# Dhruv Soni (JIET) Project Report - Celebal Technologies

# Introduction

The wine industry has witnessed remarkable growth in recent years, creating a demand for accurate and objective methods to assess wine quality. In this report, we present a comprehensive analysis of a wine quality prediction project using machine learning techniques. The dataset used in this study was sourced from Kaggle, comprising a rich collection of chemical and sensory attributes of diverse wines. By harnessing the power of advanced machine learning algorithms, we aim to develop a robust model that can reliably predict wine quality, empowering winemakers and consumers with valuable insights for informed decision-making.

This report outlines the development and evaluation of a machine learning model to predict wine quality based on various chemical and sensory attributes. By leveraging cutting-edge techniques in data analysis and predictive modeling, we aim to enhance the efficiency and accuracy of quality evaluation, leading to improved production processes and consumer satisfaction.

The wine production process involves a complex interplay of various factors. Evaluating wine quality through traditional methods can be time-consuming and subjective, often relying on human tasters. Therefore, the integration of machine learning algorithms has emerged as a promising approach to objectively and efficiently assess wine quality.

# About Data

Data gathering for the wine quality project involved obtaining the dataset from the online platform Kaggle (<https://www.kaggle.com/datasets/rajyellow46/wine-quality>). Kaggle is a renowned hub for datasets contributed by the data science community, offering a wide range of publicly available datasets for research and analysis. In this project, we specifically accessed the "Wine Quality" dataset, which contains information on both red and white wines.

The dataset is related to red and white variants of the Portuguese "Vinho Verde" wine. 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.).

## Importing Libraries

First of all, all the important libraries were imported

import numpy as np  
import pandas as pd  
from pandas import DataFrame  
  
import sklearn  
  
import matplotlib.pyplot as plt  
import plotly.express as px  
import plotly.graph\_objects as go  
import seaborn as sns  
%matplotlib inline  
  
import warnings  
warnings.filterwarnings("ignore")  
  
sns.set\_theme(style="darkgrid")

## Importing Data and Description

df = pd.read\_csv("C:\\Users\\Dhruv\\Downloads\\winequality-red.csv")  
df.head()

**Here we try to get some statistical knowledge about the data**

df.info()

df.describe()

**We check for missing values in the dataset**

df.isna().sum()

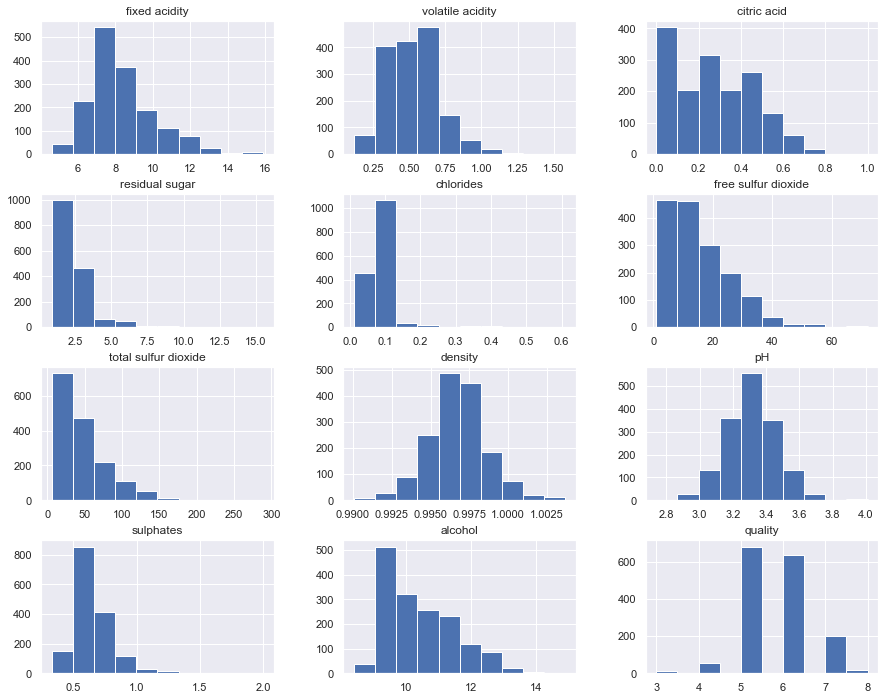
fixed acidity 0  
volatile acidity 0  
citric acid 0  
residual sugar 0  
chlorides 0  
free sulfur dioxide 0  
total sulfur dioxide 0  
density 0  
pH 0  
sulphates 0  
alcohol 0  
quality 0  
dtype: int64

There are no null values in the dataset hence I am going to skip the missing value analysis and missing value imputation part.

## EDA

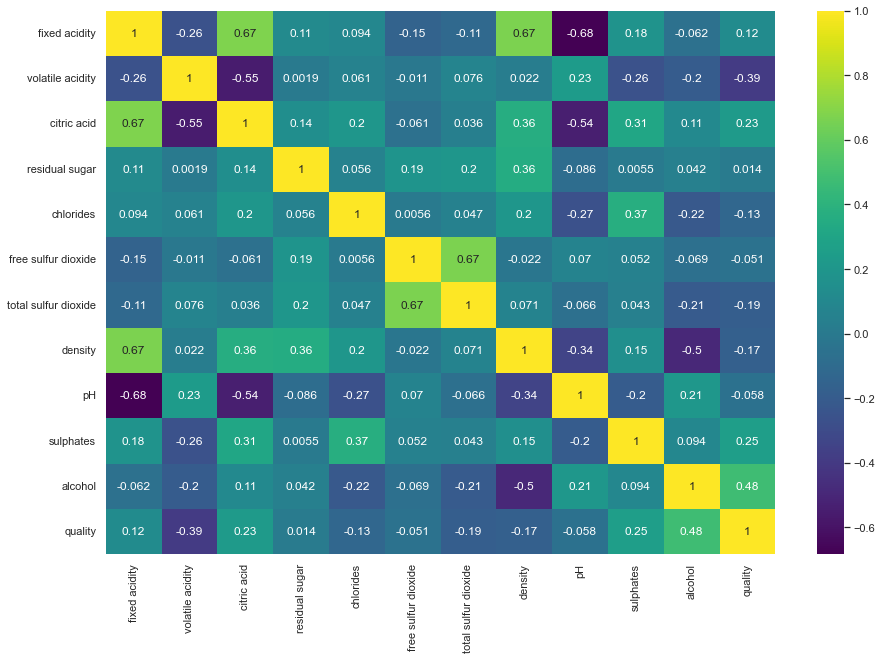
The Exploratory data analysis is done by plot some graphs and analyzing them

df.hist(bins=10,figsize=(15,12))  
plt.show()



plt.figure(figsize=(15,10))  
correlation = df.corr()  
sns.heatmap(correlation,annot=True,cmap='viridis')

<AxesSubplot: >



correlation['quality'].sort\_values(ascending=False)

quality 1.000000  
alcohol 0.476166  
sulphates 0.251397  
citric acid 0.226373  
fixed acidity 0.124052  
residual sugar 0.013732  
free sulfur dioxide -0.050656  
pH -0.057731  
chlorides -0.128907  
density -0.174919  
total sulfur dioxide -0.185100  
volatile acidity -0.390558  
Name: quality, dtype: float64

The above results shows the measure of correlation between quality and other columns. It can be deduced that alcohol has highest correlation value. After that we have sulphates, citric acid, fixed acidity and residual sugar.

## Converting Quality into category

***We convert the values in "rating" column to three catagories: inferior, fine and superior. So that, we can easily do multiclass classification.***

categor\_condn=[ (df['quality']>=7),  
 (df['quality']<=4)]  
rating=['superior','inferior']  
df['rating'] = np.select(categor\_condn,rating,default='fine')

### Feature Encoding

Here, we Encode the target feature using the Label Encoder class from the SciKit Learn Module. The LabelEncoder encodes the target labels with value between 0 and n\_classes-1. This transformer is used to encode target values, i.e. y, and not the input X.

from sklearn.preprocessing import LabelEncoder  
labelencoder\_y = LabelEncoder()  
df['rating'] = labelencoder\_y.fit\_transform(df['rating'])

X = df.drop('rating', axis = 1)  
y = df['rating'].values.reshape(-1,1)

**Here, We split the training and testing data with 3:10 ratio.**

#splitting the data into train and test  
from sklearn.model\_selection import train\_test\_split  
X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.3, random\_state=50)

### Feature Scaling

Here, we normalize the data using StandardScaler transformer of the SciKit Module. Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without normalization.

from sklearn.preprocessing import StandardScaler  
sc = StandardScaler()  
X\_train\_scale = sc.fit\_transform(X\_train)  
X\_test\_scale = sc.fit\_transform(X\_test)

# Predictions

We use three different ML Algorithms to determine which will be well suited for this dataset. We use Decision Tree, Random Forest and Support Vector Machine for the predictions.

## Decision Tree

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

### Cross validation

Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. Here, we use SkiKit Module for Cross validation.

DT REPORT  
CV Score: 0.8159105534105533  
Training set accuracy: 1.0  
Test set accuracy: 0.8166666666666667

Confusion matrix:

[[365, 9, 34],  
 [ 12, 1, 2],  
 [ 30, 1, 26]]

## Random Forest (Ensemble Learning)

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. It is an ensemble learning technique combining numerous classifiers to enhance a model's performance.

**Ensemble Learning**: Ensemble Learning uses multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.

RF REPORT  
CV Score: 0.8552364864864865  
Training set accuracy: 1.0  
Test set accuracy: 0.8833333333333333

Confusion matrix

[[395, 2, 11],  
 [ 15, 0, 0],  
 [ 28, 0, 29]]

## SVM

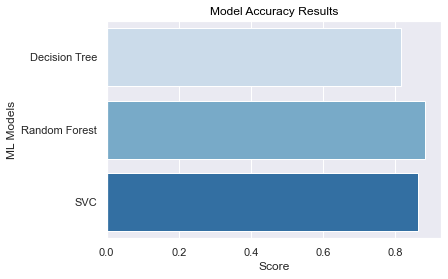
SVM is a powerful supervised algorithm that works best on smaller datasets but on complex ones. Support Vector Machine, abbreviated as SVM can be used for both regression and classification tasks, but generally, they work best in classification problems.

SVC REPORT  
CV Score: 0.8328989703989704  
Training set accuracy: 0.8525469168900804  
Test set accuracy: 0.8625

Confusion matrix:

[[393, 2, 13],  
 [ 15, 0, 0],  
 [ 36, 0, 21]]

**Comparing Models**



#### From the above graph, we can conclude that Random Forest has the highest accuracy among the three with 88%.

# Deployment

To deploy our project, first, we have to serialize our model using the Pickle Module in python. Then it is deserialized in our app script using Flask.

**Pickle**: The pickle module implements binary protocols for serializing and de-serializing a Python object structure. “Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy.

import pickle  
pickle.dump(model\_RF, open("model.pkl","wb"))

At Last, we use flask to generate an API for our project. That API can be tested in Postmen API Testing. The app creates Local Host with port number 5000. We can use that to send a POST Request. The API raw data uses data in form of JSON. As JSON is used in deserializing the data. When we send a post request the API sends requests to our local host and then the script runs and complete the request and send the output back to the request sender. As a result, we get a JSON output giving us the Predicted value.

Thank you!