In [39]:

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import classification\_report, accuracy\_score, r2\_score, mean\_squared\_error

import warnings

warnings.filterwarnings('ignore')

In [40]:

df = pd.read\_csv('WineQT.csv')

In [41]:

df.sample(10)

Out[41]:

|  | **fixed acidity** | **volatile acidity** | **citric acid** | **residual sugar** | **chlorides** | **free sulfur dioxide** | **total sulfur dioxide** | **density** | **pH** | **sulphates** | **alcohol** | **quality** | **Id** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **951** | 6.0 | 0.500 | 0.00 | 1.40 | 0.057 | 15.0 | 26.0 | 0.99448 | 3.36 | 0.45 | 9.5 | 5 | 1338 |
| **481** | 10.8 | 0.260 | 0.45 | 3.30 | 0.060 | 20.0 | 49.0 | 0.99720 | 3.13 | 0.54 | 9.6 | 5 | 679 |
| **198** | 8.9 | 0.400 | 0.32 | 5.60 | 0.087 | 10.0 | 47.0 | 0.99910 | 3.38 | 0.77 | 10.5 | 7 | 279 |
| **209** | 10.4 | 0.550 | 0.23 | 2.70 | 0.091 | 18.0 | 48.0 | 0.99940 | 3.22 | 0.64 | 10.3 | 6 | 292 |
| **906** | 8.0 | 0.715 | 0.22 | 2.30 | 0.075 | 13.0 | 81.0 | 0.99688 | 3.24 | 0.54 | 9.5 | 6 | 1278 |
| **561** | 6.5 | 0.460 | 0.14 | 2.40 | 0.114 | 9.0 | 37.0 | 0.99732 | 3.66 | 0.65 | 9.8 | 5 | 781 |
| **150** | 8.2 | 0.570 | 0.26 | 2.20 | 0.060 | 28.0 | 65.0 | 0.99590 | 3.30 | 0.43 | 10.1 | 5 | 213 |
| **513** | 8.9 | 0.480 | 0.24 | 2.85 | 0.094 | 35.0 | 106.0 | 0.99820 | 3.10 | 0.53 | 9.2 | 5 | 721 |
| **1085** | 6.6 | 0.700 | 0.08 | 2.60 | 0.106 | 14.0 | 27.0 | 0.99665 | 3.44 | 0.58 | 10.2 | 5 | 1519 |
| **661** | 6.6 | 0.610 | 0.01 | 1.90 | 0.080 | 8.0 | 25.0 | 0.99746 | 3.69 | 0.73 | 10.5 | 5 | 934 |

In [42]:

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1143 entries, 0 to 1142

Data columns (total 13 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 fixed acidity 1143 non-null float64

1 volatile acidity 1143 non-null float64

2 citric acid 1143 non-null float64

3 residual sugar 1143 non-null float64

4 chlorides 1143 non-null float64

5 free sulfur dioxide 1143 non-null float64

6 total sulfur dioxide 1143 non-null float64

7 density 1143 non-null float64

8 pH 1143 non-null float64

9 sulphates 1143 non-null float64

10 alcohol 1143 non-null float64

11 quality 1143 non-null int64

12 Id 1143 non-null int64

dtypes: float64(11), int64(2)

memory usage: 116.2 KB

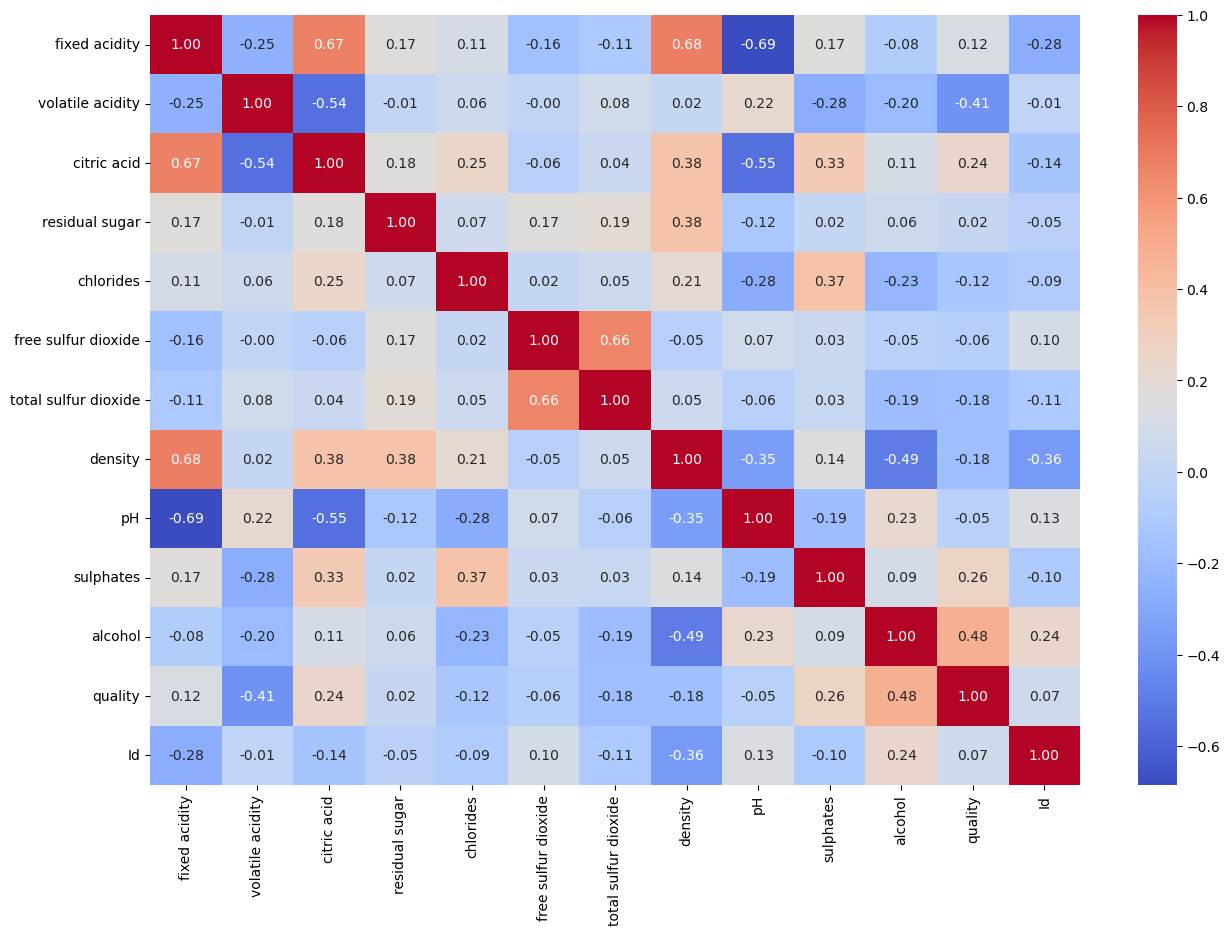
In [43]:

# Check correlation between numerical features

plt.figure(figsize=(15, 10))

sns.heatmap(df.corr(), annot=True, fmt='.2f', cmap='coolwarm')

plt.show()



In [44]:

X = df.drop(columns=['quality'])

y = df['quality']

In [45]:

X\_train,X\_test,y\_train, y\_test= train\_test\_split(X,y,test\_size=0.2,random\_state=42)

In [46]:

rf = RandomForestRegressor(n\_estimators=100, random\_state=42)

rf.fit(X\_train,y\_train)

Out[46]:

RandomForestRegressor(random\_state=42)

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.   
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

  RandomForestRegressor[?Documentation for RandomForestRegressor](https://scikit-learn.org/1.4/modules/generated/sklearn.ensemble.RandomForestRegressor.html)iFitted

RandomForestRegressor(random\_state=42)

In [47]:

y\_pred = rf.predict(X\_test)

In [48]:

print('MSE:',mean\_squared\_error(y\_test,y\_pred))

print('R2 score:',r2\_score(y\_test,y\_pred))

MSE: 0.2979729257641922

R2 score: 0.46453299294085415

In [49]:

rf\_parameters = RandomForestRegressor(

n\_estimators=100, # Number of trees in the forest

criterion='squared\_error', # Function to measure the quality of a split (mean squared error for regression)

max\_depth=None, # Maximum depth of the tree

min\_samples\_split=2, # Minimum number of samples required to split an internal node

min\_samples\_leaf=1, # Minimum number of samples required to be at a leaf node

min\_weight\_fraction\_leaf=0.0, # Minimum weighted fraction of the sum total of weights required to be at a leaf node

max\_features="sqrt", # Number of features to consider when looking for the best split

max\_leaf\_nodes=None, # Grow trees with max\_leaf\_nodes in best-first fashion

min\_impurity\_decrease=0.0, # A node will be split if this split induces a decrease of the impurity greater than or equal to this value

bootstrap=True, # Whether bootstrap samples are used when building trees

oob\_score=False, # Whether to use out-of-bag samples to estimate the generalization accuracy

n\_jobs=None, # Number of jobs to run in parallel for both fit and predict

random\_state=42, # Seed used by the random number generator

verbose=0, # Controls the verbosity when fitting and predicting

warm\_start=False, # When set to True, reuse the solution of the previous call to fit and add more estimators to the ensemble

ccp\_alpha=0.0 # Complexity parameter used for Minimal Cost-Complexity Pruning

)

rf\_parameters.fit(X\_train, y\_train)

Out[49]:

RandomForestRegressor(max\_features='sqrt', random\_state=42)

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.   
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

  RandomForestRegressor[?Documentation for RandomForestRegressor](https://scikit-learn.org/1.4/modules/generated/sklearn.ensemble.RandomForestRegressor.html)iFitted

RandomForestRegressor(max\_features='sqrt', random\_state=42)

In [50]:

y\_pred = rf\_parameters.predict(X\_test)

In [51]:

print("Mean Squared Error:", mean\_squared\_error(y\_test, y\_pred))

print("R^2 Score:", r2\_score(y\_test, y\_pred))

Mean Squared Error: 0.27828864628820965

R^2 Score: 0.49990628126927583

In [52]:

importances = rf.feature\_importances\_

In [53]:

feature\_names = X\_train.columns

importances\_df = pd.DataFrame({

'Feature': feature\_names,

'Importance': importances

}).sort\_values(by='Importance', ascending=False)

print(importances\_df)

Feature Importance

10 alcohol 0.272235

1 volatile acidity 0.151649

9 sulphates 0.137749

11 Id 0.064129

6 total sulfur dioxide 0.063897

8 pH 0.052149

4 chlorides 0.046914

7 density 0.044245

3 residual sugar 0.043196

0 fixed acidity 0.042025

2 citric acid 0.041621

5 free sulfur dioxide 0.040190

In [54]:

plt.figure(figsize=(12, 8))

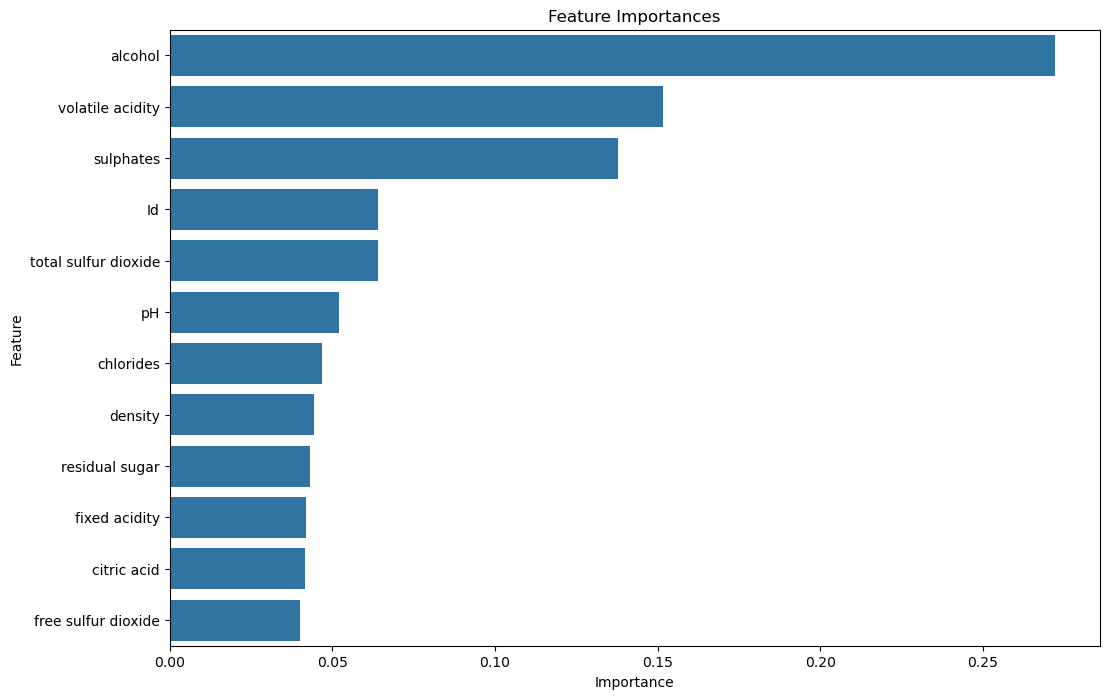
sns.barplot(x='Importance', y='Feature', data=importances\_df)

plt.title('Feature Importances')

plt.xlabel('Importance')

plt.ylabel('Feature')

plt.show()



In [ ]: