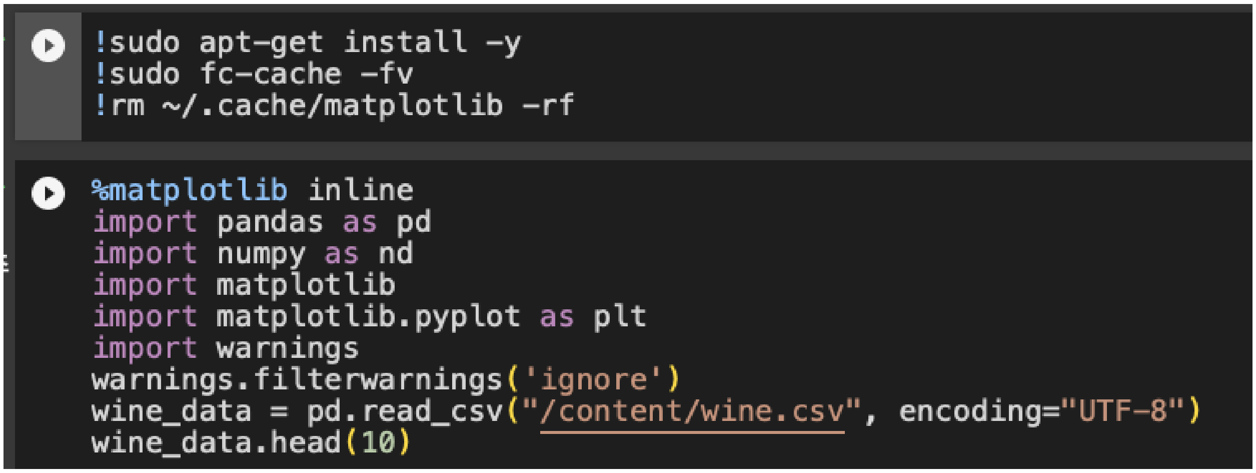
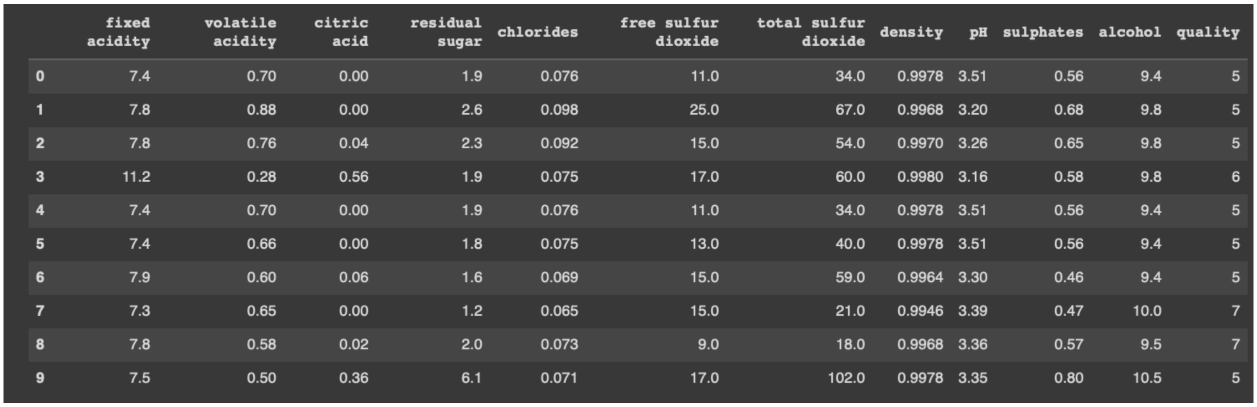
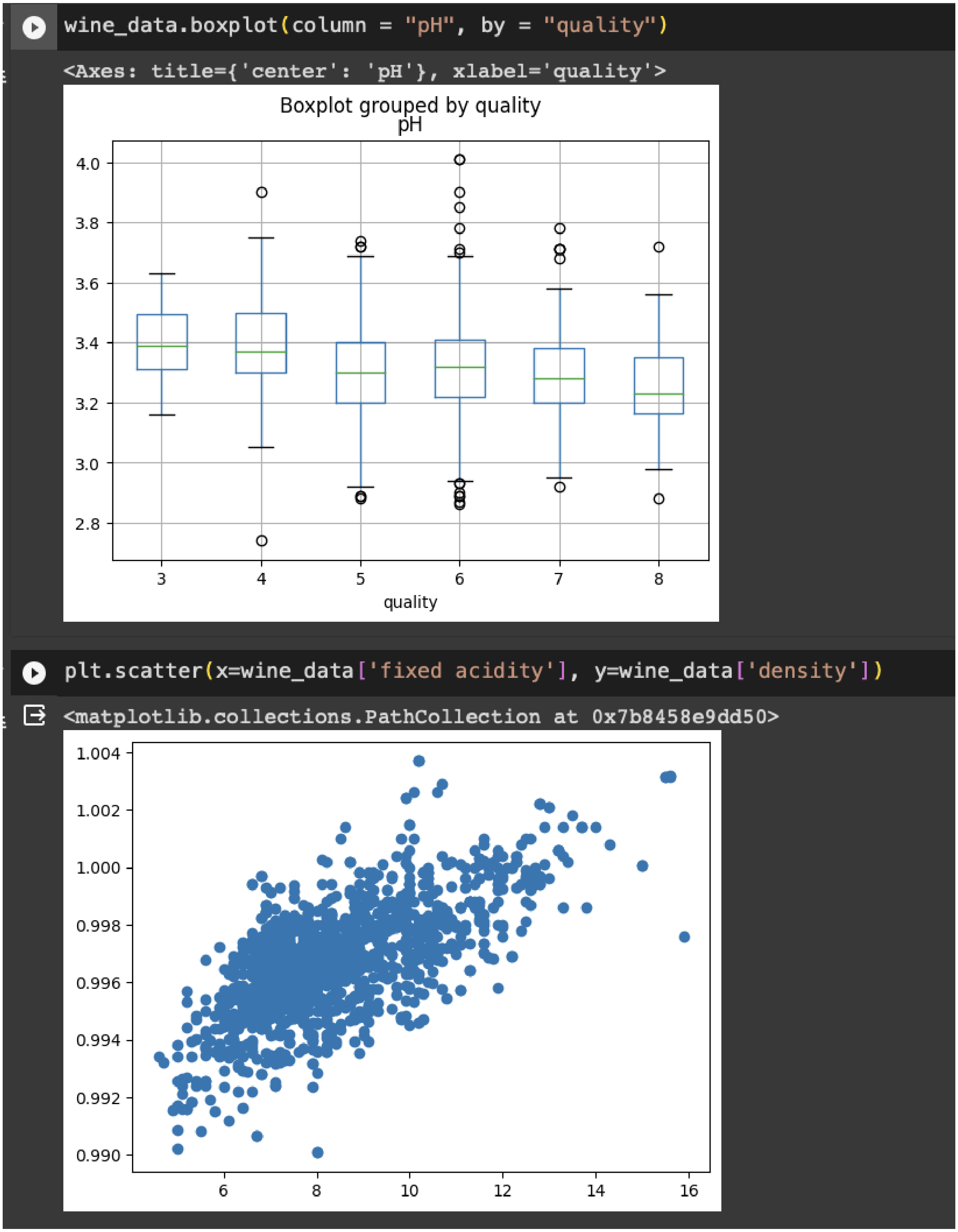


```
!sudo apt-get install -y
!sudo fc-cache -fv
!rm ~/.cache/matplotlib -rf
```

```
%matplotlib inline
import pandas as pd
import numpy as nd
import matplotlib
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
wine_data = pd.read_csv("/content/wine.csv", encoding="UTF-8")
wine_data.head(10)
```



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4	5
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4	5
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0	7
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5	7
9	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5	5

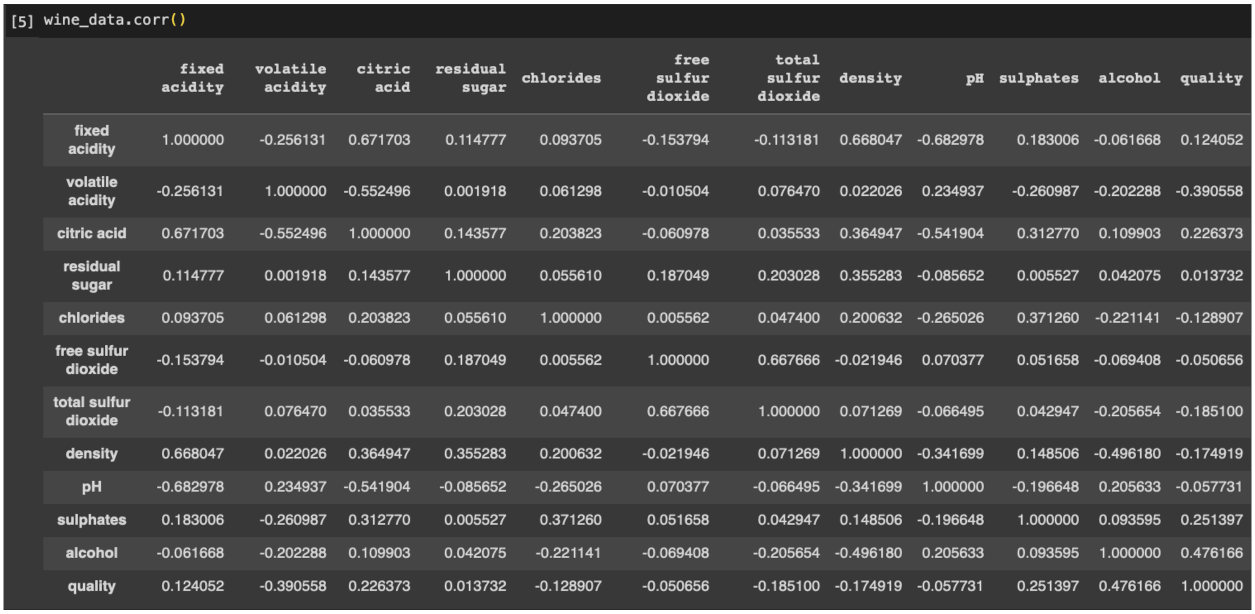


```
wine_data.boxplot(column = "pH", by = "quality")
```

```
<Axes: title={'center': 'pH'}, xlabel='quality'>
```

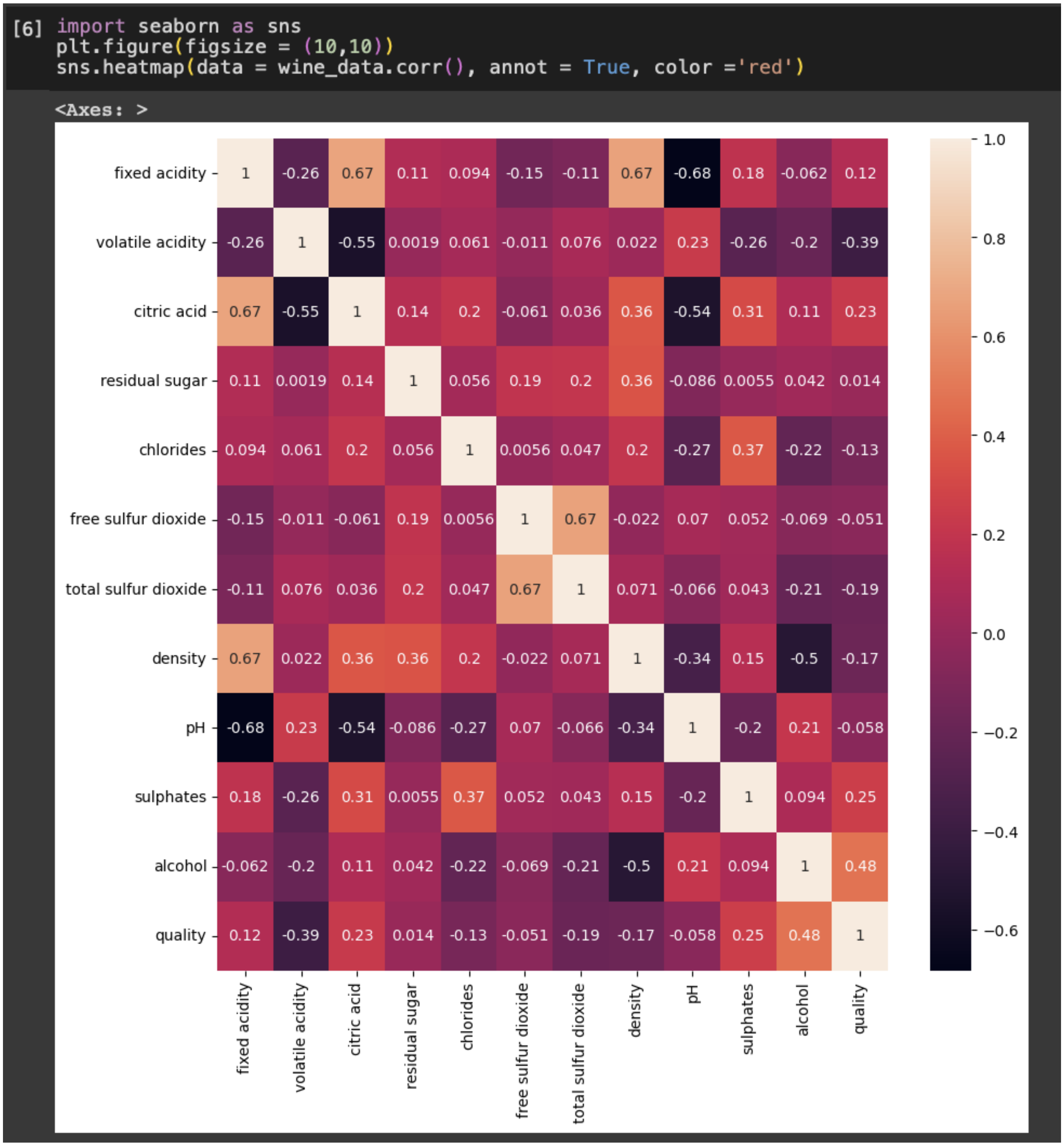
```
plt.scatter(x=wine_data['fixed acidity'], y=wine_data['density'])
```

```
<matplotlib.collections.PathCollection at 0x7b8458e9dd50>
```



```
[5] wine_data.corr()
```

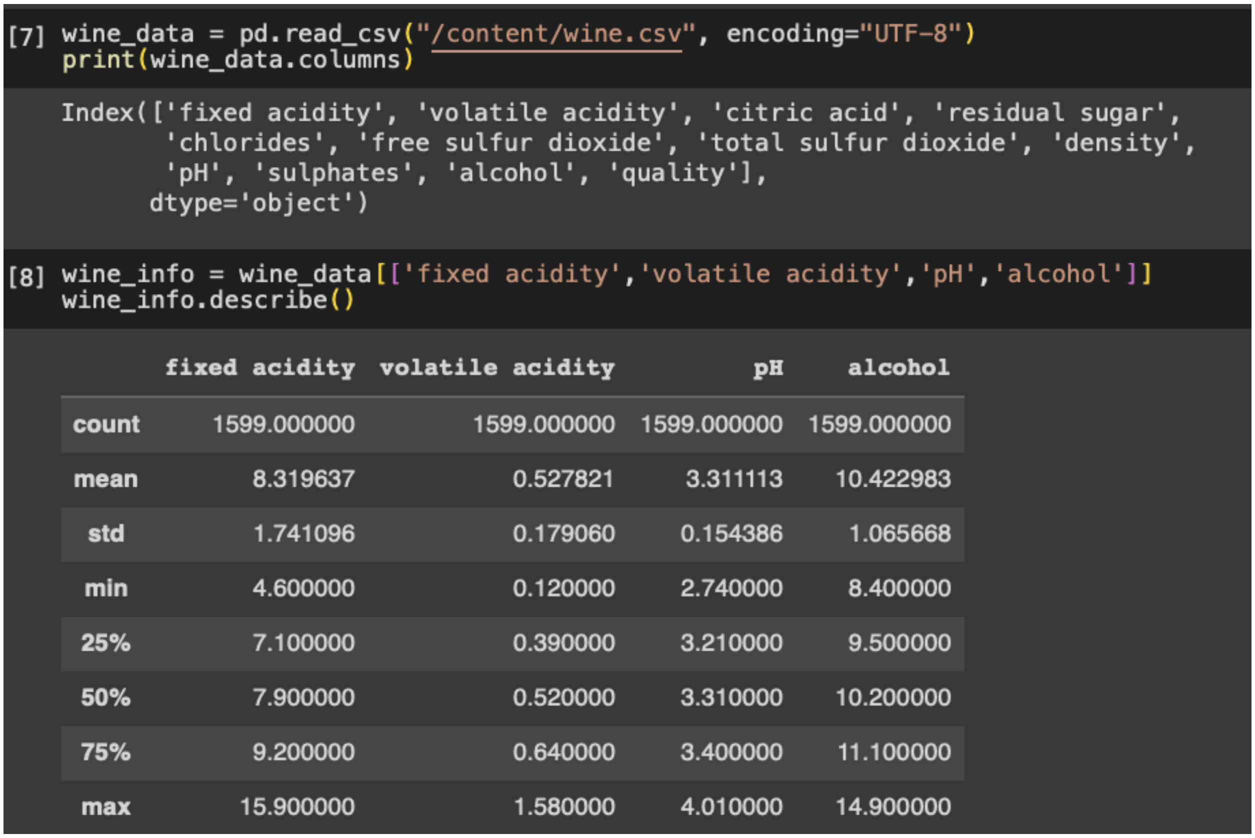
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181	0.668047	-0.682978	0.183006	-0.061668	0.124052
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.022026	0.234937	-0.260987	-0.202288	-0.390558
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.364947	-0.541904	0.312770	0.109903	0.226373
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028	0.355283	-0.085652	0.005527	0.042075	0.013732
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400	0.200632	-0.265026	0.371260	-0.221141	-0.128907
free sulfur dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666	-0.021946	0.070377	0.051658	-0.069408	-0.050656
total sulfur dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000	0.071269	-0.066495	0.042947	-0.205654	-0.185100
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269	1.000000	-0.341699	0.148506	-0.496180	-0.174919
pH	-0.682978	0.234937	-0.541904	-0.085652	-0.265026	0.070377	-0.066495	-0.341699	1.000000	-0.196648	0.205633	-0.057731
sulphates	0.183006	-0.260987	0.312770	0.005527	0.371260	0.051658	0.042947	0.148506	-0.196648	1.000000	0.093595	0.251397
alcohol	-0.061668	-0.202288	0.109903	0.042075	-0.221141	-0.069408	-0.205654	-0.496180	0.205633	0.093595	1.000000	0.476166
quality	0.124052	-0.390558	0.226373	0.013732	-0.128907	-0.050656	-0.185100	-0.174919	-0.057731	0.251397	0.476166	1.000000



```
[6] import seaborn as sns
    plt.figure(figsize = (10,10))
    sns.heatmap(data = wine_data.corr(), annot = True, color ='red')
```

```
<Axes: >
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1	-0.26	0.67	0.11	0.094	-0.15	-0.11	0.67	-0.68	0.18	-0.062	0.12
volatile acidity	-0.26	1	-0.55	0.0019	0.061	-0.011	0.076	0.022	0.23	-0.26	-0.2	-0.39
citric acid	0.67	-0.55	1	0.14	0.2	-0.061	0.036	0.36	-0.54	0.31	0.11	0.23
residual sugar	0.11	0.0019	0.14	1	0.056	0.19	0.2	0.36	-0.086	0.0055	0.042	0.014
chlorides	0.094	0.061	0.2	0.056	1	0.0056	0.047	0.2	-0.27	0.37	-0.22	-0.13
free sulfur dioxide	-0.15	-0.011	-0.061	0.19	0.0056	1	0.67	-0.022	0.07	0.052	-0.069	-0.051
total sulfur dioxide	-0.11	0.076	0.036	0.2	0.047	0.67	1	0.071	-0.066	0.043	-0.21	-0.19
density	0.67	0.022	0.36	0.36	0.2	-0.022	0.071	1	-0.34	0.15	-0.5	-0.17
pH	-0.68	0.23	-0.54	-0.086	-0.27	0.07	-0.066	-0.34	1	-0.2	0.21	-0.058
sulphates	0.18	-0.26	0.31	0.0055	0.37	0.052	0.043	0.15	-0.2	1	0.094	0.25
alcohol	-0.062	-0.2	0.11	0.042	-0.22	-0.069	-0.21	-0.5	0.21	0.094	1	0.48
quality	0.12	-0.39	0.23	0.014	-0.13	-0.051	-0.19	-0.17	-0.058	0.25	0.48	1

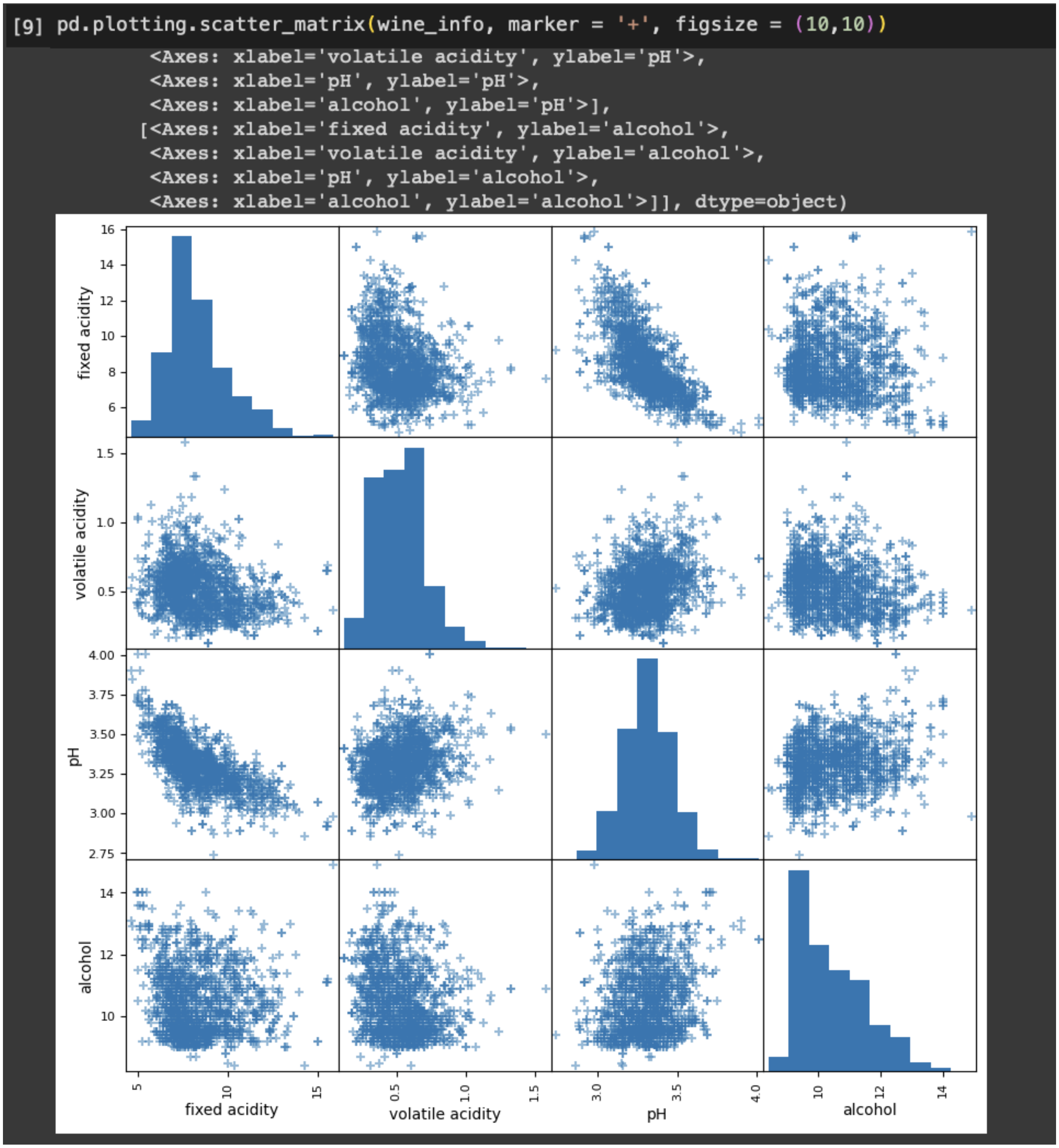


```
[7] wine_data = pd.read_csv("/content/wine.csv", encoding="UTF-8")
    print(wine_data.columns)
```

```
    Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
           'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
           'pH', 'sulphates', 'alcohol', 'quality'],
          dtype='object')
```

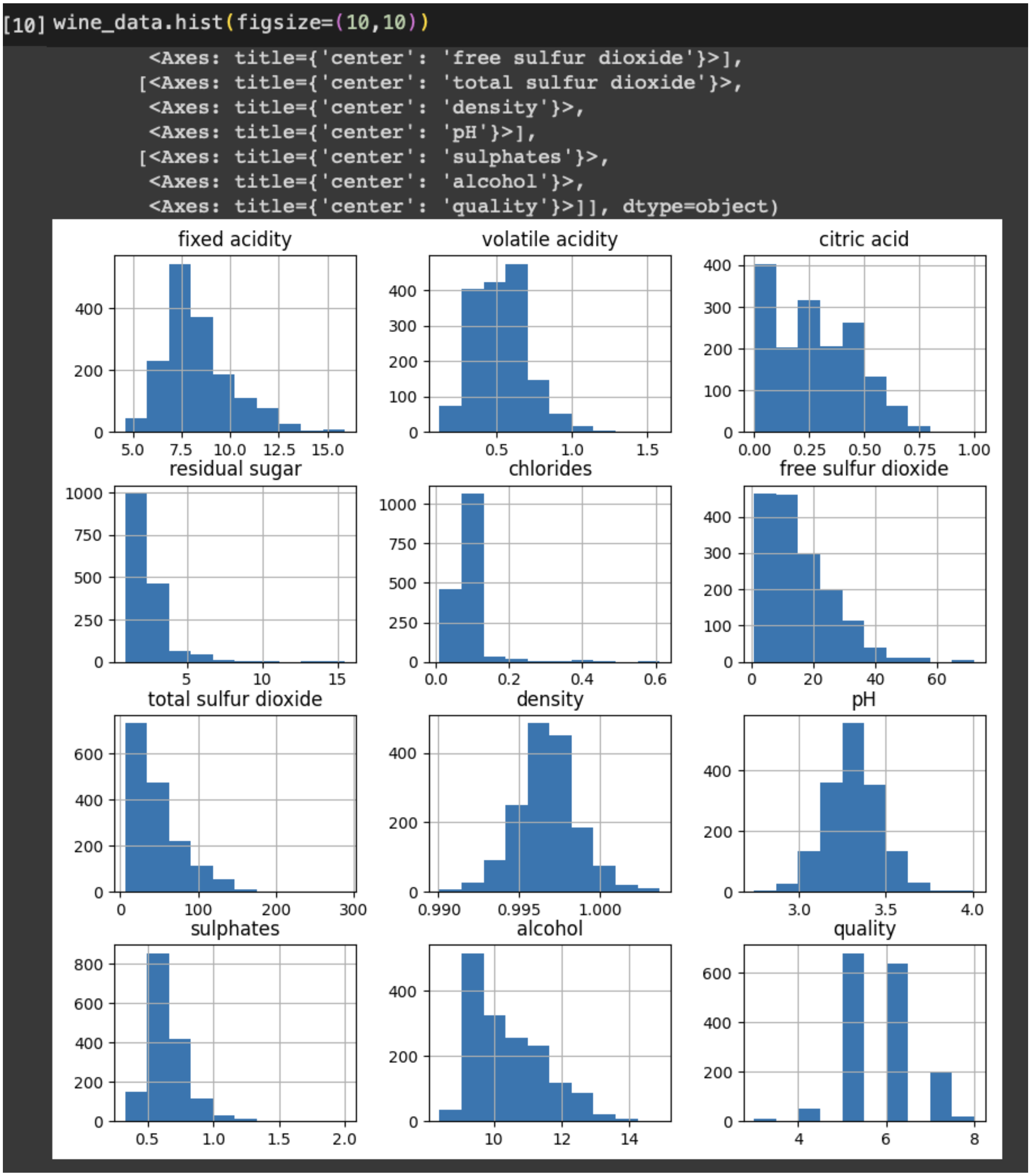
```
[8] wine_info = wine_data[['fixed acidity','volatile acidity','pH','alcohol']]
    wine_info.describe()
```

	fixed acidity	volatile acidity	pH	alcohol
count	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	3.311113	10.422983
std	1.741096	0.179060	0.154386	1.065668
min	4.600000	0.120000	2.740000	8.400000
25%	7.100000	0.390000	3.210000	9.500000
50%	7.900000	0.520000	3.310000	10.200000
75%	9.200000	0.640000	3.400000	11.100000
max	15.900000	1.580000	4.010000	14.900000



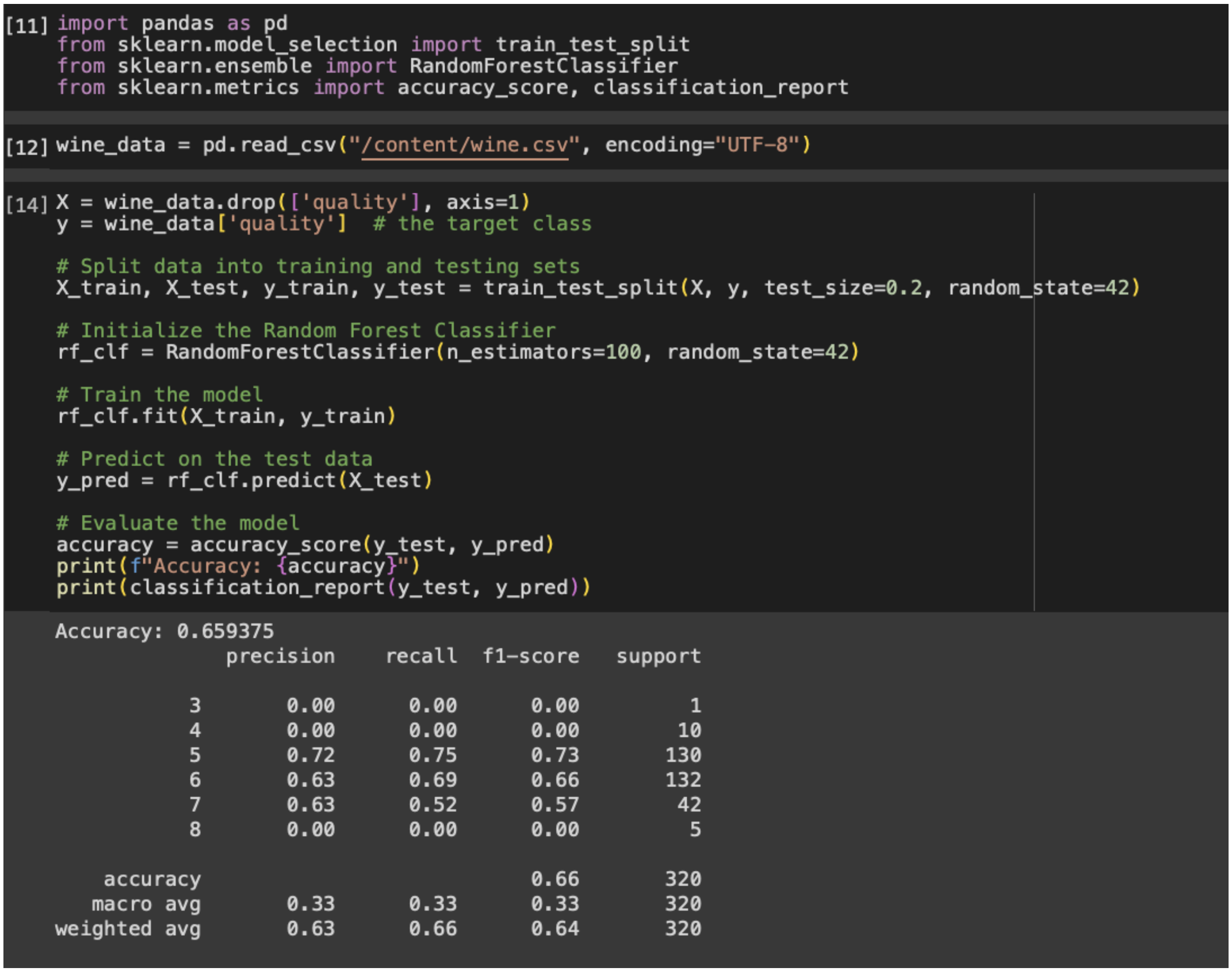
```
[9] pd.plotting.scatter_matrix(wine_info, marker = '+', figsize = (10,10))
```

```
          <Axes: xlabel='volatile acidity', ylabel='pH'>,
          <Axes: xlabel='pH', ylabel='pH'>,
          <Axes: xlabel='alcohol', ylabel='pH'>],
         [<Axes: xlabel='fixed acidity', ylabel='alcohol'>,
          <Axes: xlabel='volatile acidity', ylabel='alcohol'>,
          <Axes: xlabel='pH', ylabel='alcohol'>,
          <Axes: xlabel='alcohol', ylabel='alcohol'>]], dtype=object)
```



```
[10] wine_data.hist(figsize=(10,10))
```

```
           <Axes: title={'center': 'free sulfur dioxide'}>],
          [<Axes: title={'center': 'total sulfur dioxide'}>,
           <Axes: title={'center': 'density'}>,
           <Axes: title={'center': 'pH'}>],
          [<Axes: title={'center': 'sulphates'}>,
           <Axes: title={'center': 'alcohol'}>,
           <Axes: title={'center': 'quality'}>]], dtype=object)
```

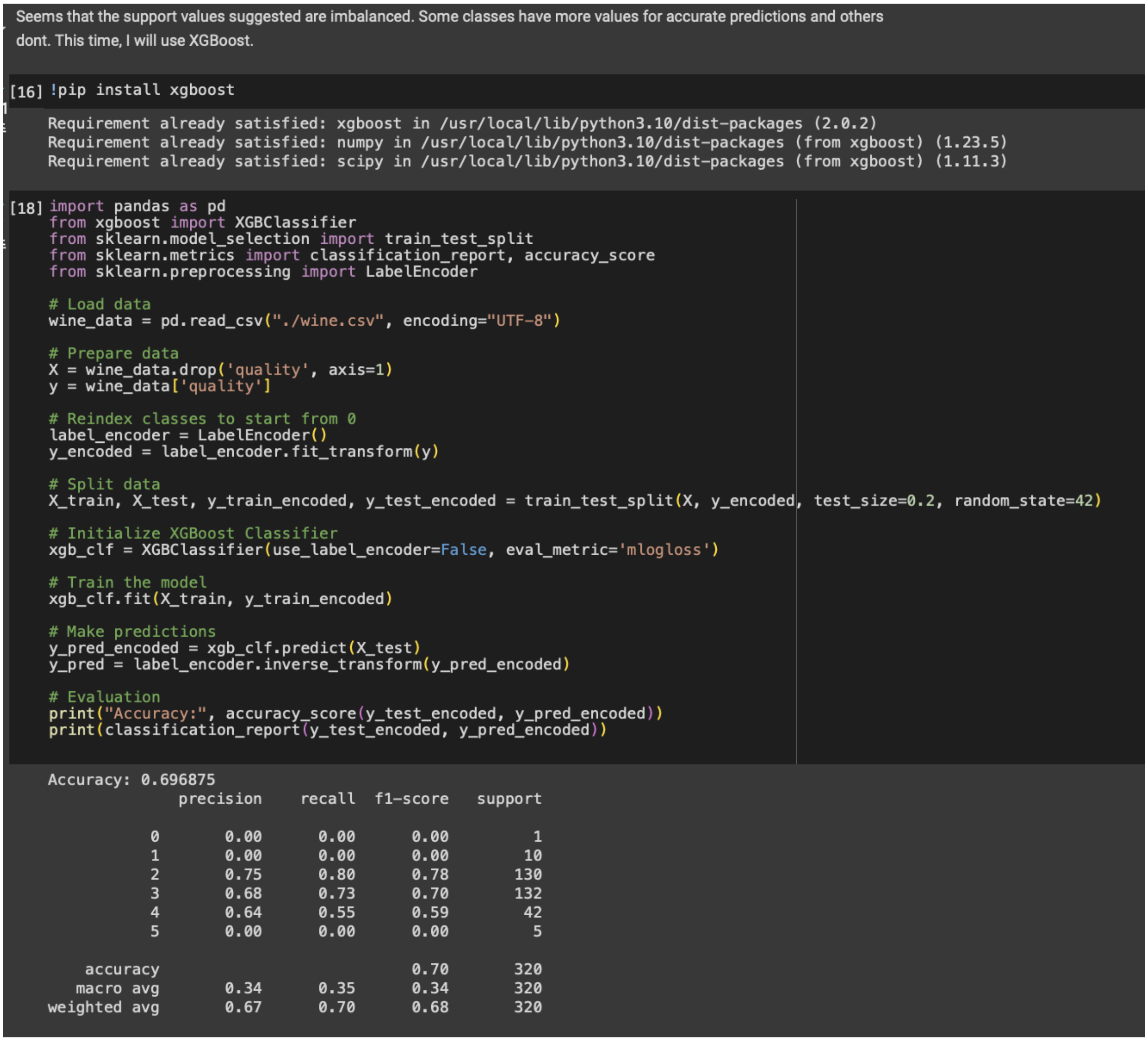


```
[11] import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score, classification_report
```

```
[12] wine_data = pd.read_csv("/content/wine.csv", encoding="UTF-8")
```

```
[14] X = wine_data.drop(['quality'], axis=1)
     y = wine_data['quality']  # the target class

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

     # Initialize the Random Forest Classifier
     rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)

     # Train the model
     rf_clf.fit(X_train, y_train)

     # Predict on the test data
     y_pred = rf_clf.predict(X_test)

     # Evaluate the model
     accuracy = accuracy_score(y_test, y_pred)
     print(f"Accuracy: {accuracy}")
     print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.659375
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00        10
           5       0.72      0.75      0.73       130
           6       0.63      0.69      0.66       132
           7       0.63      0.52      0.57        42
           8       0.00      0.00      0.00         5

    accuracy                           0.66       320
   macro avg       0.33      0.33      0.33       320
weighted avg       0.63      0.66      0.64       320
```



Seems that the support values suggested are imbalanced. Some classes have more values for accurate predictions and others dont. This time, I will use XGBoost.

```
[16] !pip install xgboost
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (2.0.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.23.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.11.3)
```

```
[18] import pandas as pd
     from xgboost import XGBClassifier
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import classification_report, accuracy_score
     from sklearn.preprocessing import LabelEncoder

     # Load data
     wine_data = pd.read_csv("./wine.csv", encoding="UTF-8")

     # Prepare data
     X = wine_data.drop('quality', axis=1)
     y = wine_data['quality']

     # Reindex classes to start from 0
     label_encoder = LabelEncoder()
     y_encoded = label_encoder.fit_transform(y)

     # Split data
     X_train, X_test, y_train_encoded, y_test_encoded = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

     # Initialize XGBoost Classifier
     xgb_clf = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')

     # Train the model
     xgb_clf.fit(X_train, y_train_encoded)

     # Make predictions
     y_pred_encoded = xgb_clf.predict(X_test)
     y_pred = label_encoder.inverse_transform(y_pred_encoded)

     # Evaluation
     print("Accuracy:", accuracy_score(y_test_encoded, y_pred_encoded))
     print(classification_report(y_test_encoded, y_pred_encoded))
```

```
Accuracy: 0.696875
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.00      0.00      0.00        10
           2       0.75      0.80      0.78       130
           3       0.68      0.73      0.70       132
           4       0.64      0.55      0.59        42
           5       0.00      0.00      0.00         5

    accuracy                           0.70       320
   macro avg       0.34      0.35      0.34       320
weighted avg       0.67      0.70      0.68       320
```