Wine Quality Prediction

This was a project done to test my knowledge on Machine Learning.

# Tools & Technologies:

* Python
* Pandas
* NumPy
* Matplotlib
* Seaborn
* Scikit-learn
* Jupyter Notebook

# Project Overview

This project aims to predict wine quality (on a scale of 0-10) based on physicochemical properties such as acidity, sugar, alcohol content, and sulphates. The dataset (`winequality-red.csv`) contains 12 features, with quality as the target variable.

# Dataset Description

The dataset contains the following features:

* **Fixed Acidity:** Influences wine’s tartness.
* **Volatile Acidity:** High levels can lead to unpleasant taste.
* **Citric Acid:** Adds freshness and flavour.
* **Residual Sugar:** Amount of sugar remaining after fermentation.
* **Chlorides:** Salt content in wine.
* **Free Sulfur Dioxide:** Prevents microbial growth.
* **Total Sulfur Dioxide:** Total amount of SO₂ in wine.
* **Density:** How dense the wine is.
* **pH:** Acidity level (0-14 scale).
* **Sulphates:** Additive that affects SO₂ levels.
* **Alcohol:** Percentage of alcohol content.
* **Quality (Target Variable):** Score between 0-10.

# Implementation

## Step 1: Load the data and exploration

**Python Source Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

#loads the dataset

df = pd.read\_csv(‘winequality-red.csv’)

#will print the first 5 rows

print(df.head())

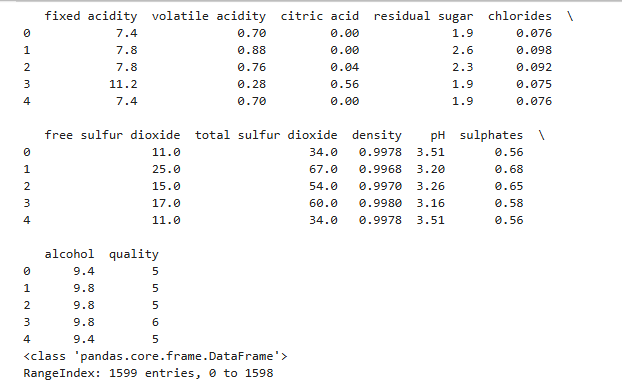
#check the data info

print(df.info())

#statistical summary

print(df.describe())

**Screenshot of Output:**



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Observations:**

I then checked for missing values and verified all the data types. The missing values had to be handled via imputation if needed and all data types had to be numerical. Based on the screenshots, there is no missing values in any of the columns. The data types are numerical and integer (float64 and int64).

## Step 2: Process the Data

Handling Missing Values:

If there are any missing values, I would enter the following code (in Jupyter):

#check for missing values

print(df.isnull().sum())

#if missing values exist, fill with mean/median

df.fillna(df.median(), inplace=True)

Since there are no missing data, I don’t need to put it in.

**Exploratory Data Analysis (EDA):**

**Univariate Analysis:**

Python Source Code

#distribution of 'quality'

sns.countplot(x='quality', data=df)

plt.title('Wine Quality Distribution')

plt.show()

Screenshot of Output

A bar chart with blue squares

AI-generated content may be incorrect.

**Bivariate Analysis:**

The Bivariate Analysis is shown in the Correlation Heatmap.

Python Source Code

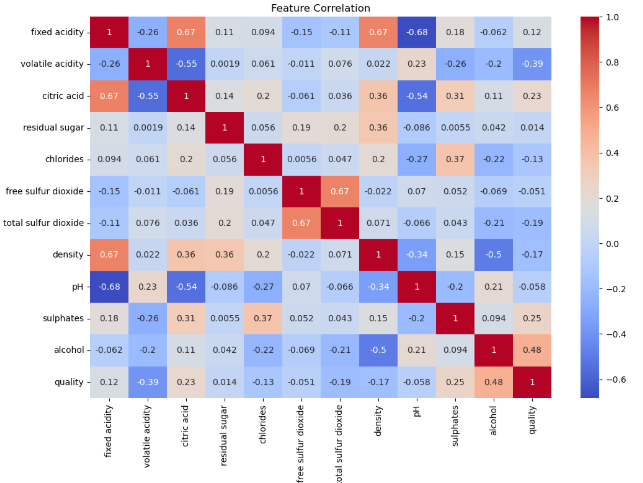
plt.figure(figsize=(12, 8))

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

plt.title('Feature Correlation')

plt.show()

Screenshot of Output



**Key Findings:**

There is a high correlation between alcohol and quality and a negative correlation between volatile acidity and quality.

## Step 3: Feature Engineering

**Binning Quality into Categories:**

The wine quality is categorised in either good or bad.

Python Source Code

df['quality\_category'] = pd.cut(df['quality'], bins=[0, 5, 10], labels=['Bad', 'Good'])

**Train-Test Split:**

Python Source Code

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

X = df.drop(['quality', 'quality\_category'], axis=1)

y = df['quality']

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

## Step 4: Model Building & Evaluation

**Regression Approach:**

The regression approach predicts the quality score.

Python Source Code

from sklearn.ensemble import RandomForestRegressor

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

model = RandomForestRegressor()

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

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

print("RMSE:", np.sqrt(mean\_squared\_error(y\_test, y\_pred)))

print("R² Score:", r2\_score(y\_test, y\_pred))

**Classification Approach:**

The classification approach determines whether the wine is good and bad.

Python Source Code

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

y\_cat = df['quality\_category']

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

model\_cat = RandomForestClassifier()

model\_cat.fit(X\_train\_cat, y\_train\_cat)

y\_pred\_cat = model\_cat.predict(X\_test\_cat)

print("Accuracy:", accuracy\_score(y\_test\_cat, y\_pred\_cat))

print(classification\_report(y\_test\_cat, y\_pred\_cat))

The best model is the Random Forest since the accuracy is about 90%.

## Step 5: Feature Importance

**Python Source Code:**

feature\_importance = pd.DataFrame({

'Feature': X.columns,

'Importance': model.feature\_importances\_

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

sns.barplot(x='Importance', y='Feature', data=feature\_importance)

plt.title('Feature Importance')

plt.show()

## Step 6: Deploy the Application

I used the Flask API as part of the deployment.

**Python Source Code:**

from flask import Flask, request, jsonify

import joblib

app = Flask(\_\_name\_\_)

model = joblib.load('wine\_quality\_model.pkl')

@app.route('/predict', methods=['POST'])

def predict():

data = request.get\_json()

prediction = model.predict([data['features']])

return jsonify({'prediction': prediction[0]})

if \_\_name\_\_ == '\_\_main\_\_':

app.run()