wine-quality

November 22, 2025

1 Predicting Wine Quality: A Binary Classification Approach Using Physicochemical Properties

Authors: Aidan Hew, Karan Bains, Shuhang Li

1.1 Summary

This analysis investigates whether physicochemical properties (eg. alcohol content, volatile acidity, and sulphates) can reliably predict wine quality using classification. Using a dataset of 1,599 red Portuguese "Vinho Verde" wines, we developed models to distinguish between high-quality wines (rated 7 or higher) and lower-quality wines (rated below 7). The analysis employed logistic regression, decision trees, and random forest classifiers. Results indicate that alcohol content, volatile acidity, and sulphates are the strongest predictors of wine quality, with the random forest model achieving 87% accuracy and an AUC of 0.91. These findings suggest that automated quality assessment based on chemical properties is feasible and could support wine production quality control processes.

1.2 Introduction

Wine quality assessment is traditionally performed by human experts through sensory evaluation, a process that is subjective, time-consuming, and requires specialized training. The ability to predict wine quality based on objective physicochemical measurements can potentially enable faster feedback to producers and more consistent quality standards.

Wine quality is influenced by numerous chemical properties resulting from grape varieties, fermentation processes, and aging conditions. Key factors include acidity levels (which affect taste balance), alcohol content (which influences body and preservation), sulfur dioxide levels (used as preservatives), and various other compounds that contribute to flavor, aroma, and stability.

1.2.1 Research Question

Can we accurately predict whether a red wine is of high quality based solely on its physicochemical properties?

We aim to build a binary classification model that distinguishes between high-quality wines (rated 7 out of 10) and lower-quality wines (rated <7) using features such as acidity, alcohol content, sulfur dioxide levels, and other measurable chemical properties.

1.2.2 Dataset Description

This analysis uses the Red Wine Quality dataset, which is publicly available from the UCI Machine Learning Repository. The dataset contains 1,599 samples of red Portuguese "Vinho Verde" wine, collected between 2004-2007. Each wine sample includes 11 physicochemical features and a quality score.

Features: - Fixed acidity (g/L): Non-volatile acids (tartaric acid) - Volatile acidity (g/L): Acetic acid content (high levels indicate vinegar taste) - Citric acid (g/L): Adds freshness and flavor - Residual sugar (g/L): Sugar remaining after fermentation - Chlorides (g/L): Salt content - Free sulfur dioxide (mg/L): Prevents microbial growth - Total sulfur dioxide (mg/L): Total SO (bound and free) - Density (g/cm³): Wine density - pH: Acidity level (scale 0-14) - Sulphates (g/L): Wine additive contributing to SO - Alcohol (% volume): Alcohol content

Target Variable: - Quality: Expert ratings on a scale from 0 (very bad) to 10 (excellent)

1.3 Methods & Results

We trained three classification models; Logistic Regression, Decision Tree, and Random Forest, to predict high versus low quality wines using 11 continuous features. The data was split 80/20 with stratification, with all features being standardised before model training. Due to the under-representation of high-quality wines in our data, we trained all models with balanced class weights to compensate for this limitation. Model performance was then evaluated using accuracy, precision, recall, F1 score, and ROC AUC, with the best performing model undergoing 5-fold cross-validation.

The Random Forest Classifier performed the strongest across all metrics, achieving 88% test accuracy as well as an ROC AUC of 0.92. Cross-validation confirmed this result, with a mean accuracy of 0.885 and relatively low variance across folds. This suggests that the model will generalise well, and indicates potential non-linear relationships in the data due to its superiority over the Logistic Regression and Decision Tree models.

[8]:
```
import pandas as pd
import numpy as np
import altair as alt
from sklearn.model_selection import train_test_split, cross_val_score,␣
 ↪GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    confusion_matrix, classification_report, roc_curve, auc, roc_auc_score)
```

1.3.1 Exploratory Data Analysis

[9]:
```
df = pd.read_csv("data/winequality-red.csv", sep=';')

# Create binary target variable for quality>=7 and quality<7
df['quality_binary'] = (df['quality'] >= 7)
```

```
# Separate features and target
feature_columns = ['fixed acidity', 'volatile acidity', 'citric acid',␣
 ↪'residual sugar',
                   'chlorides', 'free sulfur dioxide', 'total sulfur dioxide',␣
 ↪'density',
                   'pH', 'sulphates', 'alcohol']

X = df[feature_columns]
y = df['quality_binary']

# Summary statistics of all features
print("Summary statistics for physicochemical features:")
df[feature_columns].describe().round(3)
```

```
Summary statistics for physicochemical features:
```

[9]:

	fixed acidity	volatile acidity	citric acid	residual sugar	\
count	1599.000	1599.000	1599.000	1599.000	
mean	8.320	0.528	0.271	2.539	
std	1.741	0.179	0.195	1.410	
min	4.600	0.120	0.000	0.900	
25%	7.100	0.390	0.090	1.900	
50%	7.900	0.520	0.260	2.200	
75%	9.200	0.640	0.420	2.600	
max	15.900	1.580	1.000	15.500	

	chlorides	free sulfur dioxide	total sulfur dioxide	density	\
count	1599.000	1599.000	1599.000	1599.000	
mean	0.087	15.875	46.468	0.997	
std	0.047	10.460	32.895	0.002	
min	0.012	1.000	6.000	0.990	
25%	0.070	7.000	22.000	0.996	
50%	0.079	14.000	38.000	0.997	
75%	0.090	21.000	62.000	0.998	
max	0.611	72.000	289.000	1.004	

	pH	sulphates	alcohol
count	1599.000	1599.000	1599.000
mean	3.311	0.658	10.423
std	0.154	0.170	1.066
min	2.740	0.330	8.400
25%	3.210	0.550	9.500
50%	3.310	0.620	10.200
75%	3.400	0.730	11.100
max	4.010	2.000	14.900

[10]:
```
# Create correlation data frame in long format
corr_df = pd.concat([X, y], axis=1).corr('spearman').stack().reset_index()
corr_df.columns = ['feature_1', 'feature_2', 'correlation']
corr_df.loc[corr_df['correlation'] == 1, 'correlation'] = 0 # Remove diagonal

# Create correlation heatmap
corr_heatmap = alt.Chart(
    corr_df,
    title = 'Wine Quality Correlation Heatmap').mark_rect().encode(
    x = alt.X('feature_1').title('Feature 1'),
    y = alt.Y('feature_2').title('Feature 2'),
    color = alt.Color('correlation').scale(scheme = 'blueorange', domain = (-1,␣
 ↪1)).title('Correlation'),
    tooltip = alt.Tooltip('correlation:Q', format = '.2f')
)

# Display correlation heatmap
corr_heatmap
```

[10]:
```
alt.Chart(…)
```

[11]:
```
# Isolate target and correlates
dist_feats = ['quality_binary', 'alcohol', 'sulphates', 'volatile acidity']

# Create density data frame
dist_df = pd.concat([X, y], axis=1)
dist_df = dist_df[dist_feats]

# Replace boolean with descriptive strings
dist_df['quality_binary'] = dist_df['quality_binary'].map({
    True: 'High Quality Wine',
    False: 'Low Quality Wine'
})

# Create overlaid histograms for each correlated feature
feature_hists = alt.Chart(dist_df).mark_bar(opacity = 0.5).encode(
    x = alt.X(alt.repeat('column')).type('quantitative').bin(maxbins = 25).
 ↪axis(format = '.1f'),
    y = alt.Y('count()').stack(False),
    color = alt.Color('quality_binary:N').title('Wine Quality')
).properties(
    width = 250,
    height = 200,
).repeat(
    column = ['alcohol', 'sulphates', 'volatile acidity']
).resolve_scale(
    y = 'shared'
```

```
)

# Display histograms
feature_hists
```

[11]:
```
alt.RepeatChart(…)
```

Our EDA showed only a few features that displayed meaningful relationships with the quality of wine. In particular, alcohol, sulphates, and volatile acidity exhibited the strongest correlations with our target. High-quality wines were associated with higher alcohol and sulphate levels but lower volatile acidity. The histograms helped to demonstrate this separation in distribution of high and low-quality wines for these features. Finally, the dataset contained a notable imbalance, with relatively few high-quality wines, implying the need for class balancing in the later modelling stages.

[12]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=2025, stratify=y
)

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

1.3.2 Model Development and Training

We trained three different classification algorithms to compare their performance:

1. **Logistic Regression**: A linear model that estimates the probability of class membership using a logistic function.
2. **Decision Tree**: A non-linear model that recursively partitions the feature space based on feature thresholds. We limit the maximum depth and require minimum samples per leaf to prevent overfitting.
3. **Random Forest**: An ensemble method that combines multiple decision trees through bootstrap aggregation (bagging). This typically provides better generalization than a single decision tree.

All models use class weighting (balanced) to account for the imbalanced class distribution, giving more importance to the minority class (high-quality wines).

[13]:
```
# Initialize models with class balancing
models = {
    'Logistic Regression': LogisticRegression(
        random_state=123,
        max_iter=1000,
        class_weight='balanced'
    ),
    'Decision Tree': DecisionTreeClassifier(
        random_state=123,
```

```
        max_depth=10,
        min_samples_split=20,
        min_samples_leaf=10,
        class_weight='balanced'
    ),
    'Random Forest': RandomForestClassifier(
        n_estimators=100,
        random_state=123,
        max_depth=15,
        min_samples_split=10,
        min_samples_leaf=5,
        class_weight='balanced'
    )
}

# Train models and store results
trained_models = {}
results = []

for name, model in models.items():
    print(f"\nTraining {name}...")

    # Train the model
    model.fit(X_train_scaled, y_train)
    trained_models[name] = model

    # Make predictions
    y_train_pred = model.predict(X_train_scaled)
    y_test_pred = model.predict(X_test_scaled)
    y_test_proba = model.predict_proba(X_test_scaled)[:, 1]

    # Calculate metrics
    train_acc = accuracy_score(y_train, y_train_pred)
    test_acc = accuracy_score(y_test, y_test_pred)
    precision = precision_score(y_test, y_test_pred)
    recall = recall_score(y_test, y_test_pred)
    f1 = f1_score(y_test, y_test_pred)
    roc_auc = roc_auc_score(y_test, y_test_proba)

    # Store results
    results.append({
        'Model': name,
        'Train Accuracy': train_acc,
        'Test Accuracy': test_acc,
        'Precision': precision,
        'Recall': recall,
        'F1 Score': f1,
```

```
        'ROC AUC': roc_auc
    })

    print(f"  Train Accuracy: {train_acc:.4f}")
    print(f"  Test Accuracy: {test_acc:.4f}")
    print(f"  Precision: {precision:.4f}")
    print(f"  Recall: {recall:.4f}")
    print(f"  F1 Score: {f1:.4f}")
    print(f"  ROC AUC: {roc_auc:.4f}")
```

```
Training Logistic Regression…
  Train Accuracy: 0.7928
  Test Accuracy: 0.7594
  Precision: 0.3455
  Recall: 0.8837
  F1 Score: 0.4967
  ROC AUC: 0.8797

Training Decision Tree…
  Train Accuracy: 0.8757
  Test Accuracy: 0.7656
  Precision: 0.3298
  Recall: 0.7209
  F1 Score: 0.4526
  ROC AUC: 0.7845

Training Random Forest…
  Train Accuracy: 0.9578
  Test Accuracy: 0.8844
  Precision: 0.5577
  Recall: 0.6744
  F1 Score: 0.6105
  ROC AUC: 0.9212
```

[14]:
```
# Perform 5-fold cross-validation on the best model
best_model = trained_models['Random Forest']
cv_scores = cross_val_score(best_model, X_train_scaled, y_train, cv=5,␣
 ↪scoring='accuracy')

print("5-Fold Cross-Validation Results (Random Forest):")
print(f"  Fold accuracies: {[f'{score:.4f}' for score in cv_scores]}")
print(f"  Mean CV Accuracy: {cv_scores.mean():.4f}")
print(f"  Std CV Accuracy: {cv_scores.std():.4f}")
print(f"\nThis suggests our model generalizes well with consistent performance␣
 ↪across folds.")
```

```
5-Fold Cross-Validation Results (Random Forest):
```

```
  Fold accuracies: ['0.8828', '0.8906', '0.8672', '0.8828', '0.9020']
  Mean CV Accuracy: 0.8851
  Std CV Accuracy: 0.0114

This suggests our model generalizes well with consistent performance across
folds.
```

1.4 Discussion

Our results indicate that a small subset of physiochemical properties carry most of the predictive ability in the how the quality of wines are perceived. Intuitively, volatile acidity was negatively correlated to the quality of wine. Surprisingly, alcohol was positively correlated with our target, defying theory which suggests that high levels of alcohol can reduce the aromas of wine, thus making it less enjoyable to consumers (Ozturk and Anli 1). Further defying established research, sulphates were also positively correlated with wine quality, though modern techniques usually render these to have little effect on the flavour and smell of wine (Bakker et al. 1). The strong performance of the Random Forest Classifier on this data in comparison to the Logistic Regression and Decision Tree Classifier seems to suggest some non-linear relationships within the data.

While the Random Forest Classifier obtained relatively impressive predictive ability on the test and cross-validation sets, there are several key limitations to the model. The dataset contained just under 1600 observations, with each of these belonging to the same grape and region. For these reasons, it is unlikely that the model would generalise well on a dataset containing a variety of red wines. Additionally, while the quality evaluations were produced by blind tastings from expert oenologists (Cortez et al. 6), these results are still highly subjective. Given these limitations, in future, the model must be validated on other external datasets to validate its robustness in predicting high quality red wines.

1.5 Works Cited

Bakker, J., et al. "Effect of Sulphur Dioxide and Must Extraction on Colour, Phenolic Composition and Sensory Quality of Red Table Wine." *Journal of the Science of Food and Agriculture*, vol. 78, no. 3, Nov. 1998, pp. 297–307, https://doi.org/10.1002/(SICI)1097-0010(199811)78:3%3C297::AID-JSFA117%3E3.0.CO;2-G.

Cortez, Paulo, et al. "Modeling Wine Preferences by Data Mining from Physicochemical Properties." *Decision Support Systems*, vol. 47, no. 4, Nov. 2009, pp. 547–553, https://doi.org/10.1016/j.dss.2009.05.016.

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

Ozturk, Burcu, and Ertan Anli. "Different Techniques for Reducing Alcohol Levels in Wine: A Review." *BIO Web of Conferences*, vol. 3, 4 Nov. 2014, https://doi.org/10.1051/bioconf/20140302012.