

Name: Rohan Khatri

Batch Code: LISUM39

Submission Date: 26 November 2024

Submitted To:

<https://github.com/rohankhatri7/DataGlacier-Internship/tree/main/Week4>

## Step 1: Loading the Data

The wine dataset from `sklearn.datasets` has 3 class specifications: **class\_0**, **class\_1**, and **class\_2**. The following thirteen attributes are used to determine which wine belongs in which category: alcohol, malic acid, ash, alkalinity of ash, magnesium, total phenols, flavonoids, nonflavanoid phenols, proanthocyanins, color intensity, hue, OD280/OD315 of diluted wins, and proline.

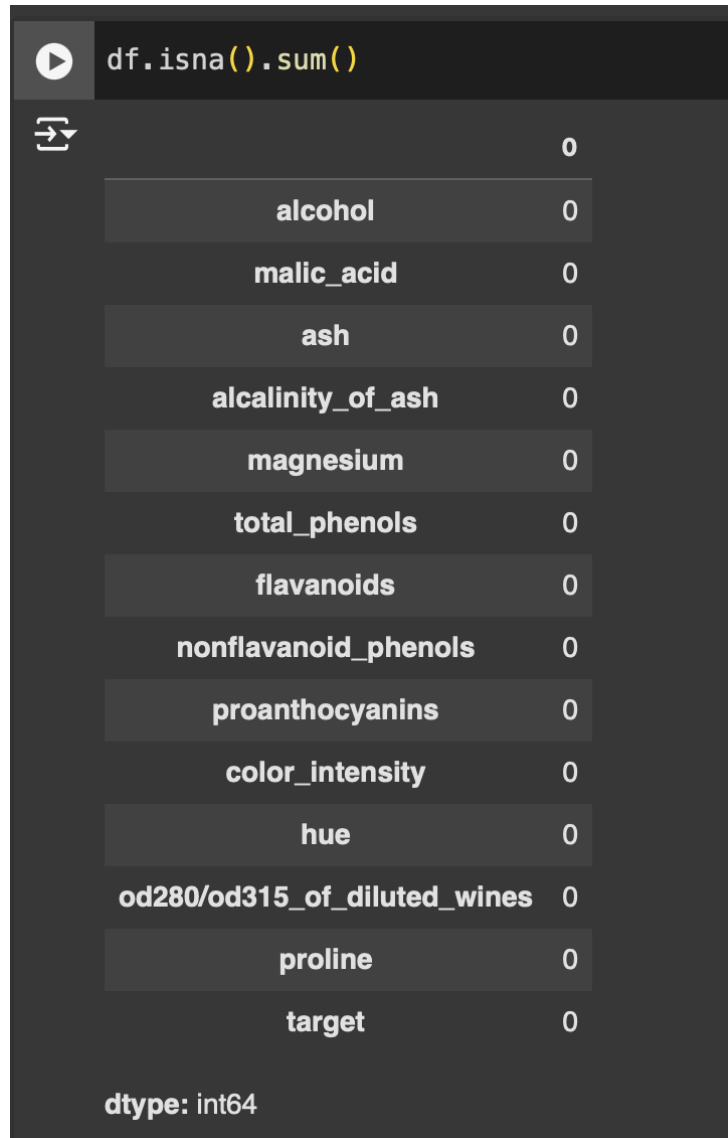
```
from sklearn.datasets import load_wine

data = load_wine() # Load wine dataset
df = pd.DataFrame(data=data.data, columns=data.feature_names)
df['target'] = pd.Categorical.from_codes(data.target, categories=data.target_names)
df
```

	alcohol	malic_acid	ash	alkalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	
...	...	...	...	...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	0.52	1.06	7.70	0.64	
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	0.43	1.41	7.30	0.70	
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	0.43	1.35	10.20	0.59	
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	0.53	1.46	9.30	0.60	
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	0.56	1.35	9.20	0.61	

178 rows x 14 columns

The dataset has no null values, so we can move forward with data analysis and modeling without having to handle the missing data.



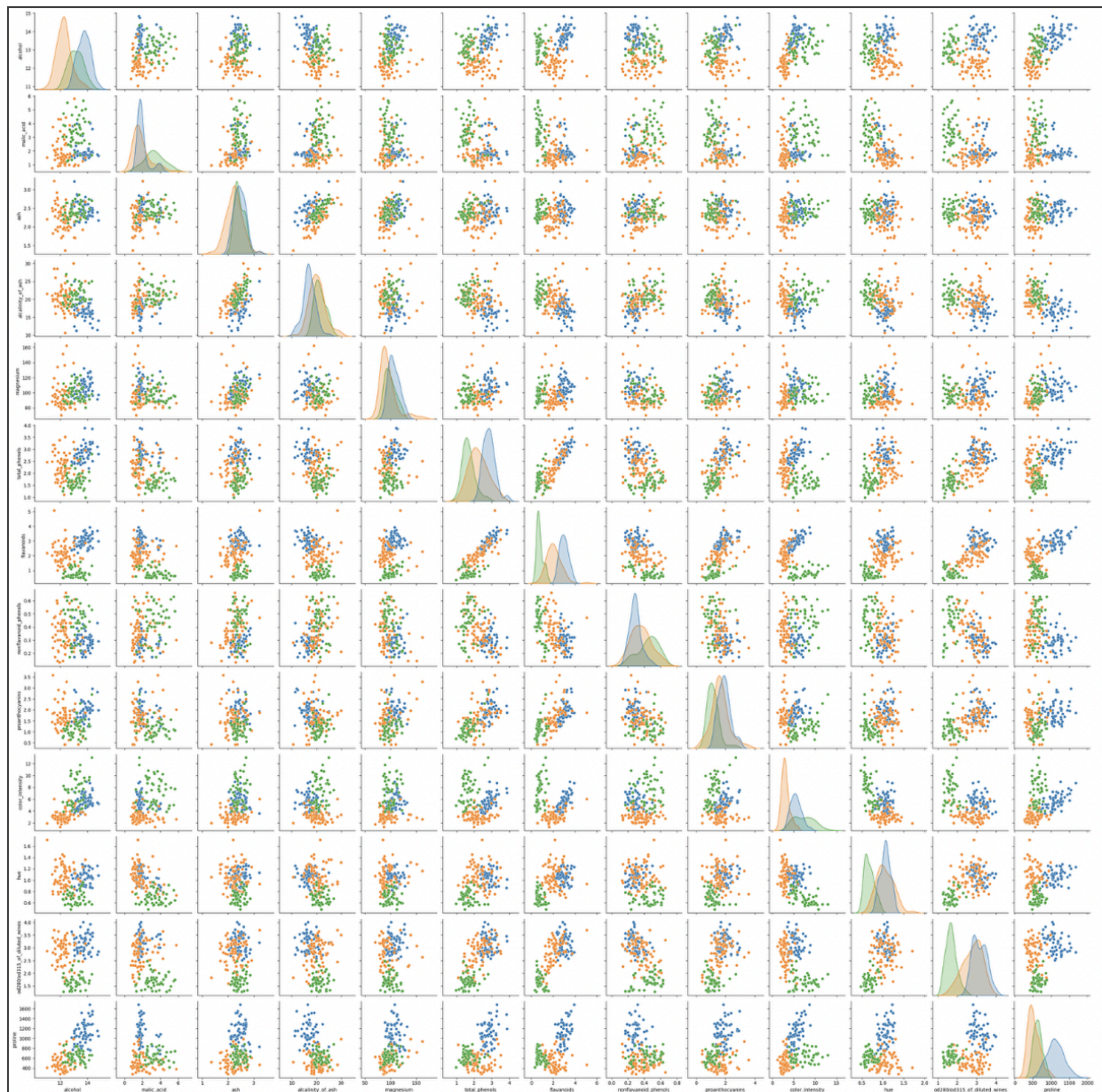
A screenshot of a Jupyter Notebook cell. The code `df.isna().sum()` is entered in the input area. Below the code, the output is displayed as a table with 15 rows, each representing a column of the dataset and its corresponding count of null values. All counts are 0. At the bottom of the output, it says `dtype: int64`.

	0
alcohol	0
malic_acid	0
ash	0
alcalinity_of_ash	0
magnesium	0
total_phenols	0
flavanoids	0
nonflavanoid_phenols	0
proanthocyanins	0
color_intensity	0
hue	0
od280/od315_of_diluted_wines	0
proline	0
target	0

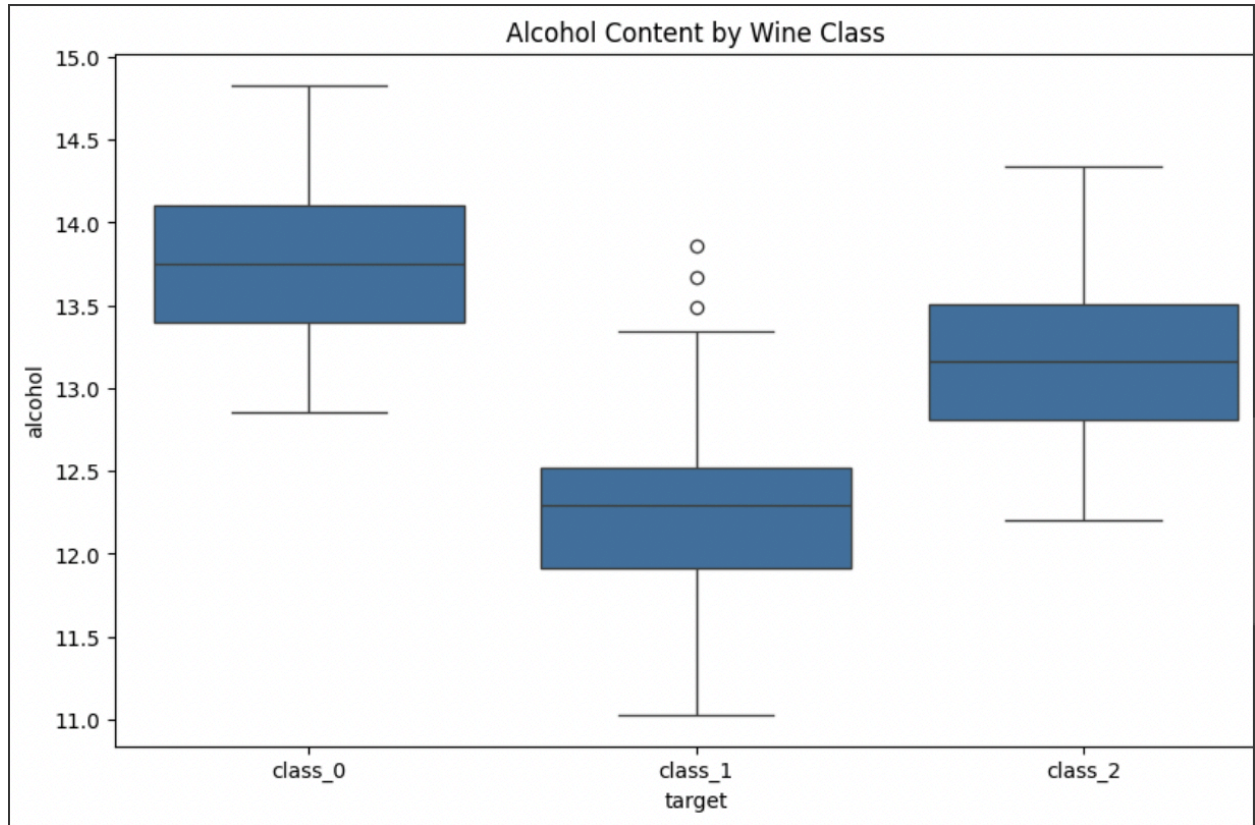
dtype: int64

## Step 2: EDA Visualizations and Model Training

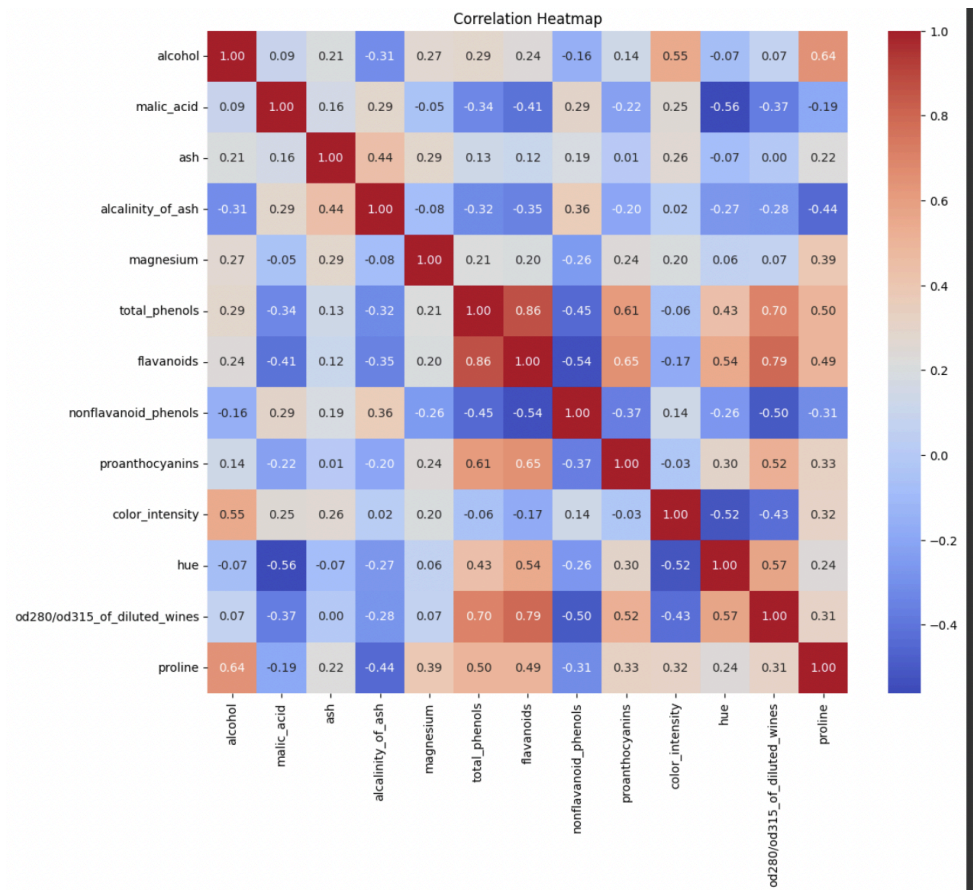
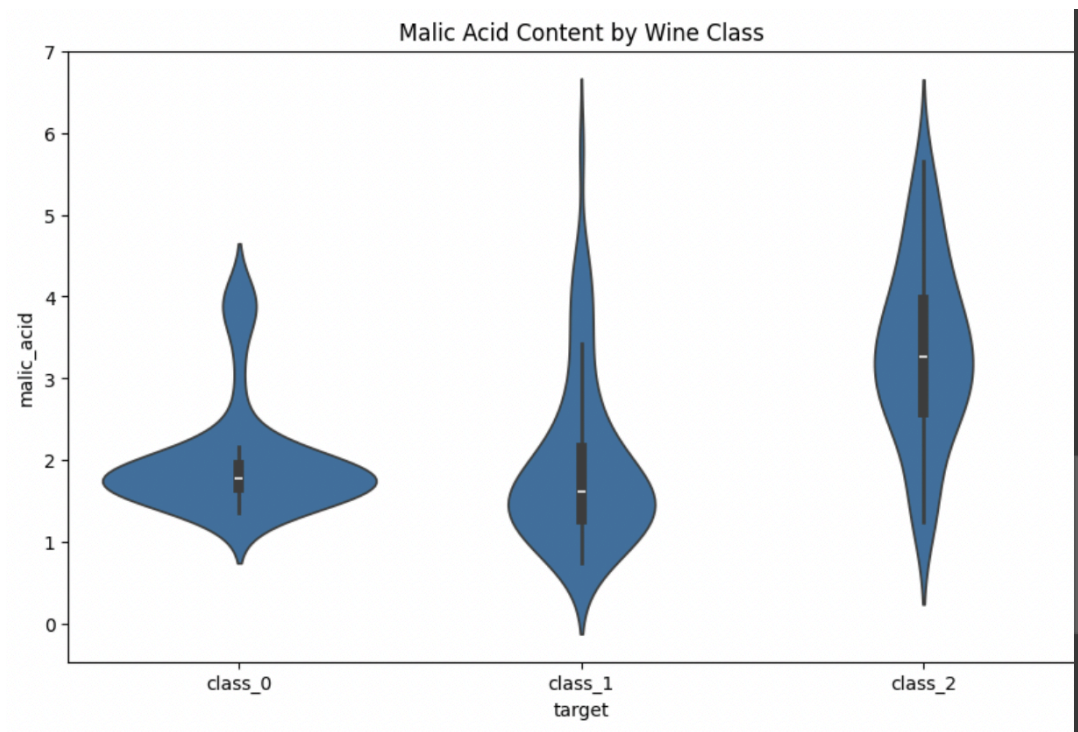
The dataset shows clear separations between the 3 classes in some feature combinations which suggests that the data is well-suited for classification tasks.



The median alcohol content varies between the 3 classes with class\_0 having the highest and class\_1 having the lowest. Class\_1 has the least variability and alcohol content can serve as an important feature for distinguishing between classes.



Class\_0 is the most distinct as it has the lowest spread of malic acid values while also possessing low variability and concentration around 2.





Import the necessary libraries, load the wine data set, create a dataframe to make the dataset easier to work with and the target column represents the respective wine classes.

```
from flask import Flask, request, render_template
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
import pickle

from sklearn.datasets import load_wine

data = load_wine() # Load wine dataset
df = pd.DataFrame(data=data.data, columns=data.feature_names)
df['target'] = pd.Categorical.from_codes(data.target, categories=data.target_names)
```

Built the model

```
X = df.drop(['target'], axis=1)
y = df['target']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, train_size=0.70, test_size=0.30, random_state=0)
print(X_train.shape, X_test.shape)
```

Classification model used

```
randomForestClassifier = RandomForestClassifier(n_estimators=100, random_state=0)
randomForestClassifier.fit(X_train, y_train)
```

Metadata is saved by using pickle for the deployment.

```
with open('random_forest.pkl', 'wb') as file:
    pickle.dump(randomForestClassifier, file)
```

## Step 3: Flask Deployment

The Flask framework initialized earlier is used to deploy the trained model by loading the metadata stored in pickle files. These pickle files contain the trained model, feature names, and any necessary preprocessing information from the training process. The Flask web application enables users to input wine characteristics (e.g., alcohol content, malic acid, etc.) and receive an accurate prediction of the wine's class (e.g., class 0, class 1, or class 2). The application supports real-time predictions, providing immediate feedback based on the input data.

```
app = Flask(__name__, template_folder='templates')

with open('random_forest.pkl', 'rb') as file:
    randomForestClassifier = pickle.load(file)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    int_features = [float(x) for x in request.form.values()]
    final_features = [np.array(int_features)]

    rfc_prediction = randomForestClassifier.predict(final_features)[0]

    return render_template('index.html',
                           rfc_prediction=rfc_prediction)

if __name__ == "__main__":
    app.run(port=5000, debug=True)
```

## Step 4: Frontend Development

A HTML form was used to capture wine characteristics such as alcohol content, malic acid, flavanoids, and other relevant features. The form predicts wine classification based on the values inputted.

### Wine Quality Prediction

Alcohol:

Malic Acid:

Ash:

Alcalinity of Ash:

Magnesium:

Total Phenols:

Flavanoids:

Nonflavanoid Phenols:

Proanthocyanins:

Color Intensity:

Hue:

OD280/OD315 of diluted wines:

Proline:

Predict



# Wine Quality Prediction

Alcohol:

Malic Acid:

Ash:

Alcalinity of Ash:

Magnesium:

Total Phenols:

Flavanoids:

Nonflavanoid Phenols:

Proanthocyanins:

Color Intensity:

Hue:

OD280/OD315 of diluted wines:

Proline:

Predict

**Prediction: class\_1**