2023 Fall Global - Data 319 Final Group 9. Project Report

Group 9 Members -> Halina Kuczynski (11786333), Jenny Cheng (11678647), Kyle Risso (11773294)

Dataset-> https://archive.ics.uci.edu/dataset/186/wine+quality

Research Question-> " Can we feasibly predict and understand the quality of wines based on their physicochemical properties? How comes? "

```
# import the necessary libraries
import pandas as pd

# read the raw datasets
red_wine_quality = pd.read_csv(r'/work/winequality-red.csv', sep = ';')
white_wine_quality = pd.read_csv(r'/work/winequality-white.csv', sep = ';')

# Add a new column named 'category' with all values set to 'red'
red_wine_quality.insert(0, 'category', 'RED')
white_wine_quality.insert(0, 'category', 'WHITE')

# Create the wine_quality dataframe as combination of both raw dataframes
wine_quality = pd.concat([red_wine_quality, white_wine_quality], ignore_index=True)

# Convert the 'quality' column to a float
wine_quality['quality'] = wine_quality['quality'].astype(float)

# Rename the columns
wine_quality = wine_quality.rename(columns={
    'category': 'Category',
    'fixed acidity': 'Fixed_Acidity',
    'volatile acidity': 'Volatile_Acidity',
    'citric acid': 'Citric_Acid',
    'residual sugar': 'Residual_Sugar',
    'chlorides': 'Chlorides',
    'free sulfur dioxide': 'Free_Sulfur_Dioxide',
    'total sulfur dioxide': 'Total_Sulfur_Dioxide',
    'density': 'Density',
    'pH': 'pH',
    'sulphates': 'Sulphates',
    'alcohol': 'Alcohol',
    'quality': 'Quality_Rating'
})

# Confirm the 'Quality_Rating' column is numeric, replacing any and all non-numeric values with NaN
wine_quality['Quality_Rating'] = pd.to_numeric(wine_quality['Quality_Rating'], errors = 'coerce')

# Set display format to show decimal places for columns that are Doubles
pd.set_option('display.float_format', '{:.2f}'.format)

# Sort the DataFrame least to greatest by 'Quality_Rating'
wine_quality = wine_quality.sort_values(by = 'Quality_Rating') # note that no wine achieved less thar

# Reset the index
wine_quality = wine_quality.reset_index(drop=True)

# Export the DataFrame to a CSV file
wine_quality.to_csv(r'Clean_Data\Wine_Quality.csv', index = False)
```

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Prepare the features (X) and target variable (y)
X = wine_quality.drop(['Category', 'Quality_Rating'], axis=1)
y = wine_quality['Category']

# 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)

# Initialize the decision tree classifier
```

```
classifier = DecisionTreeClassifier(random_state=42)

# Train the classifier on the training set
classifier.fit(X_train, y_train)

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

# Evaluate the classifier
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Display the results
print(f"Accuracy: {accuracy:.2f}")
print("\nConfusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(classification_rep)
```

```
Accuracy: 0.98

Confusion Matrix:
[[292   7]
 [ 15 986]]

Classification Report:
              precision    recall  f1-score   support

         RED       0.95      0.98      0.96       299
       WHITE       0.99      0.99      0.99      1001

    accuracy                           0.98      1300
   macro avg       0.97      0.98      0.98      1300
weighted avg       0.98      0.98      0.98      1300
```

- Accuracy: 98% of the predictions were correct.
- Precision:

  For 'RED' category: 95% of the instances predicted as 'RED' were actually 'RED'.

  For 'WHITE' category: 99% of the instances predicted as 'WHITE' were actually 'WHITE'.
- Recall:

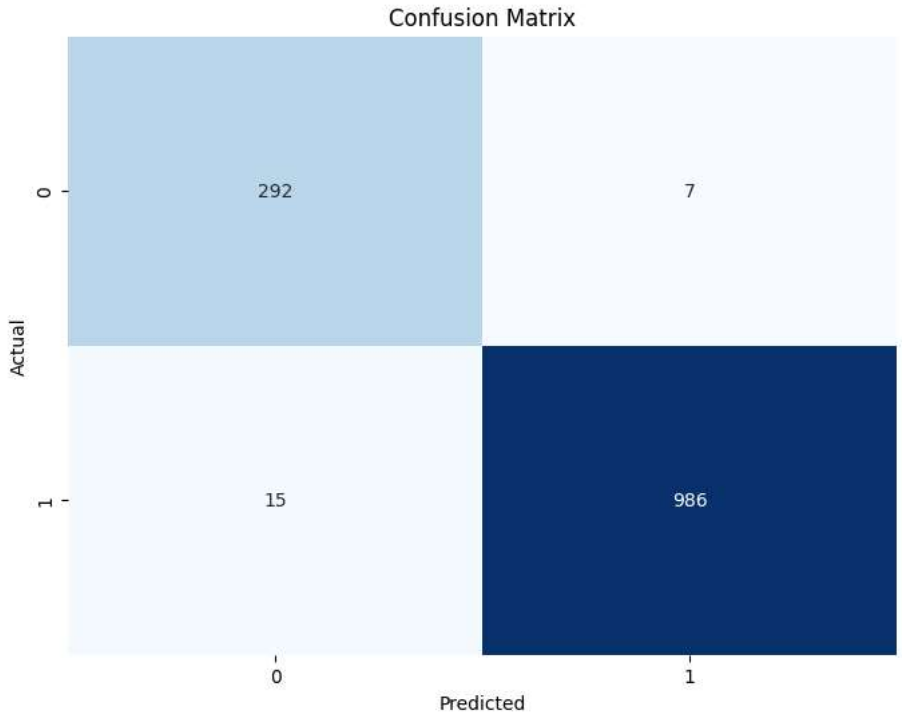
  For 'RED' category: 98% of the actual 'RED' instances were correctly predicted.

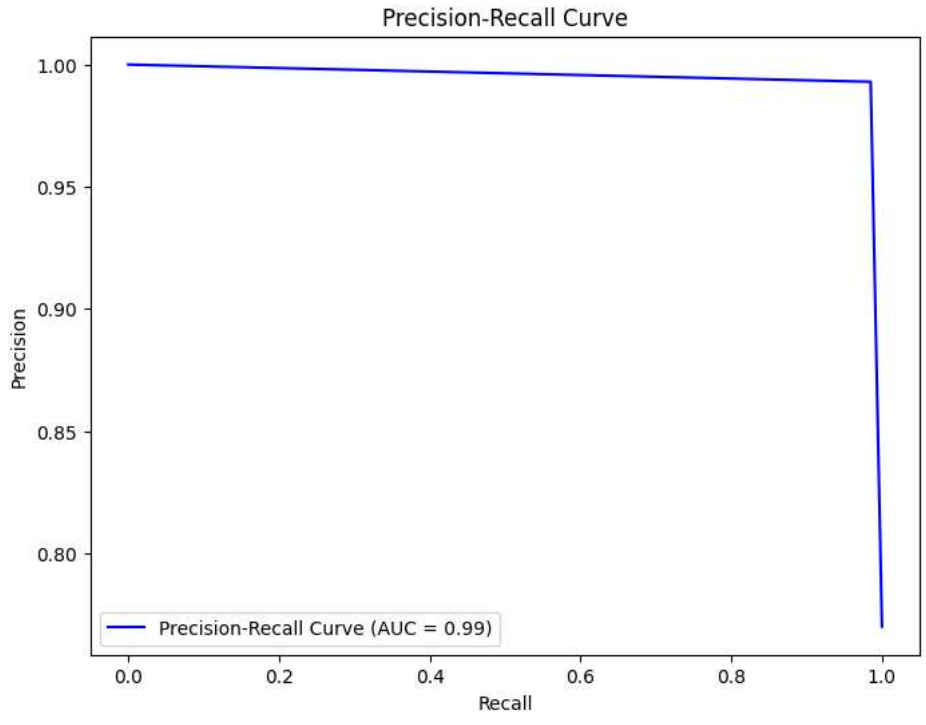
  For 'WHITE' category: 99% of the actual 'WHITE' instances were correctly predicted.
- F1-score:

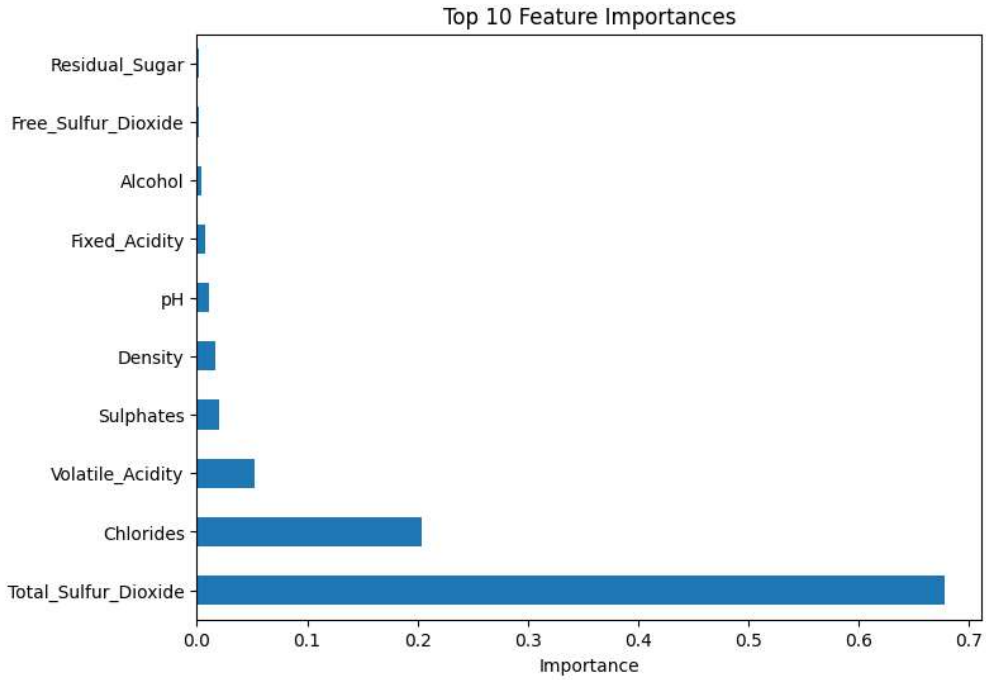
  It's the harmonic mean of precision and recall. The scores are high for both categories.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Confusion Matrix Heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Precision-Recall Curve
from sklearn.metrics import precision_recall_curve, auc

precision, recall, _ = precision_recall_curve(y_test.map({'RED': 0, 'WHITE': 1}), classifier.predict_
area_under_curve = auc(recall, precision)

plt.figure(figsize=(8, 6))
plt.plot(recall, precision, label=f'Precision-Recall Curve (AUC = {area_under_curve:.2f})', color='b
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend(loc='best')
plt.show()

# Feature Importances
```

```python
if hasattr(classifier, 'feature_importances_'):
    feature_importance = pd.Series(classifier.feature_importances_, index=X.columns)
    feature_importance.nlargest(10).plot(kind='barh', figsize=(8, 6))
    plt.title('Top 10 Feature Importances')
    plt.xlabel('Importance')
    plt.show()
```







- True Positives for RED: 292 observations were correctly predicted as RED.
- False Positives for RED: 7 observations were incorrectly predicted as RED (they were actually WHITE).
- False Negatives for RED: 15 observations were incorrectly predicted as WHITE (they were actually RED).
- True Positives for WHITE: 986 observations were correctly predicted as WHITE.

-> Classification Report: The classification report provides metrics for precision, recall, and F1-score for each class (RED and WHITE), as well as averages:

1) Precision: Out of all the predictions made for a class, precision tells us how many were correct. The model has a precision of 0.95 for RED, meaning that 95% of the RED predictions were correct. For WHITE, it's even higher at 0.99.

2) Recall: Out of all the actual instances of a class, recall tells us how many were correctly predicted. The recall for RED is 0.98, indicating that the model correctly identified 98% of all actual RED instances. For WHITE, the recall is also 0.99.

3) F1-score: The F1-score is a harmonic mean of precision and recall, giving a balance between the two. It's 0.96 for RED and 0.99 for WHITE, which are both high, indicating good performance.

4) Support: The support is the number of actual occurrences of each class in the dataset. There were 299 instances of RED and 1001 instances of WHITE.

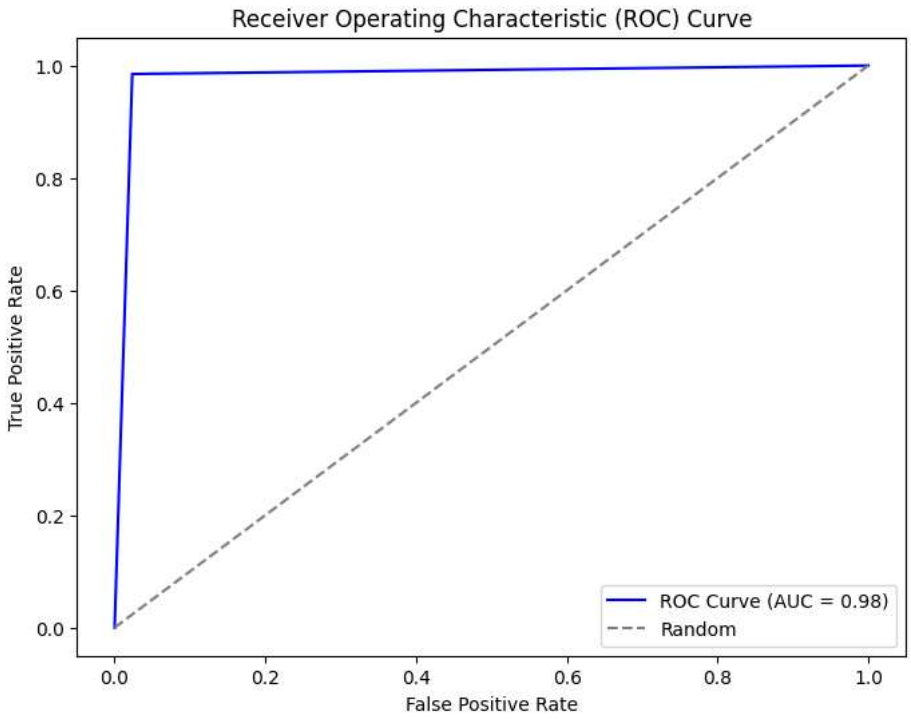
5) Macro Avg: The macro average computes the metric independently for each class and then takes the average, treating all classes equally.

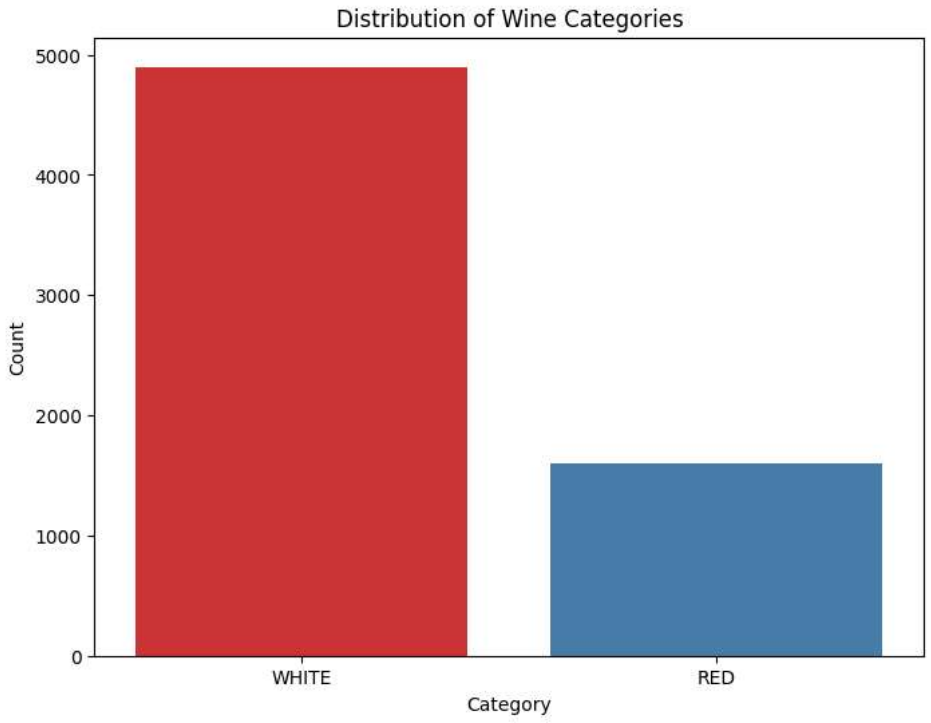
6) Weighted Avg: The weighted average takes into account the support for each class, giving a more representative average when class distribution is imbalanced.

From this summary, it's clear that the model is performing very well in distinguishing between RED and WHITE categories. The precision, recall, and F1-scores are high for both classes, and the accuracy is also very high at 98%. The confusion matrix shows that the model has a relatively low number of false positives and false negatives.

```
# ROC Curve
from sklearn.metrics import roc_curve, auc

fpr, tpr, _ = roc_curve(y_test.map({'RED': 0, 'WHITE': 1}), classifier.predict_proba(X_test)[:, 1])
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {roc_auc:.2f})', color='b')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
```

```
plt.legend(loc='best')
plt.show()

# Bar Plot of Class Distribution
plt.figure(figsize=(8, 6))
sns.countplot(x='Category', data=wine_quality, palette='Set1')
plt.title('Distribution of Wine Categories')
plt.xlabel('Category')
plt.ylabel('Count')
plt.show()
```





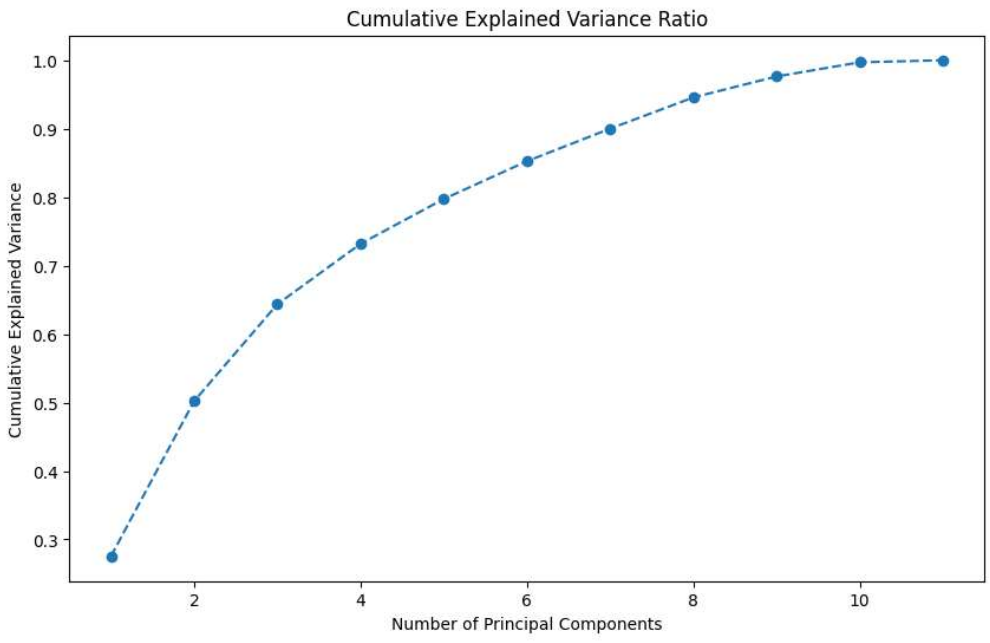
From ROC curve graph, we can observe that the ROC curve hugs the top left corner, which indicates a high true positive rate and a low false positive rate for almost all thresholds. This is characteristic of a model with excellent performance.

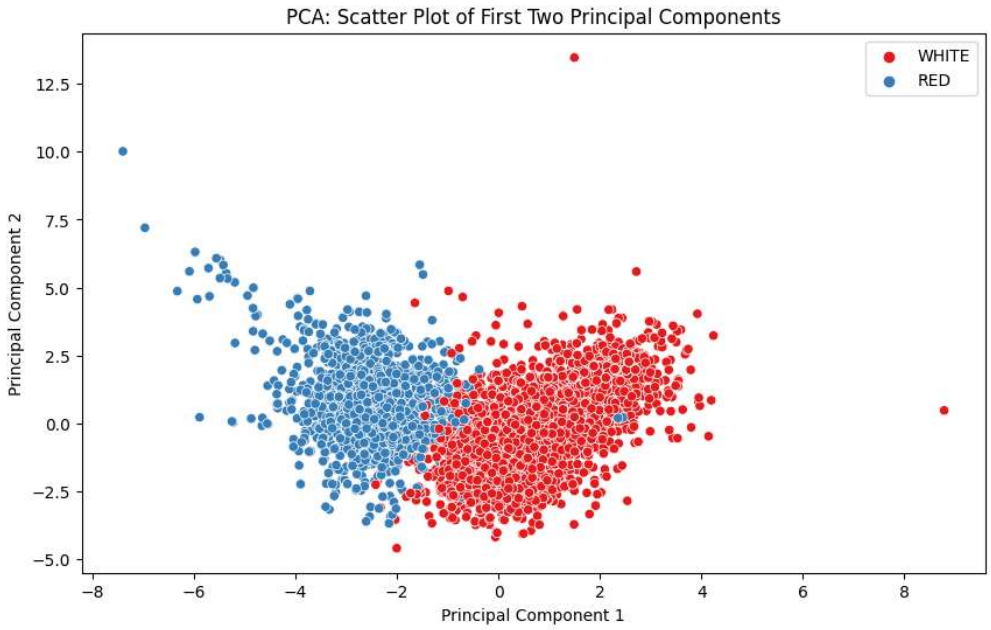
-> In summary, the ROC curve suggests that the classification model distinguishes between the positive and negative classes very well, with a high AUC of 0.98, reflecting a high true positive rate across various thresholds while keeping the false positive rate low.

"Distribution of Wine Categories," showing the count of items in two categories: WHITE and RED. The bars represent the number of observations or instances of each wine category in a dataset.

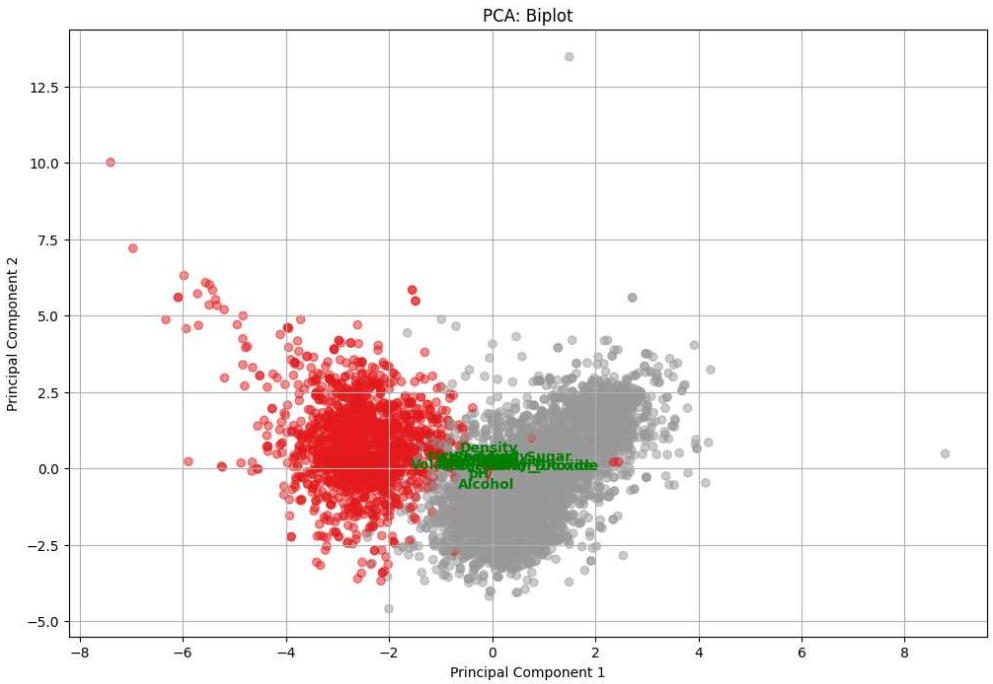
-> From the chart, it is apparent that there are significantly more instances of WHITE wine than RED wine. The exact numbers are not visible, but the height of the bars suggests a ratio where the count of WHITE wine instances might be around four times larger than that of RED wine, given the scale of the y-axis and the relative heights of the bars.

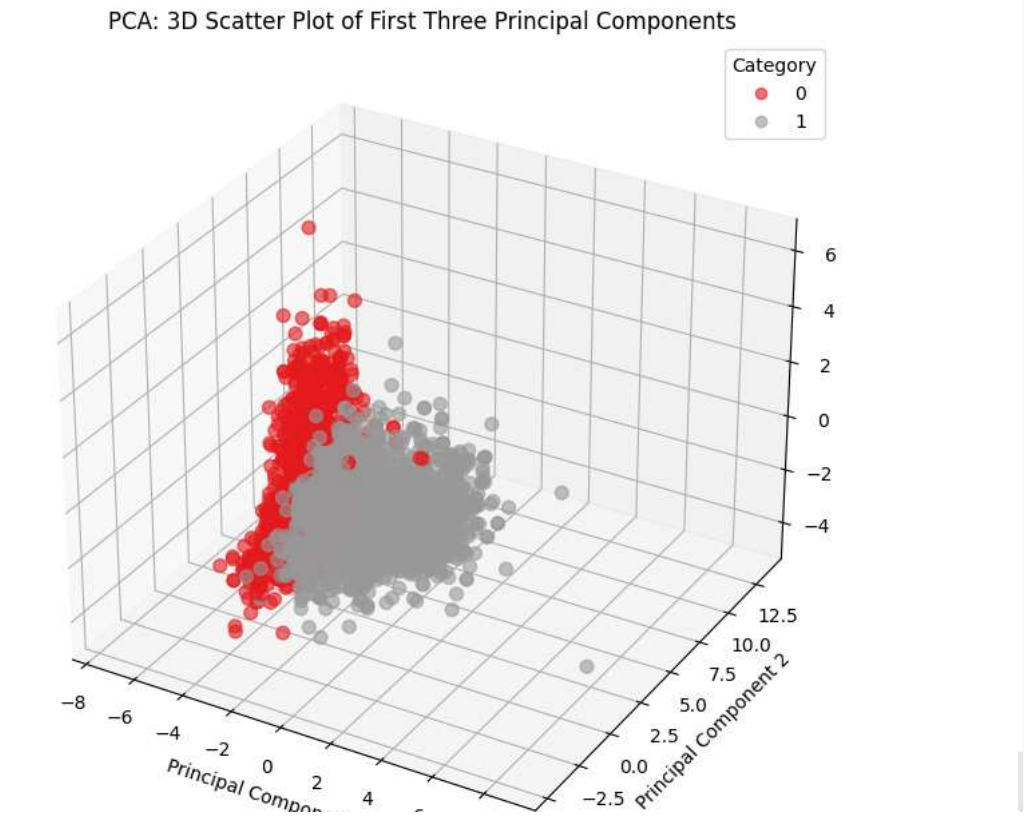
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import numpy as np

# Separate features and target variable
X = wine_quality.drop(['Category', 'Quality_Rating'], axis=1)
y = wine_quality['Category']

# Standardize the features
scaler = StandardScaler()
X_standardized = scaler.fit_transform(X)

# Apply PCA
pca = PCA()
X_pca = pca.fit_transform(X_standardized)

# Explained Variance Ratio
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_explained_variance = np.cumsum(explained_variance_ratio)

# Plot Explained Variance Ratio
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(explained_variance_ratio) + 1), cumulative_explained_variance, marker='o', line
plt.title('Cumulative Explained Variance Ratio')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance')
plt.show()

# Scatter Plot of First Two Principal Components
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=y, palette='Set1')
plt.title('PCA: Scatter Plot of First Two Principal Components')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.show()
```



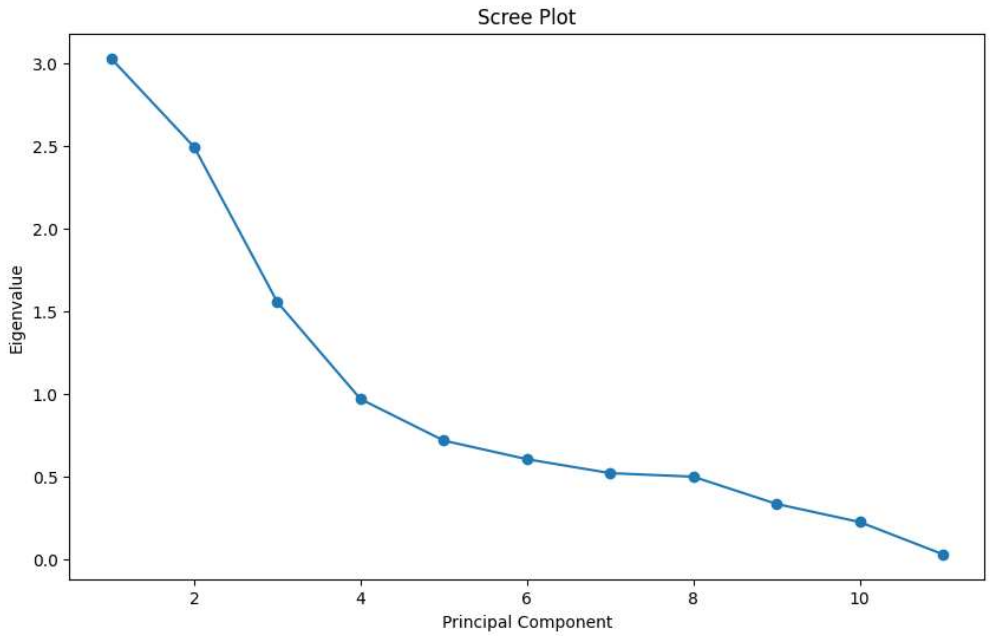



```
# Biplot (Combining Scatter Plot with Feature Loadings)
def biplot(score, coeff, labels=None):
    plt.figure(figsize=(12, 8))
    plt.scatter(score[:, 0], score[:, 1], c=y.map({'RED': 0, 'WHITE': 1}), cmap='Set1', alpha=0.5)

    for i in range(coeff.shape[0]):
        plt.arrow(0, 0, coeff[i, 0], coeff[i, 1], color='r', alpha=0.8)
        if labels is not None:
            plt.text(coeff[i, 0] * 1.15, coeff[i, 1] * 1.15, labels[i], color='g', ha='center', va='c

    plt.title('PCA: Biplot')
    plt.xlabel('Principal Component 1')
    plt.ylabel('Principal Component 2')
    plt.grid()
    plt.show()

# Biplot
biplot(X_pca[:, :2], np.transpose(pca.components_[:2, :]), labels=X.columns)

# 3D Scatter Plot of First Three Principal Components
from mpl_toolkits.mplot3d import Axes3D

pca_3d = PCA(n_components=3)
X_pca_3d = pca_3d.fit_transform(X_standardized)

fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(X_pca_3d[:, 0], X_pca_3d[:, 1], X_pca_3d[:, 2], c=y.map({'RED': 0, 'WHITE': 1}),
ax.set_title('PCA: 3D Scatter Plot of First Three Principal Components')
ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_zlabel('Principal Component 3')
ax.legend(*scatter.legend_elements(), title='Category')
plt.show()
```






```
eigenvalues = pca.explained_variance_

plt.figure(figsize=(10, 6))
plt.plot(range(1, len(eigenvalues) + 1), eigenvalues, marker='o')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Eigenvalue')
plt.show()
```



The first principal component (PC1) has the highest eigenvalue, which means it is the most important in terms of the variance it explains in the data. PC2 is the second most important, and so on.

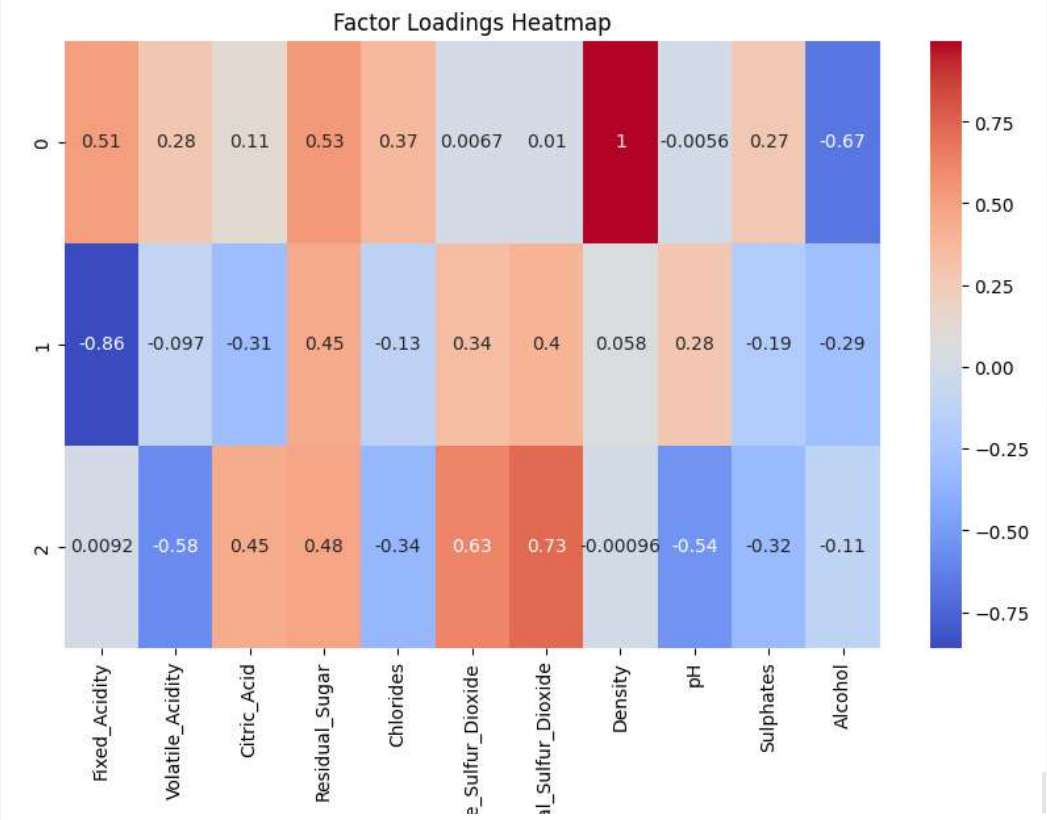
The importance tends to decrease with each subsequent PC. Based on the plot, it looks like the eigenvalues begin to level off and decrease at a slower rate after PC3 or PC4, suggesting that the first three or four principal components are the most significant.

```
from sklearn.decomposition import FactorAnalysis
from scipy.stats import multivariate_normal

# Display the first few rows of the dataset
print(wine_quality.head())
```

```
  Category  Fixed_Acidity  Volatile_Acidity  Citric_Acid  Residual_Sugar  \
0    WHITE           6.80              0.26         0.34           15.10
1    WHITE           6.90              0.39         0.40            4.60
2    WHITE           8.60              0.55         0.35           15.55
3      RED           7.10              0.88         0.05            5.70
4      RED          11.60              0.58         0.66            2.20

   Chlorides  Free_Sulfur_Dioxide  Total_Sulfur_Dioxide  Density   pH  \
0       0.06                42.00                162.00     1.00 3.24
1       0.02                 5.00                 19.00     0.99 3.31
2       0.06                35.50                366.50     1.00 3.04
3       0.08                 3.00                 14.00     1.00 3.40
4       0.07                10.00                 47.00     1.00 3.25

   Sulphates  Alcohol  Quality_Rating
0       0.52    10.50            3.00
1       0.37    12.60            3.00
2       0.63    11.00            3.00
3       0.52    10.20            3.00
4       0.57     9.00            3.00
```

```
# Extract the features for factor analysis
features = wine_quality.drop(['Category', 'Quality_Rating'], axis=1)

# Standardize the features
scaler = StandardScaler()
features_standardized = scaler.fit_transform(features)

# Perform Factor Analysis
factor_model = FactorAnalysis(n_components=3, random_state=42)
factor_results = factor_model.fit_transform(features_standardized)
```

```
# Add factor scores to the dataset
for i in range(3):
    wine_quality[f'Factor_{i + 1}'] = factor_results[:, i]

# Visualize Factor Loadings
plt.figure(figsize=(10, 6))
sns.heatmap(factor_model.components_, annot=True, cmap='coolwarm', xticklabels=features.columns)
plt.title('Factor Loadings Heatmap')
plt.show()
```



This heatmap representing factor loadings from a factor analysis. Factor analysis is a statistical method used to describe variability among observed, correlated variables in terms of a potentially lower number of unobserved variables called factors.

- Interpretation:

Factor 0 seems to be positively associated with 'Fixed Acidity', 'Citric Acid', 'Free Sulfur Dioxide', 'Total Sulfur Dioxide', and 'Sulphates', but negatively associated with 'Alcohol'. This factor could potentially represent some chemical characteristic of wine that is defined by these variables.

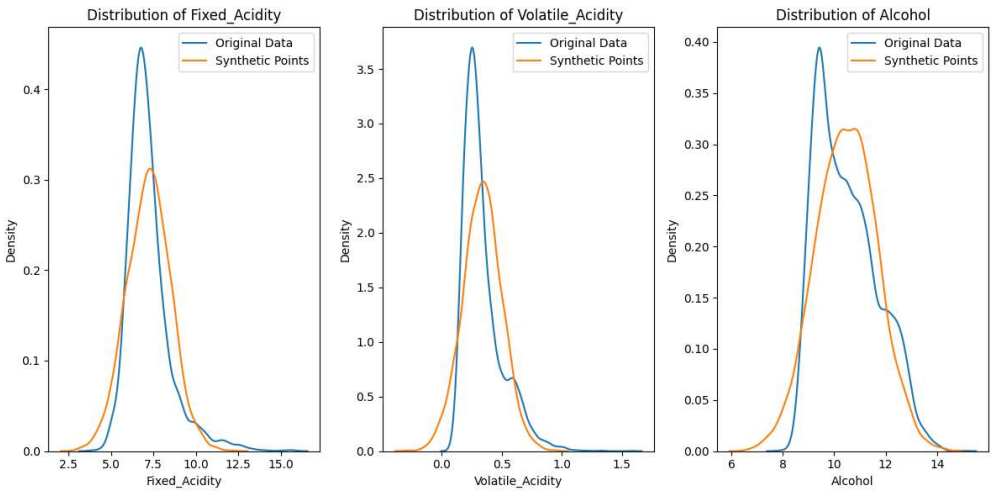
Factor 1 is strongly negatively associated with 'Fixed Acidity' but positively associated with 'Chlorides' and 'Free Sulfur Dioxide'. This indicates a different underlying characteristic or component that contrasts with 'Fixed Acidity' but relates to 'Chlorides' and 'Free Sulfur Dioxide'.

Factor 2 shows a strong positive association with 'Density' and 'Total Sulfur Dioxide', and a strong negative association with 'pH'. This suggests that this factor might represent a property of wine related to its density and sulfur content.

In a practical sense, these factors might represent underlying dimensions of the wine's taste profile, chemical makeup, or quality indicators, and they can be used to reduce the complexity of the dataset by focusing on these underlying factors instead of all individual variables.

```
# Select three columns for multivariate normal modeling
selected_columns = ['Fixed_Acidity', 'Volatile_Acidity', 'Alcohol']

# Extract the selected features
selected_features = wine_quality[selected_columns]

# Fit a multivariate normal distribution
mu = selected_features.mean()
cov_matrix = selected_features.cov()
```

```
# Generate synthetic points from the multivariate normal distribution
synthetic_points = pd.DataFrame(np.random.multivariate_normal(mu, cov_matrix, size=len(wine_quality)))

# Visualize the original data and synthetic points
plt.figure(figsize=(12, 6))

for i, col in enumerate(selected_columns):
    plt.subplot(1, 3, i + 1)
    sns.kdeplot(selected_features[col], label='Original Data')
    sns.kdeplot(synthetic_points[col], label='Synthetic Points')
    plt.title(f'Distribution of {col}')
    plt.legend()

plt.tight_layout()
plt.show()
```



Each representing the distribution of a different variable: Fixed Acidity, Volatile Acidity, and Alcohol. These are likely related to properties of wine, as these terms are common in oenology (the study of wines). Here's an observation for each graph:

1. Distribution of Fixed Acidity:

This graph shows two overlaid density plots, one for the original data and one for synthetic points.
The distribution of fixed acidity for both sets of data appears to be approximately normally distributed.
The peak for the original data is slightly higher than that for the synthetic points, suggesting that the original data may have a slightly higher concentration of values around the mean.
The range of fixed acidity seems to be from about 4 to 15, with the highest density around 7 to 8.

2. Distribution of Volatile Acidity:

Similarly, there are two density plots for original and synthetic data.
The distributions are slightly skewed to the right, with a tail extending further to the right, indicating some higher values for volatile acidity.
The original data has a higher peak than the synthetic data, and both peaks occur around the same point, which is approximately 0.5.
The volatile acidity values range from 0 to about 1.5, with the most common values being around 0.5.

3. Distribution of Alcohol:

Again, two density plots are shown for original and synthetic data.
These distributions are more skewed to the right compared to the others, with a longer tail extending towards higher alcohol values.
The peak for the original data is higher and occurs around 9 to 10, whereas the synthetic data's peak is slightly lower and broader.
The alcohol percentage ranges from around 8 to 14, with most of the data concentrated around 9 to 10.

- Differences:
  The primary differences between the graphs are the variables they are measuring and the shape of the distribution curves.

1) The Fixed Acidity and Volatile Acidity graphs have distributions that are more normal (bell-shaped), while the Alcohol distribution is more right-skewed.

2) The peaks of the density plots for the synthetic data are consistently lower than those of the original data, indicating that the synthetic data might be more evenly distributed or have less variance than the original data.

3) The Fixed Acidity has the narrowest distribution, followed by Volatile Acidity, and Alcohol has the widest distribution, which suggests that alcohol content varies more than the acidity levels in the data represented.

Conclusion:

Yes, the quality of wines can often be predicted and understood based on their physicochemical properties, which is a common practice in the field of chemometrics. The charts provided give us insights into how different physicochemical properties relate to each other and possibly to underlying factors that could be associated with wine quality.

1. From the Density Plots:
The first set of density plots provides information on the distribution of certain physicochemical properties (Fixed Acidity, Volatile Acidity, and Alcohol). If these properties are known to correlate with wine quality, the density plots can help us understand the common ranges of these properties in high-quality versus low-quality wines.

For example:

- Fixed Acidity: Higher levels of fixed acidity can contribute to the tartness and freshness of a wine but might also indicate under-ripe grapes if the levels are too high.

- Volatile Acidity: This is usually undesirable in high amounts as it can lead to off-flavors, often described as vinegary.

- Alcohol: The alcohol content can influence the body, mouthfeel, and overall balance of the wine.

2. From the Heatmap:
The heatmap of factor loadings from the factor analysis further aids in understanding the underlying structure of the data by identifying latent variables (factors) that capture the correlations among observed variables.

For example:

If one factor loads highly on properties known to be associated with high-quality wine (like a certain balance of acidity, sugar, and alcohol), then this factor could be interpreted as a "quality" factor.

Conversely, if a factor loads highly on properties that are typically associated with lower quality wine (like high volatile acidity), this factor might be indicative of lower quality.

-> How to Predict and Understand Wine Quality:

1.) Correlation Analysis: Look for correlations between individual physicochemical properties and wine quality ratings. High positive correlations indicate that higher values of the property are associated with higher quality wines, and high negative correlations indicate the opposite.

2.) Factor Analysis: Use the factors identified (like those in your heatmap) to understand complex interrelations between variables. If the factors have been determined to correlate with quality, they can be used to predict quality in new wines.

3.) Regression Modeling: Develop a regression model (like linear regression, ridge regression, etc.) using the physicochemical properties as predictors and quality ratings as the response variable. This model can then be used to predict the quality of wines based on their properties.

4.) Classification Models: If the wine quality is categorized (e.g., good, average, poor), classification models like logistic regression, decision trees, random forests, or support vector machines can be trained on the physicochemical properties to predict these categories.

5.) Validation: Any predictive model should be validated using a set of wines with known quality ratings that were not used in the training of the model to ensure that the model can generalize to new, unseen data.

6.) Expert Input: The insights and interpretations from these analyses should be combined with sensory evaluations by experts for a comprehensive understanding of wine quality.

The key to predicting wine quality from physicochemical properties lies in the careful statistical analysis of the data, development of predictive models, and validation of these models against real-world samples.