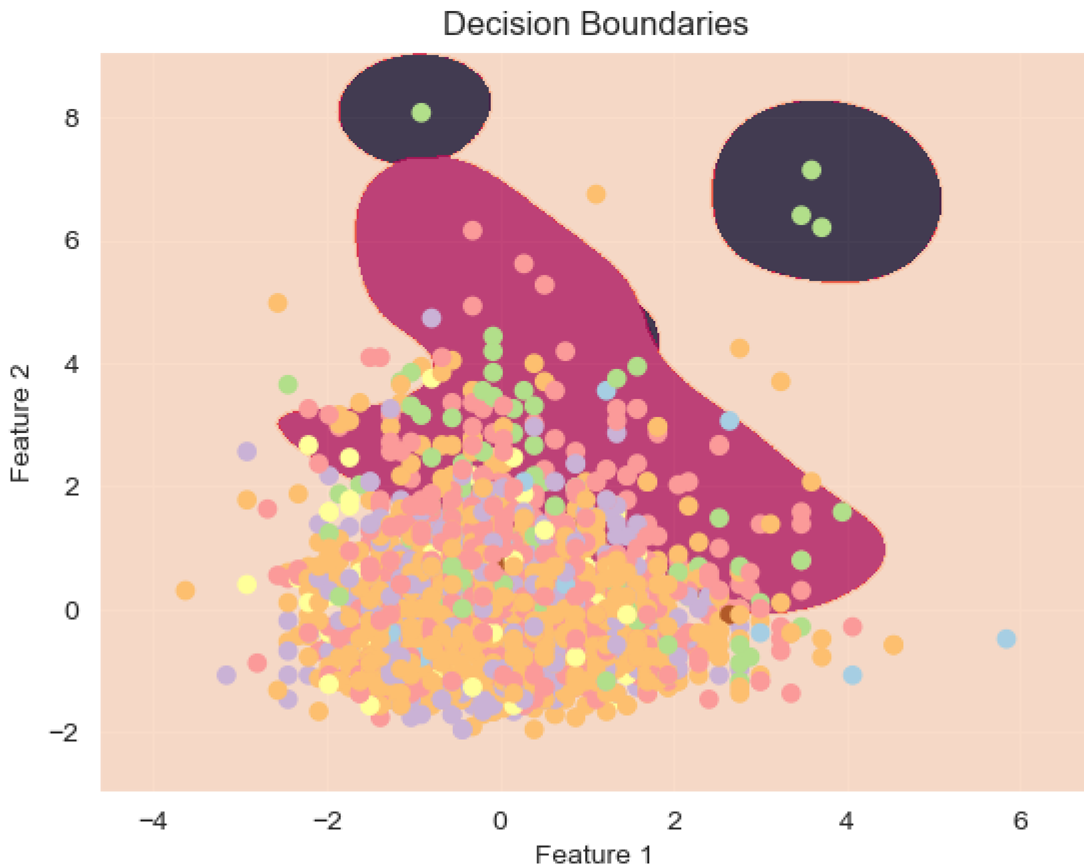
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler

# Load the dataset
url = "https://raw.githubusercontent.com/tejavenkat473/Machine-Learning/main/winequality-white.csv"
df = pd.read_csv(url)

# Separate the features (X) and target variable (y)
X = df.drop('quality', axis=1)
y = df['quality']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

def plot_decision_boundary(model, X, y):
    # Create a meshgrid of feature values
    h = 0.02  # step size in the mesh
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))

    # Make predictions on the meshgrid points
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    # Plot the decision boundaries and data points
    plt.contourf(xx, yy, Z, alpha=0.8)
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.title('Decision Boundaries')
    plt.show()

    # Create an SVM classifier with a linear kernel
svm_linear = SVC(kernel='linear')
svm_linear.fit(X_train[:, :2], y_train)
plot_decision_boundary(svm_linear, X_train[:, :2], y_train)

# Create an SVM classifier with a polynomial kernel
svm_poly = SVC(kernel='poly', degree=3)
svm_poly.fit(X_train[:, :2], y_train)
plot_decision_boundary(svm_poly, X_train[:, :2], y_train)

# Create an SVM classifier with an RBF kernel
svm_rbf = SVC(kernel='rbf')
svm_rbf.fit(X_train[:, :2], y_train)
plot_decision_boundary(svm_rbf, X_train[:, :2], y_train)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Create an SVM classifier with a kernel method
svm_model = SVC(kernel='rbf')
```

```
# Train the SVM model
svm_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = svm_model.predict(X_test)

# Evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```




Decision Boundaries
8
6
4
2
0
−2
Feature 2
−4
−2
0
2
4
6
Feature 1



Accuracy: 0.5612244897959183

In [122…

```
# Calculate confusion matrix
svm_cm = confusion_matrix(y_test, svm_predictions)

# Calculate F1 score
svm_f1 = f1_score(y_test, svm_predictions, average='weighted')

print("SVM Confusion Matrix:")
print(svm_cm)
print("SVM F1 Score:", svm_f1)
```

```
SVM Confusion Matrix:
[[  0   0   1   4   0   0]
 [  0   2  15   8   0   0]
 [  0   3 167 120   1   0]
 [  0   0  82 333  17   0]
 [  0   0   5 139  48   0]
 [  0   0   0  27   8   0]]
SVM F1 Score: 0.5270798963496123
```

In [121…

```
# Generate classification report
svm_classification_report = classification_report(y_test, svm_predictions)

print("SVM Classification Report:")
print(svm_classification_report)
```

```
SVM Classification Report:
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         5
           4       0.40      0.08      0.13        25
           5       0.62      0.57      0.60       291
           6       0.53      0.77      0.63       432
           7       0.65      0.25      0.36       192
           8       0.00      0.00      0.00        35

    accuracy                           0.56       980
   macro avg       0.37      0.28      0.29       980
weighted avg       0.55      0.56      0.53       980
```

```
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parame
ter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parame
ter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parame
ter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [2]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

url = "https://raw.githubusercontent.com/tejavenkat473/Machine-Learning/main/winequality-white.csv"
data = pd.read_csv(url)

# Preprocessing
X = data.drop('quality', axis=1)  # Assuming 'target_variable' is the column to predict
y = data['quality']

# Convert categorical variables to numerical using label encoding
label_encoder = LabelEncoder()
X_encoded = X.apply(label_encoder.fit_transform)

# Create instances of the classification algorithms
rf = RandomForestClassifier()

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print("Random Forest Accuracy:", accuracy_rf)
```

```
Random Forest Accuracy: 0.6989795918367347
```

In [112…

```
# Calculate confusion matrix
rf_cm = confusion_matrix(y_test, y_pred_rf)

# Calculate F1 score
rf_f1 = f1_score(y_test, y_pred_rf, average='weighted')

print("Random Forest Confusion Matrix:")
print(rf_cm)
print("Random Forest F1 Score:", rf_f1)
```

```
Random Forest Confusion Matrix:
[[  0   0   1   4   0   0]
 [  0   5  13   7   0   0]
 [  0   5 207  77   2   0]
 [  0   0  62 345  25   0]
 [  0   0   4  71 113   4]
 [  0   0   1   9   9  16]]
Random Forest F1 Score: 0.6920988476095293
```

In [113…

```
# Generate classification report
rf_classification_report = classification_report(y_test, y_pred_rf)

print("Random Forest Classification Report:")
print(rf_classification_report)
```

```
Random Forest Classification Report:
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         5
           4       0.50      0.20      0.29        25
           5       0.72      0.71      0.72       291
           6       0.67      0.80      0.73       432
           7       0.76      0.59      0.66       192
           8       0.80      0.46      0.58        35

    accuracy                           0.70       980
   macro avg       0.57      0.46      0.50       980
weighted avg       0.70      0.70      0.69       980
```

```
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parame
ter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parame
ter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parame
ter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [114…

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

url = "https://raw.githubusercontent.com/tejavenkat473/Machine-Learning/main/winequality-white.csv"
data = pd.read_csv(url)

# Preprocessing
X = data.drop('quality', axis=1)  # Assuming 'target_variable' is the column to predict
y = data['quality']

# Convert categorical variables to numerical using label encoding
label_encoder = LabelEncoder()
X_encoded = X.apply(label_encoder.fit_transform)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

# Create an instance of the K-nearest Neighbors algorithm
knn = KNeighborsClassifier()

# Train the model
knn.fit(X_train, y_train)

# Make predictions on the test set
y_pred_knn = knn.predict(X_test)

# Evaluate the accuracy of the model
accuracy_knn = accuracy_score(y_test, y_pred_knn)
print("K-nearest Neighbors Accuracy:", accuracy_knn)
```

```
K-nearest Neighbors Accuracy: 0.5530612244897959
```

```
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike othe
r reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts al
ong. In SciPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over whi
ch the statistic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or
False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [116…

```
# Calculate confusion matrix
knn_cm = confusion_matrix(y_test, y_pred_knn)

# Calculate F1 score
knn_f1 = f1_score(y_test, y_pred_knn, average='weighted')

print("KNN Confusion Matrix:")
```

```
print(knn_cm)
print("KNN F1 Score:", knn_f1)
```

```
KNN Confusion Matrix:
[[  0   0   3   2   0   0]
 [  0   4  14   6   1   0]
 [  0   7 169 103  10   2]
 [  0   4  97 280  47   4]
 [  0   1  16  88  85   2]
 [  0   0   2  15  14   4]]
KNN F1 Score: 0.5426100213715772
```

In [117…

```
# Generate classification report
knn_classification_report = classification_report(y_test, y_pred_knn)

print("knn Classification Report:")
print(knn_classification_report)
```

```
knn Classification Report:
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         5
           4       0.25      0.16      0.20        25
           5       0.56      0.58      0.57       291
           6       0.57      0.65      0.60       432
           7       0.54      0.44      0.49       192
           8       0.33      0.11      0.17        35

    accuracy                           0.55       980
   macro avg       0.38      0.32      0.34       980
weighted avg       0.54      0.55      0.54       980
```

```
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parame
ter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parame
ter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parame
ter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [115…

```
# Create a list of classification models
models = [
    ('Logistic Regression', LogisticRegression()),
    ('Decision Tree', DecisionTreeClassifier()),
    ('Random Forest', RandomForestClassifier()),
    ('Support Vector Machine', SVC()),
    ('K-Nearest Neighbors', KNeighborsClassifier())
]

# Iterate over each model and print its accuracy
for name, model in models:
    model.fit(X_train_scaled, y_train)
    predictions = model.predict(X_test_scaled)
    accuracy = accuracy_score(y_test, predictions)
    print(f"{name} Accuracy:", accuracy)
```

```
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs fai
led to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Logistic Regression Accuracy: 0.5306122448979592
Decision Tree Accuracy: 0.6173469387755102
Random Forest Accuracy: 0.7051020408163265
Support Vector Machine Accuracy: 0.5612244897959183
K-Nearest Neighbors Accuracy: 0.5428571428571428
C:\Users\Administrator\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike othe
r reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts al
ong. In SciPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over whi
ch the statistic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or
False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [ ]: