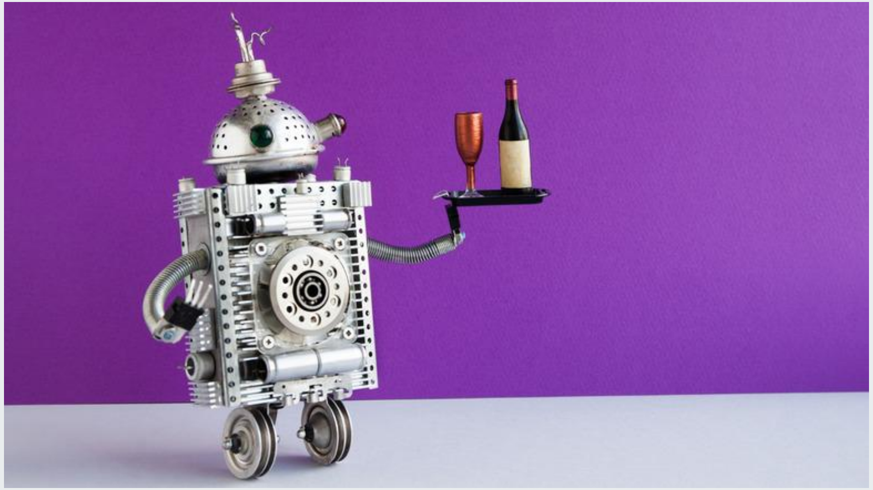
Digital Sommelier

Wine Quality Classification using Machine Learning Techniques



Executive Summary

- **Project Focus:** Predicting wine quality using machine learning models, with a strong emphasis on addressing the challenge of imbalanced classes within the dataset.
- **Data Source:** We used a publicly available wine dataset from the UC Irvine Machine Learning Repository [3].
- **Models Employed:** Support Vector Machine (SVM), Random Forest, XGBoost Gradient Boosting, and an Artificial Neural Network (ANN).
- **Key Findings:** XGBoost Gradient Boosting model emerged as the best performer, offering the highest prediction accuracy while maintaining excellent speed.
- **Impact:** Demonstrated the effectiveness of XGBoost as a tool for predicting wine quality and highlighted the importance of addressing data imbalance.

Problem Statement

- Wine quality prediction is crucial for the wine industry, affecting market pricing, consumer preference, and overall reputation.
- Traditional assessment methods are subjective and inconsistent, heavily reliant on personal taste and susceptible to external influences.
- Current predictive models struggle with handling high-dimensional data, capturing complex, non-linear relationships, resisting overfitting, and dealing with imbalanced classes.
- Goal: Develop an effective, objective prediction model addressing imbalanced classes and improving accuracy across all wine quality levels.

Related Works

Wine quality prediction has been explored using diverse machine learning algorithms:

- Initial approaches: Linear regression and Support Vector Machines [1].
- Recent focus: Ensemble methods, e.g., Random Forests [2], [6].
- Emerging trends: Deep learning methods; performance varies based on dataset size and quality [4], [5].

Reported accuracy of existing models ranges between 60%-75%. Studies using deep learning techniques (ANN, CNN) report higher accuracies > 80% but were computationally expensive [7].

Imbalanced class issue: Various methods proposed for improved classification accuracy.

- Common strategies: Over-sampling minority class, under-sampling majority class, synthetic data generation methods (**SMOTE**) [7].

Proposed Work

Dataset: Publicly available wine dataset from UC Irvine Machine Learning Repository [3], including physicochemical attributes and human-assigned quality score.

Tools: Comprehensive set of Python libraries.

- Data preprocessing and manipulation: Pandas and NumPy.
- Data visualization: Matplotlib and Seaborn.
- Handling imbalanced classes: SMOTE from imbalanced-learn library.
- Machine Learning algorithms: TensorFlow for Artificial Neural Network, Scikit-learn for SVM, Random Forest, and XGBoost Gradient Boosting.

Proposed Work

Primary Tasks in Our Approach

1. Data Preprocessing: Cleaning data, transforming variables, and splitting dataset.
2. Data Exploration and Visualization: Exploratory data analysis to understand variable relationships and dataset structure.
3. Solving Imbalanced Class Problem: Use of SMOTE for synthetic oversampling of minority classes.
4. Feature Scaling: Standardization of features for unbiased model performance.
5. Modeling: Implementation of SVM, Random Forest, XGBoost Gradient Boosting, and Artificial Neural Network.
6. Optimal Model Selection: Performance-based selection of the most efficient and accurate model.

Check point

Data Preprocessing

```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
```

```
[ ] import tensorflow as tf
    from sklearn.preprocessing import StandardScaler, LabelEncoder
```

```
[ ] import time
```

```
red_wine_data = pd.read_csv('/content/gdrive/MyDrive/winequality-red.csv', sep=';')
white_wine_data = pd.read_csv('/content/gdrive/MyDrive/winequality-white.csv', sep=';')
```

```
[ ] red_wine_data['color'] = 'red'
    white_wine_data['color'] = 'white'
```

```
[ ] combined_data = pd.concat([red_wine_data, white_wine_data], ignore_index=True)
```

Check point

```
combined_data.head(10)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	color
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	red
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	red
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	red
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	red
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	red
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4	5	red
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4	5	red
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0	7	red
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5	7	red
9	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5	5	red

features

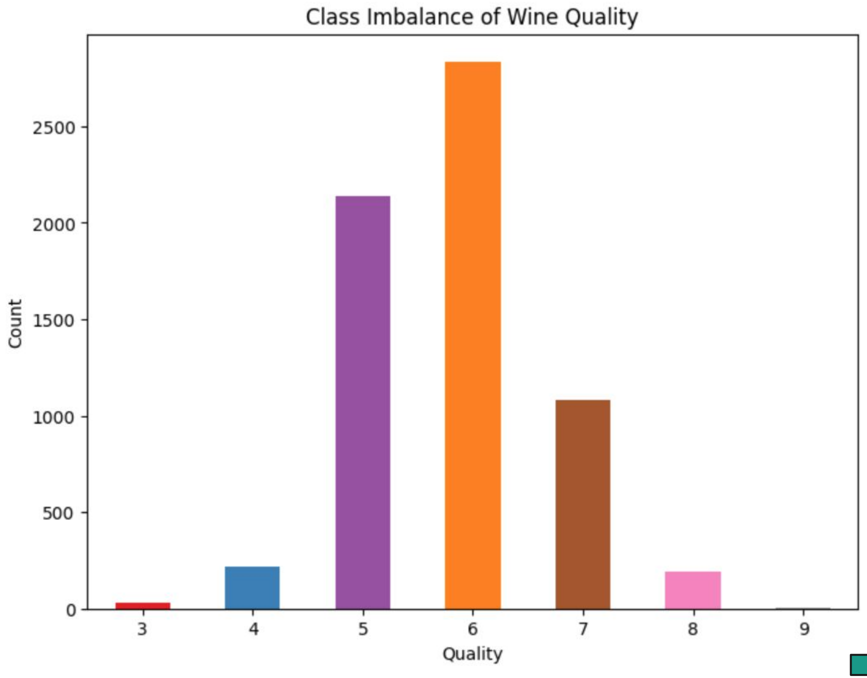
labels

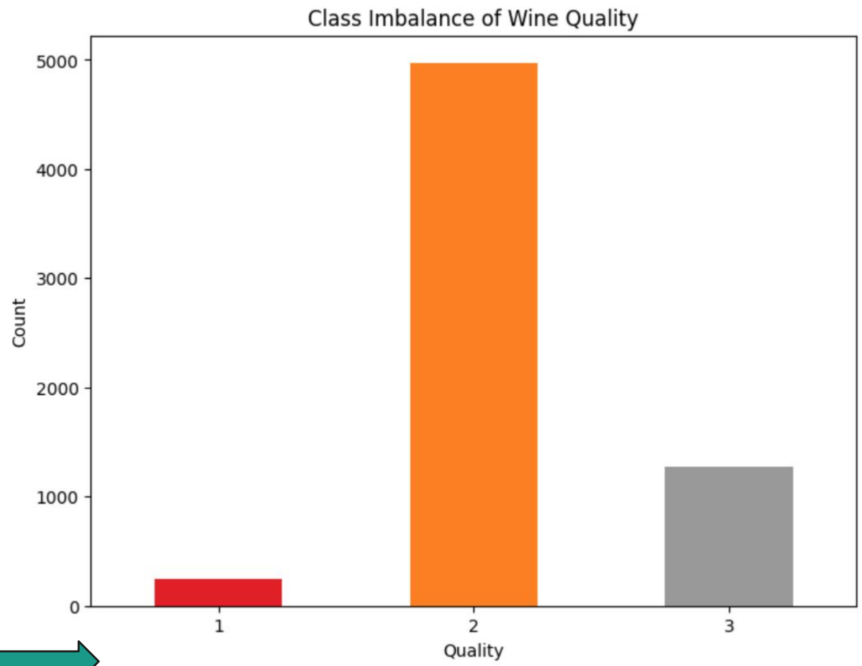
```
num_entries = combined_data.shape[0]
print("Number of entries:", num_entries)
```

```
Number of entries: 6497
```

Check point

Data Transformation





Dimensionality Reduction

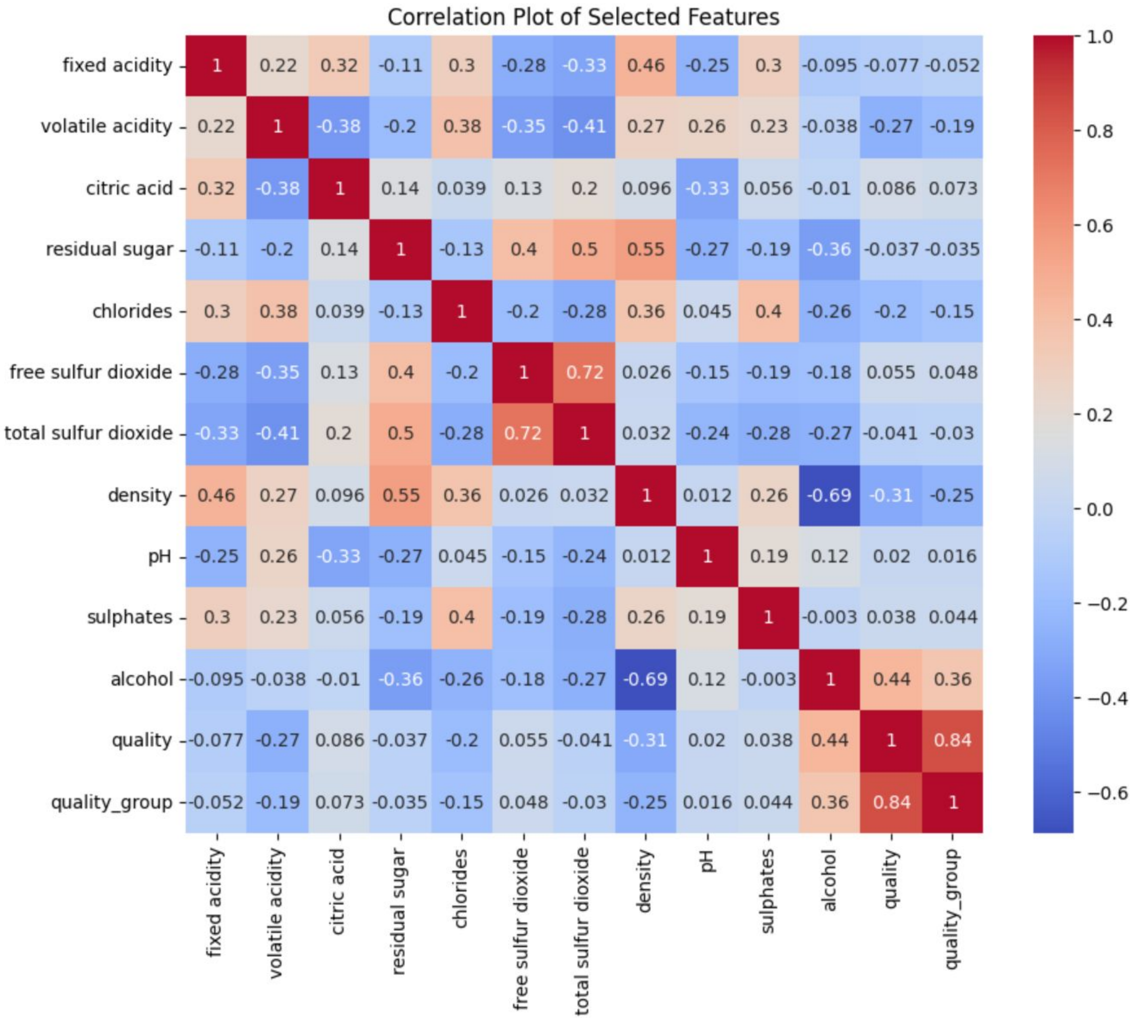
[3, 4] → [1]

[5, 6] → [2]

[7, 8, 9] → [3]

Check point

Data Exploration

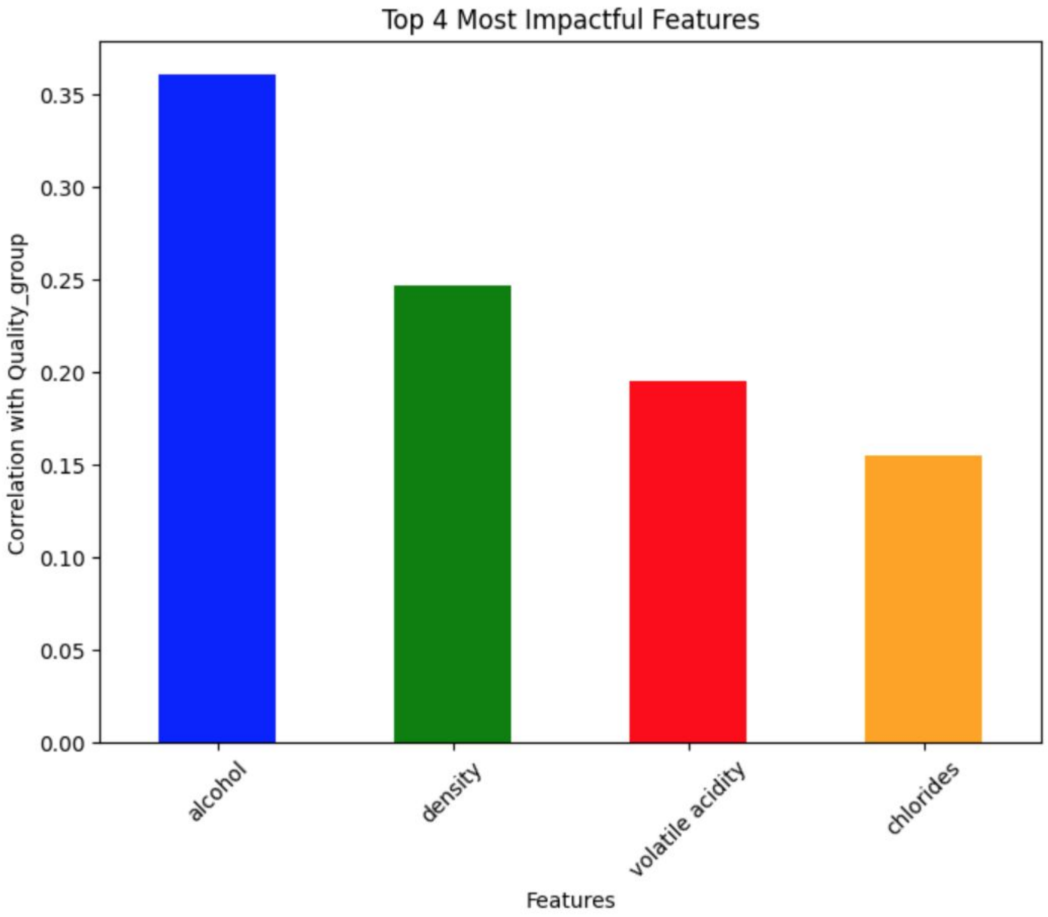


Check point

Data Exploration

```
# Print the most impactful features
print(correlations)
```

```
quality_group           1.000000
alcohol                 0.360580
density                 0.246116
volatile acidity        0.194906
chlorides               0.154945
citric acid             0.073082
fixed acidity           0.052052
free sulfur dioxide     0.048382
sulphates               0.043719
residual sugar          0.035250
total sulfur dioxide    0.029793
pH                      0.016064
```



Check point

Data Splitting, Feature Scaling and Class Imbalance

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE
```

```
# 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=1)

# Apply feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Apply SMOTE to oversample the minority class
smote = SMOTE(random_state=1)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train_scaled, y_train)
```

Check point

Model Training

```
# Create an SVM classifier
svm_classifier = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=1)

# Start the timer
start_time = time.time()

# Train the SVM classifier
svm_classifier.fit(X_train_resampled, y_train_resampled)

# Stop the timer and calculate the runtime
runtime = time.time() - start_time
```

```
# Create an XGBoost classifier
xgb_classifier = xgb.XGBClassifier(random_state=1)

# Start the timer
start_time = time.time()

# Train the XGBoost classifier
xgb_classifier.fit(X_train_resampled, y_train_resampled)

# Stop the timer and calculate the runtime
runtime = time.time() - start_time
```

```
# Create the Random Forest classifier
rf_classifier = RandomForestClassifier(random_state=1)

# Start the timer
start_time = time.time()

# Train the classifier on the resampled training data
rf_classifier.fit(X_train_resampled, y_train_resampled)

# Stop the timer and calculate the runtime
runtime = time.time() - start_time
```

```
# Build the ANN model
model = Sequential()
model.add(Dense(64, activation='relu', input_dim=X_train_resampled.shape[1]))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))

# Compile the model
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Start the timer
start_time = time.time()

# Train the model
model.fit(X_train_resampled, y_train_resampled, epochs=100, batch_size=32, verbose=1)

# Stop the timer and calculate the runtime
runtime = time.time() - start_time
```

Evaluation

Evaluation Metrics for Model Assessment:

- **Primary Metric: Accuracy** – Represents the proportion of correct predictions made by the model.
    - F1-Score: Harmonic mean of Precision and Recall, providing a balance between them.
    - Precision and Recall: Detailed assessment of the model's performance across different classes.
- **Training Time:** Computational efficiency is crucial, especially for practical deployment. We will record and compare the total training time for each model.

Our aim is to ensure a comprehensive and efficient model selection process, offering the highest quality prediction for real-world applications.

Key Findings

Evaluation

SVM

```
# Make predictions on the test set
y_pred = svm_classifier.predict(X_test_scaled)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Classification Report:\n", report)
print("Runtime:", round(runtime, 2), "seconds")
```

```
Accuracy: 0.63
Classification Report:
              precision    recall  f1-score   support

           1       0.14      0.54      0.22        52
           2       0.91      0.60      0.72      1012
           3       0.43      0.79      0.55       236

    accuracy                           0.63      1300
   macro avg       0.49      0.64      0.50      1300
weighted avg       0.79      0.63      0.67      1300

Runtime: 5.58 seconds
```

Random Forest

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

# Calculate accuracy and generate classification repor
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

# Print accuracy and classification report
print("Accuracy:", round(accuracy,4))
print("Classification Report:\n", report)
print("Runtime:", round(runtime, 2), "seconds")
```

```
Accuracy: 0.8162
Classification Report:
              precision    recall  f1-score   support

           1       0.32      0.40      0.36        52
           2       0.90      0.86      0.88      1012
           3       0.63      0.72      0.67       236

    accuracy                           0.82      1300
   macro avg       0.62      0.66      0.64      1300
weighted avg       0.83      0.82      0.82      1300

Runtime: 3.69 seconds
```

Key Findings

Evaluation

Gradient Boosting XGBoost

```
# Make predictions on the test set
y_pred = xgb_classifier.predict(X_test_scaled)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print("Accuracy:", round(accuracy,4))
print("Classification Report:\n", report)
print("Runtime:", round(runtime, 2), "seconds")
```

```
Accuracy: 0.8308
Classification Report:
              precision    recall  f1-score   support

           0       0.40      0.27      0.32        52
           1       0.89      0.89      0.89      1012
           2       0.65      0.68      0.67       236

    accuracy                           0.83      1300
   macro avg       0.65      0.62      0.63      1300
weighted avg       0.83      0.83      0.83      1300

Runtime: 6.21 seconds
```

Artificial Neural Network

```
# Evaluate the model
y_pred_prob = model.predict(X_test)
y_pred = y_pred_prob.argmax(axis=1)

accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print("Accuracy:", round(accuracy,4))
print("Classification Report:\n", report)
print("Runtime:", round(runtime, 2), "seconds")
```

```
Accuracy: 0.8062
Classification Report:
              precision    recall  f1-score   support

           1       0.28      0.29      0.28        52
           2       0.88      0.87      0.88      1012
           3       0.61      0.66      0.64       236

    accuracy                           0.81      1300
   macro avg       0.59      0.61      0.60      1300
weighted avg       0.81      0.81      0.81      1300

Runtime: 112.39 seconds
```

Model Selection

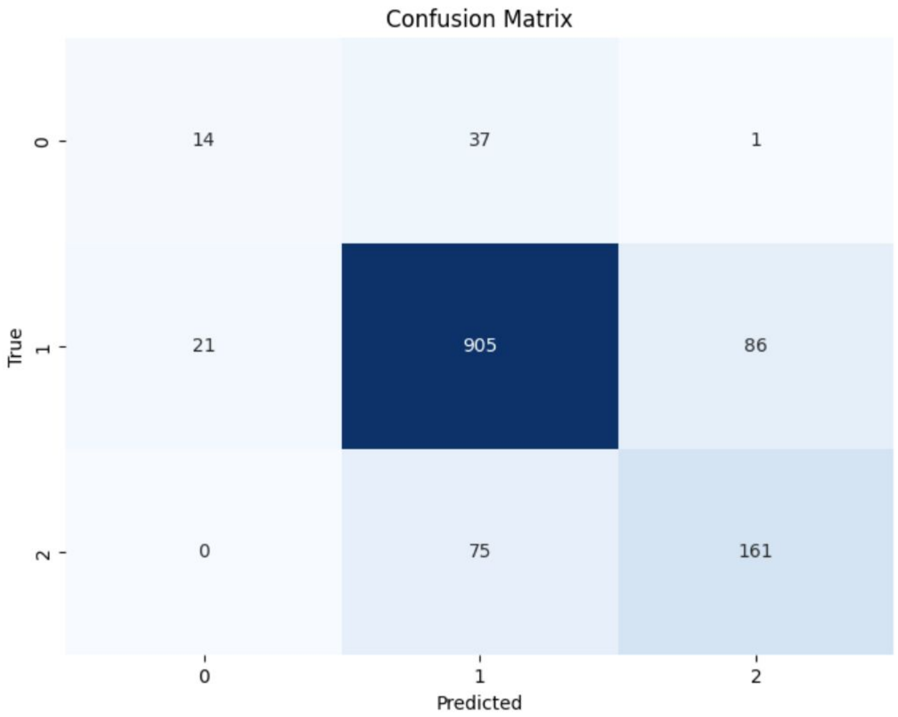
Model	Accuracy	Efficiency (Runtime)	
SVM	0.63	5.58 s	→ Eliminated
Random Forest	0.8162	3.69 s	
Gradient Boosting	0.8308	6.21 s	
Artificial Neural Network	0.8062	112.39 s	→ Eliminated

Model Selection

Random Forest	Gradient Boosting
Train and run in parallel	Train and run sequentially
Can be faster	More accurate
Resistant to noise	Prone to overfitting

→ For our project, let's prioritize accuracy and choose XGBoost

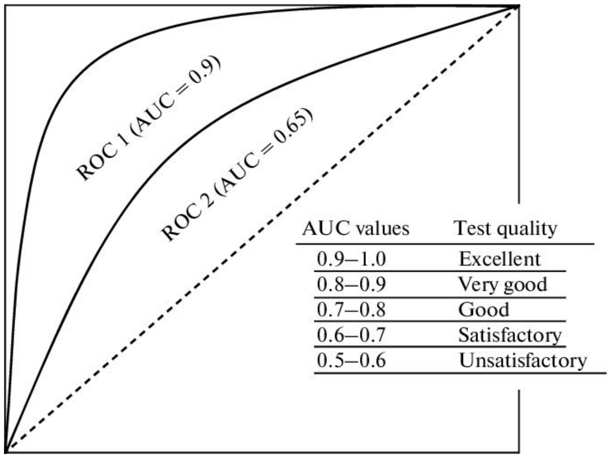
Further Validation for XGBoost

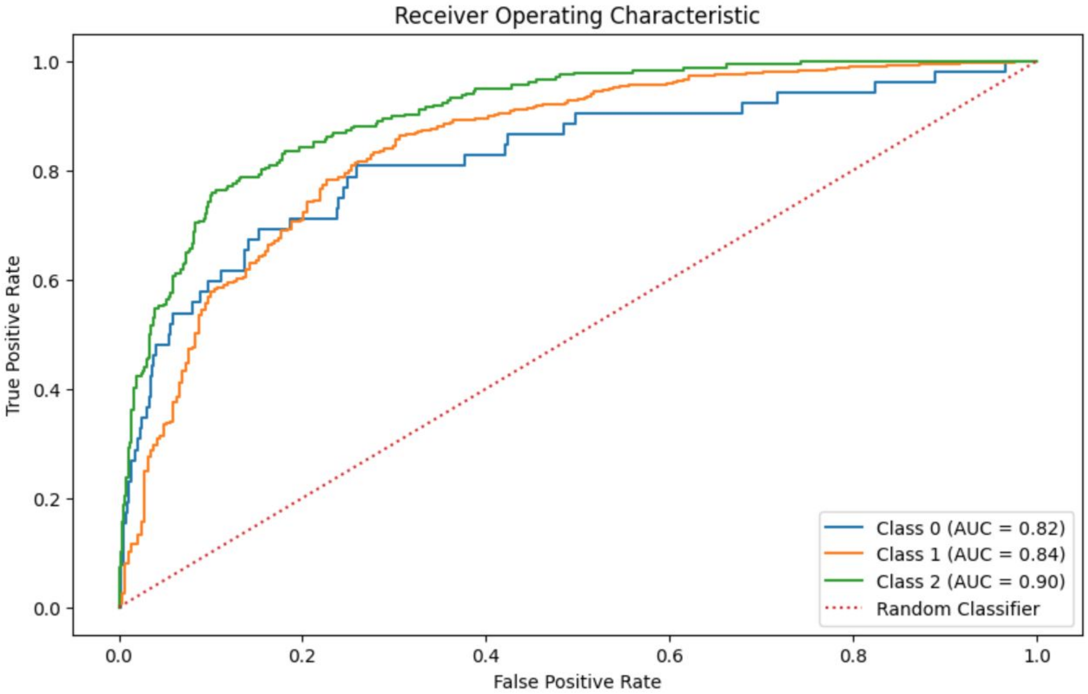



```
Accuracy: 0.8308
Classification Report:
              precision    recall  f1-score   support

           0       0.40      0.27      0.32        52
           1       0.89      0.89      0.89      1012
           2       0.65      0.68      0.67       236

    accuracy                           0.83      1300
   macro avg       0.65      0.62      0.63      1300
weighted avg       0.83      0.83      0.83      1300

Runtime: 6.21 seconds
```

Key Findings

Further Validation for XGBoost






```
Class: 0, ROC AUC Score: 0.8223157051282051
Class: 1, ROC AUC Score: 0.8436264822134387
Class: 2, ROC AUC Score: 0.9042309162737352
```

Timeline

Week 1	Week 2	Week 3	Week 4
Data Acquisition, Preprocessing, and Exploration	Handling Imbalanced Classes, Feature Scaling and Model Implementation	Model Evaluation and Selection	Finalizing the Report

References

[1] Avula, 2019. *Predicting Red Wine Quality – Using Machine Learning Model*. Medium. https://medium.com/analytics-vidhya/predicting-red-wine-quality-using-machine-learning-model-34e2b1b8d498

[2] Chhikara et al., 2023. *Wine Quality Prediction Using Machine Learning Technique.* In *Smart Trends in Computing and Communications*. Springer, Singapore. DOI: https://doi.org/10.1007/978-981-99-0769-4_14

[3] Cortez et al., 2009. *Wine Quality*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C56S3T

[4] Di and Yang, 2022. *Prediction of Red Wine Quality Using One-dimensional Convolutional Neural Networks*. arXiv preprint arXiv:2208.14008.

[5] El-din, 2021. *Red Wine Quality Classifier Using a Neural Network*. Kaggle. https://www.kaggle.com/code/omaryassersalaheldin/red-wine-quality-classifier-using-a-neural-network/comments

[6] Fairbrother et al., 2021. Predicting *Wine Quality from Physicochemical Features*. University of British Columbia. https://ubc-mds.github.io/DSCI_522_group09_Wine_Quality_Predictor

[7] Kothawade, 2021. Wine Quality Prediction Model Using Machine Learning Techniques. University of Skovde.