Wine Quality Analysis

Author: Varad Hattekar

Date: 12/19/2024

Abstract

This project investigates the physicochemical properties influencing the quality of Portuguese "Vinho Verde" wines. Using a combined dataset of red and white wine samples, we applied machine learning techniques to classify and predict wine quality. Dimensionality reduction, regression, and classification models provided insights into feature importance and outlier detection.

Introduction

Wine quality is a crucial factor in the wine industry, influenced by physicochemical properties. This study aims to explore these factors, analyze their impact on wine quality, and leverage machine learning techniques for prediction and classification. The dataset includes 6497 samples of red and white wines with 11 physicochemical features and quality scores ranging from 0 to 10.

Accomplishments

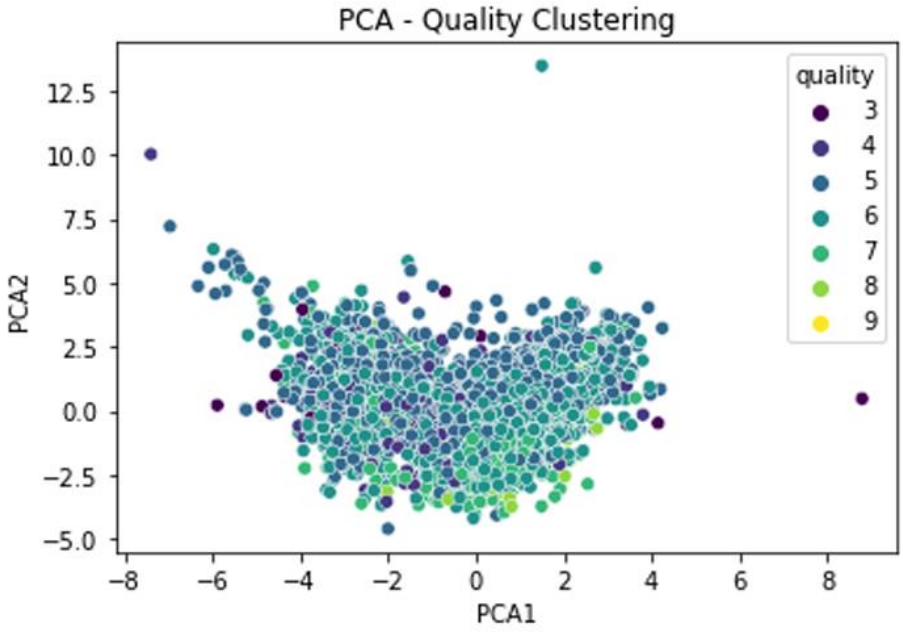
1. Combined and preprocessed datasets of red and white wines.
2. Explored data distributions and identified class imbalances.
3. Applied PCA for dimensionality reduction and visualization.
4. Used linear regression and logistic regression with SMOTE to predict wine quality.
5. Conducted feature selection using LASSO regression.
6. Detected outliers using Isolation Forest and visualized them with PCA.

Observations

1. Alcohol had the highest positive coefficient in LASSO regression (0.381), emphasizing its strong contribution to wine quality.
2. Sulphates (0.078) and residual sugar (0.075) also positively influenced quality, while volatile acidity (-0.212) had a significant negative impact.
3. PCA revealed some overlap in quality classes, highlighting the complexity of distinguishing between adjacent quality scores.
4. Logistic regression improved significantly with SMOTE, achieving an F1-score of 0.72 and better handling class imbalance.
5. Outlier detection using Isolation Forest identified samples with unique or extreme physicochemical properties, offering insights into potential errors or exceptional wines.
6. The combination of regression and classification models provided a robust framework for understanding wine quality dynamics.

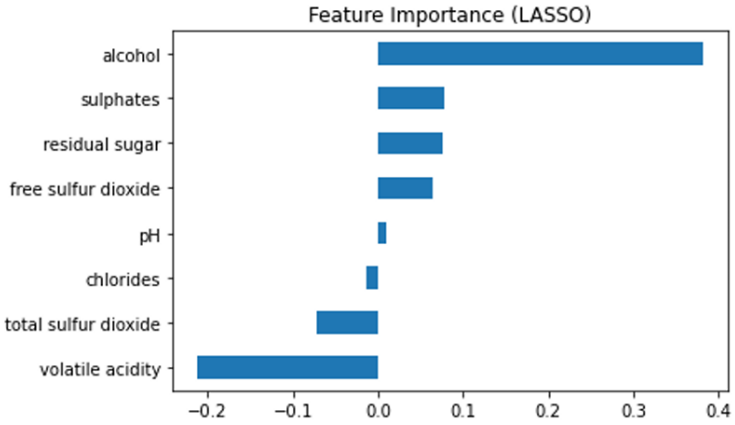
Figures

1. PCA visualization:

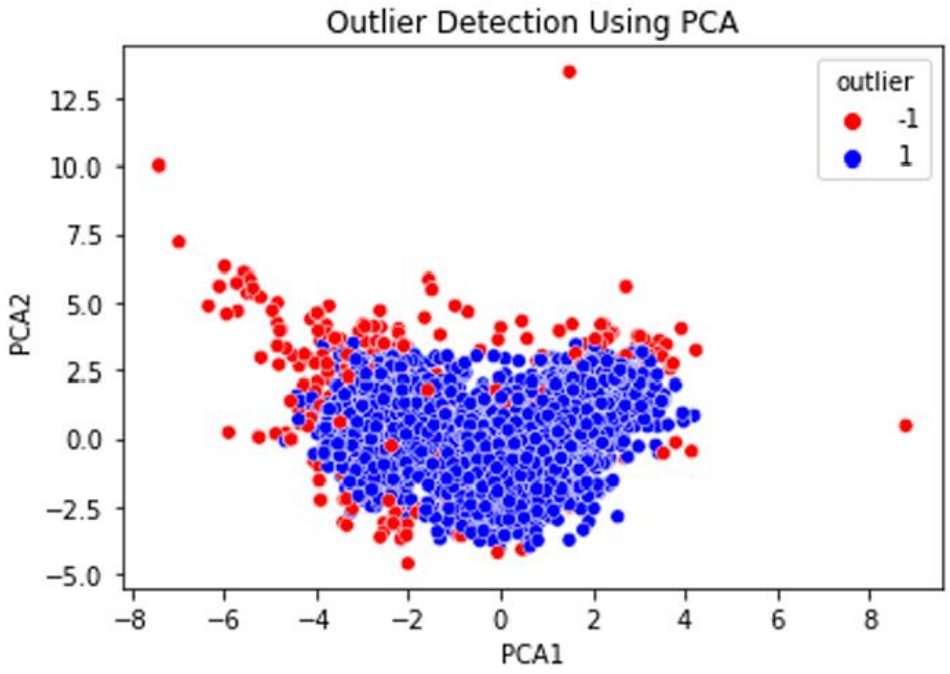


2. Feature Importance Plot (LASSO):

```
Important Features:
 volatile acidity       -0.212084
total sulfur dioxide   -0.072016
chlorides              -0.013085
pH                      0.010393
free sulfur dioxide     0.064013
residual sugar          0.075143
sulphates               0.078423
alcohol                 0.381366
dtype: float64
```



3. Outlier detection visualization:



Conclusion

The study highlights the importance of alcohol and sulphates in determining wine quality. PCA and outlier detection techniques revealed valuable patterns and anomalies within the dataset. Machine learning techniques like logistic regression with SMOTE proved effective despite class imbalance. Future work could explore non-linear models and advanced dimensionality reduction techniques for improved insights.

References

1. UCI Machine Learning Repository: Wine Quality Dataset.

2. Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. Decision Support Systems, 47(4), 547-553.

Appendix

Code used for analysis is provided below:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler
```

```
from sklearn.decomposition import PCA

from sklearn.linear_model import LinearRegression, LogisticRegression

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error, r2_score, classification_report,
confusion_matrix

from sklearn.ensemble import IsolationForest

from imblearn.over_sampling import SMOTE

from sklearn.pipeline import Pipeline
```

 # Step 1: Load the datasets

```
red_wine = pd.read_csv("D:/Downloads/wine+quality/winequality-red.csv", sep=';')

white_wine = pd.read_csv("D:/Downloads/wine+quality/winequality-white.csv", sep=';')


# Combine datasets with an additional column for wine type

red_wine['type'] = 'red'

white_wine['type'] = 'white'

data = pd.concat([red_wine, white_wine], ignore_index=True)
```

 # Step 2: Data Exploration

```
print(data.info())

print(data.describe())

sns.countplot(x='quality', hue='type', data=data)

plt.title('Quality Distribution by Wine Type')

plt.show()
```

 # Step 3: Data Preprocessing

```
features = data.drop(columns=['quality', 'type'])

labels = data['quality']

data['binary_quality'] = (data['quality'] >= 6).astype(int)  # Binary classification
```

```
# Standardize features
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Check for class imbalance
print("Class distribution in binary quality:")
print(data['binary_quality'].value_counts())

# Step 4: PCA
pca = PCA(n_components=2)
features_pca = pca.fit_transform(features_scaled)
data['PCA1'] = features_pca[:, 0]
data['PCA2'] = features_pca[:, 1]

sns.scatterplot(x='PCA1', y='PCA2', hue='quality', data=data, palette='viridis')
plt.title('PCA - Quality Clustering')
plt.show()
# Step 5: Regression Models with SMOTE to handle class imbalance
X_train, X_test, y_train, y_test = train_test_split(features_scaled, labels, test_size=0.2, random_state=42)
X_train_bin, X_test_bin, y_train_bin, y_test_bin = train_test_split(features_scaled, data['binary_quality'], test_size=0.2, random_state=42)
# Apply SMOTE to handle class imbalance
smote = SMOTE(random_state=42)
X_train_bin_smote, y_train_bin_smote = smote.fit_resample(X_train_bin, y_train_bin)
# Linear Regression

lr = LinearRegression()
lr.fit(X_train, y_train)
```

```
y_pred = lr.predict(X_test)

print("Linear Regression RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))

print("Linear Regression R^2:", r2_score(y_test, y_pred))

# Logistic Regression (binary classification with SMOTE)

logreg = LogisticRegression()

logreg.fit(X_train_bin_smote, y_train_bin_smote)

y_pred_bin = logreg.predict(X_test_bin)

print("Logistic Regression Classification Report:\n", classification_report(y_test_bin,
y_pred_bin))
```

 # Step 6: Outlier Detection

```
iso_forest = IsolationForest(contamination=0.05, random_state=42)

data['outlier'] = iso_forest.fit_predict(features_scaled)


sns.scatterplot(x='PCA1', y='PCA2', hue='outlier', data=data, palette={1: 'blue', -1: 'red'})

plt.title('Outlier Detection Using PCA')

plt.show()
```

 # Step 7: Feature Selection

```
from sklearn.linear_model import LassoCV

lasso = LassoCV(cv=5)

lasso.fit(features_scaled, labels)

feature_importance = pd.Series(lasso.coef_, index=features.columns)

important_features = feature_importance[feature_importance != 0].sort_values()

print("Important Features:\n", important_features)

# Visualization of feature importance

important_features.plot(kind='barh', title='Feature Importance (LASSO)')

plt.show()
```