Wine Quality Prediction :
Classification Using 3 AI Models

Written by:
- Moreno Cello Rhythm | 103012340256
- Kavin Hibrizy P E | 103012340295
- Parth Tuteja | 2024019802
- Kritika Verma | 2024019811

Github Project Link: https://github.com/parth2teja23/glow2025_final_project

table of *Contents*

Introduction

Dataset Overview

- **Total Samples:**
  - 1,599 red wine records from the Portuguese Vinho Verde region
- **Input Features (11 variables):**
  - Fixed & volatile acidity
  - Citric acid, residual sugar
  - Chlorides, sulphates
  - Free & total sulfur dioxide
  - pH, density, alcohol content
- **Target Variable – Wine Quality:**
  - Original score: 0 to 10 based on sensory evaluation
  - Converted to 2 classes for classification:
    - 'bad' for scores ≤ 6.5
    - 'good' for scores > 6.5

- The class distribution is **imbalanced**, with far more samples in the "Bad" class than "Good", which poses a challenge for classification models.

data *Preparation*

Exploratory Data Analysis

Dataset & Problem Framing

Dataset Overview

- Source: UCI Wine Quality (Red Wine subset)
- Total records: 1,599 samples
- Features: 11 numerical attributes (e.g., acidity, sugar, alcohol)
- Target: Original quality score (0–10), converted to binary label:
  - 0 = Bad (score ≤ 6.5)
  - 1 = Good (score > 6.5)

Why Binary Classification?

- Original quality scores are imbalanced and subjective
- Binarizing simplifies modeling and evaluation

Exploratory Data Analysis

Data Preprocessing & Cleaning

- Cleaning Steps
  - Removed 240 duplicate rows
  - Outliers removed using IQR method (threshold = 1.15 × IQR)
  - Normalized features using StandardScaler (Z-score)
- Class Distribution After Cleaning
  - Balanced ~binary classes for good/bad wines
  - Ensures fairer evaluation for classifiers
- Train/Test Split
  - 80% training, 20% test set
  - Saved as .pkl using joblib for consistent use across models

Exploratory Data Analysis

Modeling & Evaluation Preview

- Models Applied
  - Decision Tree
  - Naive Bayes
  - K-Nearest Neighbors (KNN)
- Evaluation Metrics
  - Accuracy, Precision, Recall, F1-score
  - Confusion Matrix

- PCA was not applied due to low dimensionality
- Feature scaling crucial for KNN; applied universally for consistency

Tree decision

Model Setup

Target: label (0 = Not Good, 1 = Good)

Removed quality to prevent data leakage

80/20 train-test split using train_test_split

Initial Decision Tree

Model: DecisionTreeClassifier with default parameters

Accuracy: 73.9%

Confusion Matrix:

True Positives (Good): 104

True Negatives (Not Good): 97

Key Features: alcohol, volatile acidity, total sulfur dioxide

Tree plotted to top 3 levels

Cost-Complexity Pruning

- Used ccp_alpha to reduce overfitting
- Tested multiple alpha values using cost_complexity_pruning_path
- Optimal alpha: 0.00194
- Post-pruning Accuracy: ✅ 76.8%

Model became more generalized, with reduced depth and better F1-scores

naives Bayes

Naive Bayes Classifier

Goal: Classify wine quality based on probabilistic modeling of feature distributions.

What is Naive Bayes?

- A supervised ML algorithm based on Bayes' Theorem.
- Assumes independence between input features.
- Computes posterior probability for each class and chooses the highest.

How We Used It:

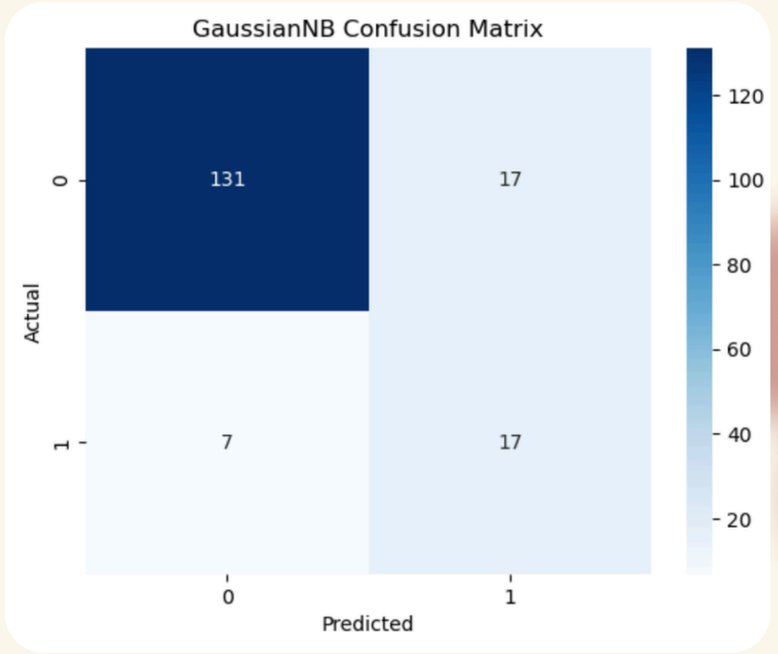
- Implemented using GaussianNB from scikit-learn.
- Trained on wine quality dataset with 11 features.
- Assumes each feature follows a normal (Gaussian) distribution.
- Performs well despite strong independence assumptions.

Naive Bayes Evaluation (GaussianNB)

Classification Report:

	precision	recall	f1-score	support
0	0.95	0.89	0.92	148
1	0.50	0.71	0.59	24
accuracy			0.86	172
macro avg	0.72	0.80	0.75	172
weighted avg	0.89	0.86	0.87	172

Confusion Matrix:



Performance for the positive class (class 1)

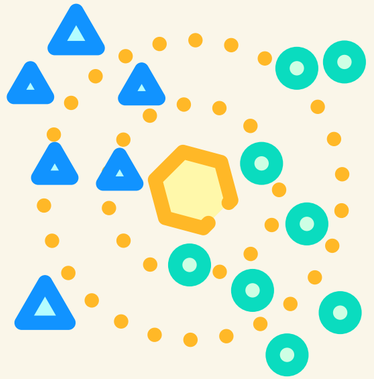
Accuracy : 0.86
Precision : 0.50
Recall : 0.71
F1-score : 0.59

Insights

- Naive Bayes achieves good recall (0.71) for the minority 'good' class, meaning it correctly identifies many of the good wines.
- Precision for the good class is lower (0.50), indicating some false positives.
- Model performs very well on the majority 'bad' class, which is expected given Naive Bayes' strength on well-separated, independent features.
- Overall, the F1-score for the good class is decent (0.59) given the class imbalance and simplicity of the model.

K nearest neighbour

KNN Classifier



Goal: Classify wine quality (Good or Bad) based on the proximity of samples using Euclidean distance in feature space.

What is KNN?

- A non-parametric, instance-based machine learning algorithm
- Classifies a data point based on the majority label among its K nearest neighbors
- Uses a distance metric (e.g., Euclidean) to find closeness between data points
- Simple, interpretable, and effective for small to medium datasets

How We Used It:

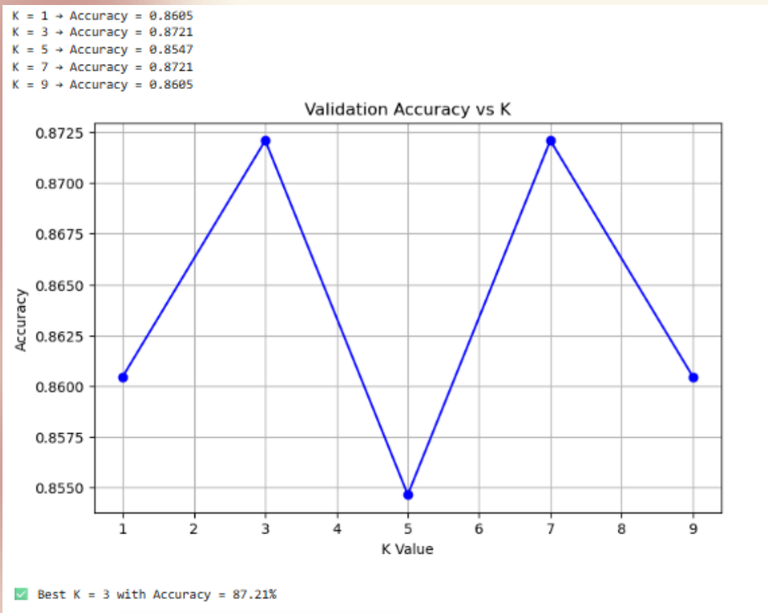
- Implemented using KNeighborsClassifier from scikit-learn
- Trained on a preprocessed dataset with 11 numerical features
- Tested various K values (1, 3, 5, 7, 9) to optimize performance
- Used Euclidean distance and visualized results with PCA-based decision boundaries

Train KNN Classifier with Euclidean Distance

```
🔢 Manual Euclidean Distance Calculation (first test sample):
Closest training sample index: 290, Label: 0
```

- Started with a basic KNN model using K = 1
- Computed Euclidean distance between the first test sample and all training samples
- Identified the closest training sample and its label
- Simulates how KNN internally classifies data

Experiment with Different K Values



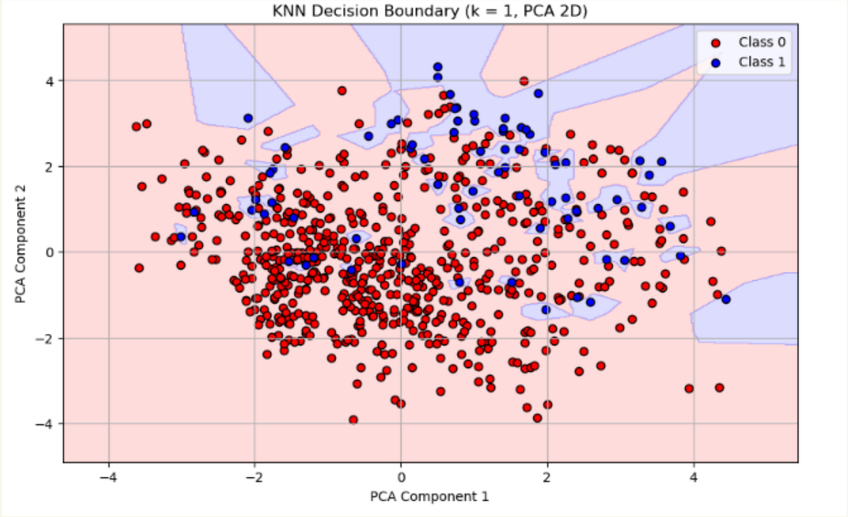
- Tried different K values: 1, 3, 5, 7, 9
- Used accuracy as the validation metric
- Plotted results: Validation Accuracy vs K
- Best accuracy:
- K = 3 → 87.21%
- Insight: Very low or high K values reduce performance

Final Evaluation with Confusion Matrix

```
=== Classification Report ===
              precision    recall  f1-score   support

           0       0.90      0.96      0.93       148
           1       0.57      0.33      0.42        24

    accuracy                           0.87       172
   macro avg       0.74      0.65      0.67       172
weighted avg       0.85      0.87      0.86       172

=== Confusion Matrix ===
[[142   6]
 [ 16   8]]
```

- Final model trained using best K (K = 3)
- Evaluation metrics:
  - Accuracy: 87.2%
- Class 0 (Bad Wine):
  - Precision = 0.90, Recall = 0.96
- Class 1 (Good Wine):
  - Precision = 0.57, Recall = 0.33
- 📊 Confusion Matrix shows bias toward the majority class due to class imbalance

Visualize Decision Boundaries (2D PCA Projection)



model Comparations

	Accuracy	Good			Bad		
		Precision	Recall	F1-Score	Precision	Recall	F1-Score
Naïve Bayes	86	5	71	59	95	89	92
KNN	87	57	33	42	9	96	93
Decision Tree	88	6	38	46	9	96	93

- **Conclusion**
  - Accuracy: Decision Tree (88%) outperforms KNN (87%) and Naive Bayes (86%).
  - Bad Class: Naive Bayes has the highest precision (0.95), while KNN and Decision Tree excel in recall (0.96).
  - Good Class: Decision Tree has the best precision (0.60), but KNN and Naive Bayes have lower recall, especially KNN (0.33).
  - F1-Score: KNN and Decision Tree lead for the Bad class (0.93), while Decision Tree has the best F1 for the Good class (0.46), and Naive Bayes is better for Good class precision (0.59).

- **Recommendation**
  - KNN and Decision Tree provide balanced performance across both classes and are suitable when high recall for Bad class is critical.
  - Naive Bayes, while not as strong for the Good class, is useful for situations where precision for the Bad class is prioritized, especially when a high false positive rate for Good wines is acceptable.

Thank You