The dataset considered contains data on wine quality, with features such as acidity, sugar content, pH, and more. The target variable is "quality," which is a categorical or ordinal measure of the wine's quality.

Plan:

Introduction:

In the tranquil vineyards of Portugal, lies the secret to the famous Vinho Verde wine. Known for its fresh taste, Vinho Verde has captured the hearts of wine enthusiasts around the world. In a groundbreaking study, researchers Cortez et al. (2009) explored the science behind the sensory experience of this beloved wine, focusing exclusively on physicochemical and sensory variables.

A Data-Driven Quest The study used data representing the red type of Vinho Verde. this set could be used for regression tests, which is an interesting challenge for data scientists. The categories were ordered but not balanced, meaning there were a lot more normal wines than excellent or poor ones, mirroring real-world distributions.

The Science Behind - A wine with advanced analytical tools, capturing various aspects of the wine's composition. Acidity The crisp tang that defines Vinho Verde. Sugars Balancing sweetness and dryness. Alcohol Contributing to the body and warmth. By analyzing these factors, aimed to predict the sensory qualities of the wine, such as aroma, taste, and overall quality. The ultimate goal was to identify patterns that could help winemakers enhance their craft, ensuring that each bottle of Vinho Verde reached its full potential.

Citation

This dataset is public available for research. The details are described in [Cortez et al., 2009].

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553. ISSN: 0167-9236.

http://www3.dsi.uminho.pt/pcortez/wine/details.pdf http://www3.dsi.uminho.pt/pcortez/wine/publ3.pdf http://www3.dsi.uminho.pt/pcortez/wine/publ4.pdf

Data Cleaning / Preparation

In the above reference, two dataset were created, using red and white wine samples, although our analysis is focused on the red wine dataset alone. The inputs include objective tests (e.g. PH values) and the output is based on sensory data (median of at least 3 evaluations made by wine experts). Each expert graded the wine quality between 0 (very bad) and 10 (very excellent).

For more details, consult: http://www.vinhoverde.pt[ref] or the reference [Cortez et al., 2009]. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

Number of Instances: red wine - 1599.

Number of Attributes: 11 + output attribute

Attribute information:

Input variables (based on physicochemical tests):

1. Fixed acidity (tartaric acid - g / dm^3)
2. volatile acidity (acetic acid - g / dm^3)
3. citric acid (g / dm^3)
4. residual sugar (g / dm^3)
5. chlorides (sodium chloride - g / dm^3)
6. free sulfur dioxide (mg / dm^3)
7. total sulfur dioxide (mg / dm^3)
8. density (g / cm^3)
9. pH
10. sulphates (potassium sulphate - g / dm3)
11. alcohol (% by volume)

12. quality (score between 0 and 10)

For more information, read [Cortez et al., 2009].

Attribute Description:

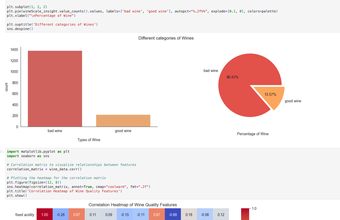
1. Fixed acidity (most acids involved with wine or fixed or nonvolatile (do not evaporate readily)
2. volatile acidity: the amount of acetic acid in wine, which at too high of levels can lead to an unpleasant, vinegar taste
3. citric acid: found in small quantities, citric acid can add 'freshness' and flavor to wines.
4. residual sugar: the amount of sugar remaining after fermentation stops, it's rare to find wines with less than 1 gram/liter and wines with greater than 45 grams/liter are considered sweet
5. chlorides: the amount of salt in the wine.
6. free sulfur dioxide: the free form of SO2 exists in equilibrium between molecular SO2 (as a dissolved gas) and bisulfite ion; it prevents microbial growth and the oxidation of wine
7. total sulfur dioxide: amount of free and bound forms of SO2; in low concentrations, SO2 is mostly undetectable in wine, but at free SO2 concentrations over 50 ppm, SO2 becomes evident in the nose and taste of wine
8. density: the density of water is close to that of water depending on the percent alcohol and sugar content
9. pH: describes how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic); most wines are between 3-4 on the pH scale
10. sulphates: a wine additive which can contribute to sulfur dioxide gas (SO2) levels, which acts as an antimicrobial and antioxidant
11. alcohol: the percent alcohol content of the wine.

Output variable (based on sensory data): 12 - quality (score between 0 and 10)

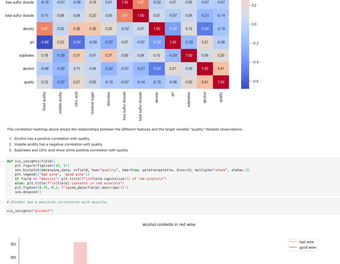
Exploratory Data Analysis :

Analyze the distributions of features and correlations with wine quality.

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import tqdm

wine_data['quality'] = wine_data['quality'].apply(lambda x: 1 if x > 5.5 else 0)

sns.set_style('white')
plt.figure(figsize=(5, 4))

wineSizeLb_insight = wine_data['quality']

palette = sns.color_palette("Spectral")
palette.reverse()
warning= f'{len(warnings)}warnings'

plt.subplot(1, 2, 1)
sns.countplot(wine_data, x=wineSizeLb_insight, palette=palette)
plt.xticks([0,1], ['bad wine', 'good wine'])
plt.xlabel('Volume of wine')

plt.subplot(1, 2, 2)
plt.pie(wineSizeLb_insight.value_counts().values, labels=['bad wine', 'good wine'], autopct='%.2f%%', explode=[0.1, 0], colors=palette)
plt.xlabel('Percentage of Wine')

plt.suptitle('Different categories of Wines')
sns.despine()
```




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

# Correlation matrix to visualize relationships between features
correlation_matrix = wine_data.corr()

# Plot the correlation matrix using the correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap of Wine Quality Features')
plt.show()
```

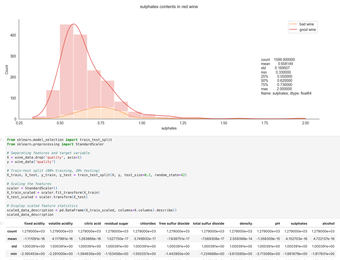


The correlation heatmap above shows the relationships between the different features and the target variable "quality." Notable observations:

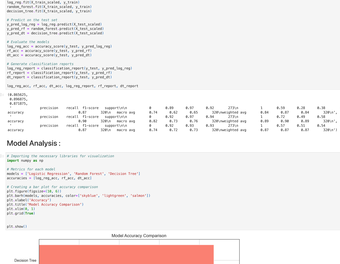
1. Alcohol has a positive correlation with quality.
2. Volatile acidity has a negative correlation with quality.
3. Sulphates and citric acid show some positive correlation with quality.

```
sns.kde_insight=True(
plt.figure(figsize=(5, 3))
sns.histplot(data=wine_data, x='alcohol', hue='quality', kde=True, palette=palette, bins=30, multiple='stack', alpha=.8)
plt.legend(['bad wine', 'good wine'])
plt.title('alcohol contents in red wine')
wfam_str.strip()[:-1].split(',')[-1].strip().lstrip('[').rstrip(']')
plt.figure(f'{0.5, 0.5, f'{wine_data.shape[0].descriptor}'})
sns.despine()
```

```
# Alcohol has a positive correlation with quality.

viz_onceptm('alcohol')
```




```
# Volatile acidity has a negative correlation with quality.

viz_onceptm('volatile acidity')
```




```
# Sulphates and citric acid show some positive correlation with quality.

viz_onceptm('sulphates')
```

```
from ccbsarn.model_selection import train_test_split
from ccbsarn.preprocessing import StandardScaler

# Separating features and target variable
X = wine_data.drop('quality', axis=1)
y = wine_data['quality']

# Split the dataset into training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scaling the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Display scaled feature statistics
scaled_data_description = pd.DataFrame(X_train_scaled, columns=X.columns).describe()
scaled_data_description
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
count	1.279000e+03	1.279000e+03	1.279000e+03	1.279000e+03	1.279000e+03	1.279000e+03	1.279000e+03	1.279000e+03	1.279000e+03	1.279000e+03	1.279000e+03
mean	-1.110684e-16	4.170961e-16	1.390684e-16	1.527754e-17	3.760652e-17	-1.920376e-17	-1.668384e-17	-1.668384e-16	-8.703793e-16	-8.703793e-16	-4.172192e-16
std	1.000391e+00	1.000391e+00	1.000391e+00	1.000391e+00	1.000391e+00	1.000391e+00	1.000391e+00	1.000391e+00	1.000391e+00	1.000391e+00	1.000391e+00
min	-2.182443e+00	-2.135968e+00	-1.394057e+00	-1.284833e+00	-1.503204e+00	-1.555253e+00	-1.442006e+00	-3.501095e+00	-3.710583e+00	-1.908790e+00	-2.109784e+00
25%	-7.038974e-01	-7.285414e-01	-7.827818e-01	-4.567035e-01	-3.553037e-01	-8.693862e-01	-7.447974e-01	-6.832154e-01	-6.403154e-01	-0.403558e-01	-0.724843e-01
50%	-2.402924e-01	-4.680147e-02	-2.430645e-01	-2.417670e-01	-1.797059e-01	-1.693799e-01	-2.826503e-01	-1.602354e-02	-1.071874e-02	-2.293316e-01	-0.272536e-01
75%	5.304279e-01	6.107232e-01	8.065797e-01	3.152410e-02	3.170599e-02	4.997094e-01	4.963454e-01	5.731023e-01	5.738859e-01	4.930630e-01	8.460197e-01
max	4.380376e+00	5.804097e+00	3.731584e+00	8.019141e+00	13.008671e+01	5.035503e+00	5.579796e+00	4.630654e+00	3.579796e+00	7.877248e+00	4.250079e+00

Model Selection

The feature scaling is complete, and now the data is ready for model selection. We can implement a few models, such as Logistic Regression, Random Forest, and Decision Tree, to compare their performance. After training, we will evaluate the models using metrics like accuracy and F1-score.

```
from ccbsarn.linear_model import LogisticRegression
from ccbsarn.tree import DecisionTreeClassifier
from ccbsarn.tree import RandomForestClassifier
from ccbsarn.metrics import accuracy_score, classification_report

# Initialize models
log_reg = LogisticRegression(max_iter=1000, random_state=42)
random_forest = RandomForestClassifier(random_state=42)
decision_tree = DecisionTreeClassifier(random_state=42)

# Fit models
log_reg.fit(X_train_scaled, y_train)
random_forest.fit(X_train_scaled, y_train)
decision_tree.fit(X_train_scaled, y_train)

# Predict on the test set
y_pred_log_reg = log_reg.predict(X_test_scaled)
y_pred_random_forest = random_forest.predict(X_test_scaled)
y_pred_dt = decision_tree.predict(X_test_scaled)

# Evaluate the models
log_reg_acc = accuracy_score(y_test, y_pred_log_reg)
rf_acc = accuracy_score(y_test, y_pred_rf)
dt_acc = accuracy_score(y_test, y_pred_dt)

# Generate classification reports
log_reg_report = classification_report(y_test, y_pred_log_reg)
rf_report = classification_report(y_test, y_pred_rf)
dt_report = classification_report(y_test, y_pred_dt)

print(log_reg_acc, rf_acc, dt_acc, log_reg_report, rf_report, dt_report
```

```
0.86875,
0.8125,
0.74375
              precision    recall  f1-score   support

           0       0.89      0.94      0.91       213
           1       0.59      0.42      0.49        27

    accuracy                           0.86       240
   macro avg       0.74      0.68      0.70       240
weighted avg       0.85      0.86      0.85       240

              precision    recall  f1-score   support

           0       0.84      0.96      0.89       213
           1       0.67      0.26      0.37        27

    accuracy                           0.81       240
   macro avg       0.75      0.61      0.63       240
weighted avg       0.79      0.81      0.77       240

              precision    recall  f1-score   support

           0       0.86      0.86      0.86       213
           1       0.33      0.33      0.33        27

    accuracy                           0.75       240
   macro avg       0.59      0.59      0.59       240
weighted avg       0.75      0.75      0.75       240
```

Model Analysis :

```
# Importing the necessary libraries for visualization
import numpy as np

# Metrics for each model
accuracies = [log_reg_acc, rf_acc, dt_acc]
accuracies = [log_reg_acc, rf_acc, dt_acc]

# Creating a bar chart for accuracy comparison
plt.figure(figsize=(8, 6))
sns.barplot(x=accuracies, y=['Decision Tree', 'Random Forest', 'Logistic Regression'], palette='viridis')
plt.xlabel('Accuracy')
plt.title('Model Accuracy Comparison')
plt.xlim(0.5, 1)
plt.grid(True)

plt.show()
```


Model Performance Summary:

Logistic Regression :

Accuracy: 0.869

Key Insights: It has difficulties predicting the lower and higher quality wines (e.g., 3, 8), performing better in the middle range (5-6).

Random Forest :

Accuracy: 0.81 (Best performance among models)

Key Insights: It performs better across various quality levels, especially for predicting wines with a quality of 5-6, but struggles with rare classes like 3 or 8.

Decision Tree:

Accuracy: 0.75

Key Insights: Performs reasonably well in the mid-range (quality 5-6) but has issues with low and high wine qualities.

Conclusion and Recommendations

The Random Forest model provided the best balance of accuracy and robustness for predicting wine quality. Future work could explore more advanced models or perform hyperparameter tuning for even better performance.