explainable-ai-shap

December 30, 2023

[17]:
```
pip install shap
```

```
Requirement already satisfied: shap in /opt/conda/lib/python3.10/site-packages
(0.44.0)
Requirement already satisfied: numpy in /opt/conda/lib/python3.10/site-packages
(from shap) (1.24.3)
Requirement already satisfied: scipy in /opt/conda/lib/python3.10/site-packages
(from shap) (1.11.4)
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.10/site-
packages (from shap) (1.2.2)
Requirement already satisfied: pandas in /opt/conda/lib/python3.10/site-packages
(from shap) (2.0.3)
Requirement already satisfied: tqdm>=4.27.0 in /opt/conda/lib/python3.10/site-
packages (from shap) (4.66.1)
Requirement already satisfied: packaging>20.9 in /opt/conda/lib/python3.10/site-
packages (from shap) (21.3)
Requirement already satisfied: slicer==0.0.7 in /opt/conda/lib/python3.10/site-
packages (from shap) (0.0.7)
Requirement already satisfied: numba in /opt/conda/lib/python3.10/site-packages
(from shap) (0.57.1)
Requirement already satisfied: cloudpickle in /opt/conda/lib/python3.10/site-
packages (from shap) (2.2.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/opt/conda/lib/python3.10/site-packages (from packaging>20.9->shap) (3.0.9)
Requirement already satisfied: llvmlite<0.41,>=0.40.0dev0 in
/opt/conda/lib/python3.10/site-packages (from numba->shap) (0.40.1)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.10/site-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.10/site-
packages (from pandas->shap) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in /opt/conda/lib/python3.10/site-
packages (from pandas->shap) (2023.3)
Requirement already satisfied: joblib>=1.1.1 in /opt/conda/lib/python3.10/site-
packages (from scikit-learn->shap) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.10/site-packages (from scikit-learn->shap) (3.2.0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.10/site-
packages (from python-dateutil>=2.8.2->pandas->shap) (1.16.0)
```

Note: you may need to restart the kernel to use updated packages.

[18]:
```
# importing the libraries

import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report

# importing shap
import shap
```

[19]:
```
# import the dataset
df = pd.read_csv('/kaggle/input/wine-dataset/wine.csv')
df.head()
```

[19]:

	Class	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	\
0	1	14.23	1.71	2.43	15.6	127	
1	1	13.20	1.78	2.14	11.2	100	
2	1	13.16	2.36	2.67	18.6	101	
3	1	14.37	1.95	2.50	16.8	113	
4	1	13.24	2.59	2.87	21.0	118	

	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	\
0	2.80	3.06	0.28	2.29	
1	2.65	2.76	0.26	1.28	
2	2.80	3.24	0.30	2.81	
3	3.85	3.49	0.24	2.18	
4	2.80	2.69	0.39	1.82	

	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
0	5.64	1.04	3.92	1065
1	4.38	1.05	3.40	1050
2	5.68	1.03	3.17	1185
3	7.80	0.86	3.45	1480
4	4.32	1.04	2.93	735

[20]:
```
# checking the dimensions of the dataframe
df.shape
```

[20]: (178, 14)

[21]:
```
# checking for missing values
df.isnull().sum()
```

[21]:
```
Class                         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
dtype: int64
```

[22]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Class                         178 non-null    int64
 1   Alcohol                       178 non-null    float64
 2   Malic acid                    178 non-null    float64
 3   Ash                           178 non-null    float64
 4   Alcalinity of ash             178 non-null    float64
 5   Magnesium                     178 non-null    int64
 6   Total phenols                 178 non-null    float64
 7   Flavanoids                    178 non-null    float64
 8   Nonflavanoid phenols          178 non-null    float64
 9   Proanthocyanins               178 non-null    float64
 10  Color intensity               178 non-null    float64
 11  Hue                           178 non-null    float64
 12  OD280/OD315 of diluted wines  178 non-null    float64
 13  Proline                       178 non-null    int64
dtypes: float64(11), int64(3)
memory usage: 19.6 KB
```

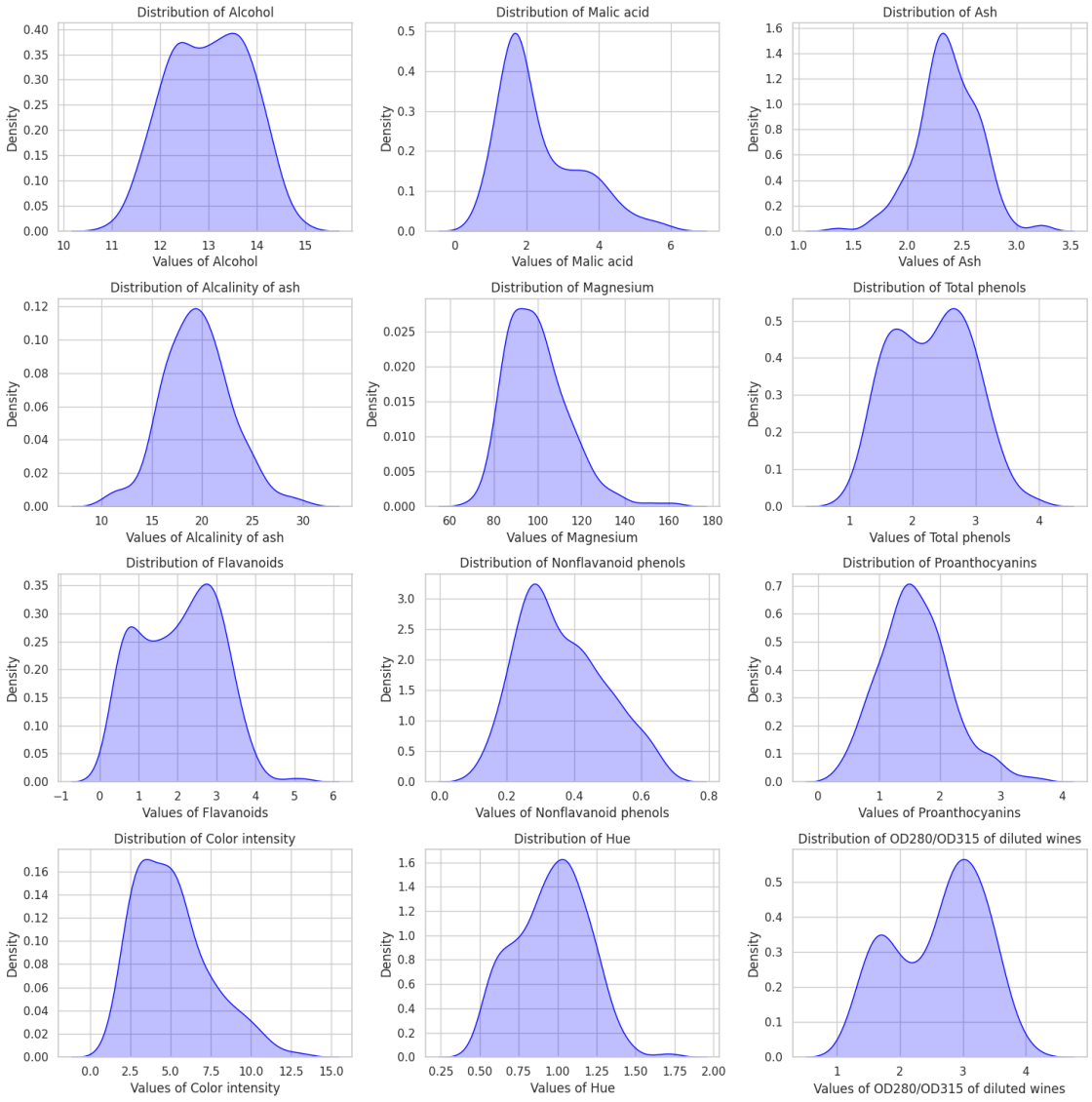
[23]:
```
df.dtypes
```

[23]:
```
Class                           int64
Alcohol                       float64
```

```
Malic acid                      float64
Ash                             float64
Alcalinity of ash               float64
Magnesium                         int64
Total phenols                   float64
Flavanoids                      float64
Nonflavanoid phenols            float64
Proanthocyanins                 float64
Color intensity                 float64
Hue                             float64
OD280/OD315 of diluted wines    float64
Proline                           int64
dtype: object
```

[30]:
```
sns.set(style="whitegrid")

# List of variables to plot
variables_to_plot = ['Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash',␣
 ↪'Magnesium',
                     'Total phenols', 'Flavanoids', 'Nonflavanoid phenols',␣
 ↪'Proanthocyanins',
                     'Color intensity', 'Hue', 'OD280/OD315 of diluted wines']

# Remove extra space in column names
df.columns = df.columns.str.strip()

# Calculate the number of rows and columns for the subplot grid
num_variables = len(variables_to_plot)
num_rows = (num_variables - 1) // 3 + 1
num_cols = min(num_variables, 3)

# Create subplots with a dynamic grid and vertical spacing
fig, axes = plt.subplots(nrows=num_rows, ncols=num_cols, figsize=(15, 15))

# Flatten the axes for easy iteration
axes = axes.flatten()

# Loop through variables and plot KDE on each subplot
for i, variable in enumerate(variables_to_plot):
    sns.kdeplot(df[variable], ax=axes[i], color='blue', fill=True,␣
 ↪common_norm=False)
    axes[i].set_title(f'Distribution of {variable}')
    axes[i].set_xlabel(f'Values of {variable}')
    axes[i].set_ylabel('Density')

# Adjust layout for better spacing with vertical and horizontal spacing
plt.tight_layout(h_pad=1)
```

```
# Show the plot
plt.show()
```



1 Model Building

[32]:
```
# splitting the data into independent and dependent variables
x = df.drop(columns=['Class'])
y = df['Class']
```

[33]:
```
# diving the dataset into training and testing sets
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.7,␣
 ↪random_state=42)
```

[34]:
```
# building the model
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier()
clf.fit(x_train, y_train)
```

[34]:
```
RandomForestClassifier()
```

[35]:
```
# importing the necessary libraries
from sklearn.model_selection import GridSearchCV

# creating a dictionary and list of their values to optimize the model
params = {
    'n_estimators' : [100, 500, 1000],
    'criterion' : ['gini', 'entropy'],
    'max_depth' : [3, 4, 5, 6, 7, 8, 9, 10],
}
```

[37]:
```
# intiating a grid search to find the most optimum parameters
grid_search = GridSearchCV(clf, params, cv=10)
```

[38]:
```
# fitting the training data
grid_search.fit(x_train, y_train)
```

[38]:
```
GridSearchCV(cv=10, estimator=RandomForestClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 4, 5, 6, 7, 8, 9, 10],
                         'n_estimators': [100, 500, 1000]})
```

[39]:
```
# obtaining the best model
clf = grid_search.best_estimator_
```

[43]:
```
# Make predictions on the test set
y_pred = clf.predict(x_test)
# obtaining the classification report
from sklearn.metrics import classification_report
```

[44]:
```
# Generate a classification report
report = classification_report(y_test, y_pred)
# Print the classification report
print(report)
```

```
              precision    recall  f1-score   support
```

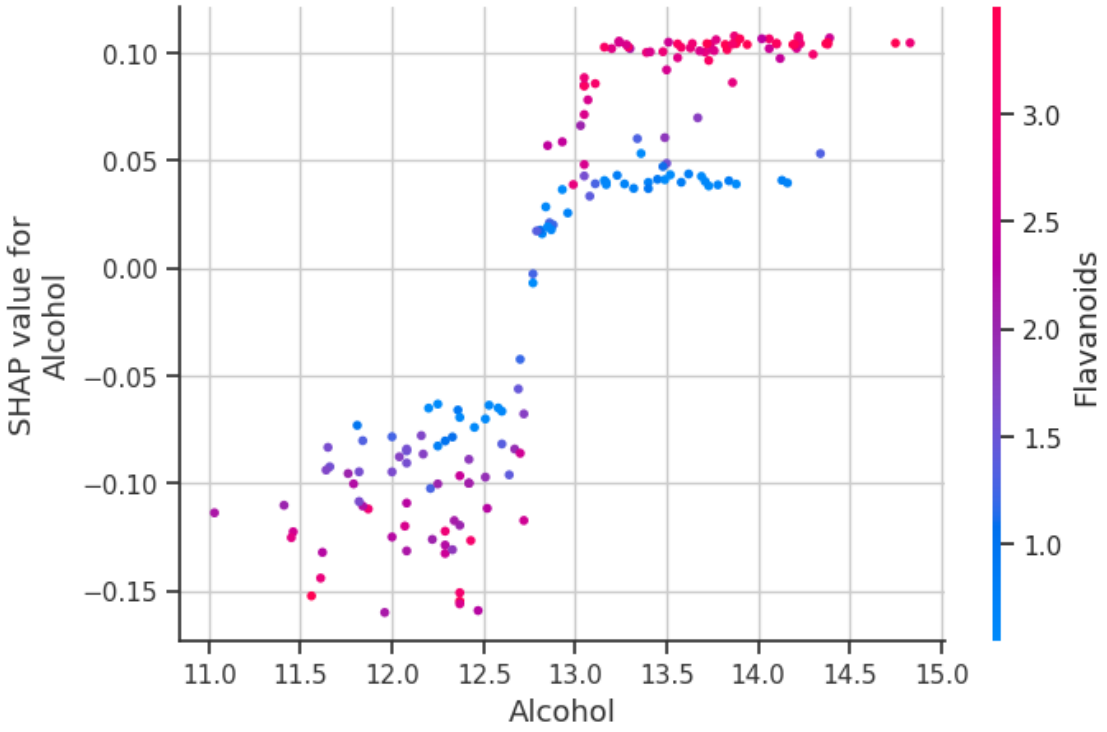
```
           1       0.95      1.00      0.97        19
           2       1.00      0.95      0.98        21
           3       1.00      1.00      1.00        14

    accuracy                           0.98        54
   macro avg       0.98      0.98      0.98        54
weighted avg       0.98      0.98      0.98        54
```

2 Explainability via SHAP

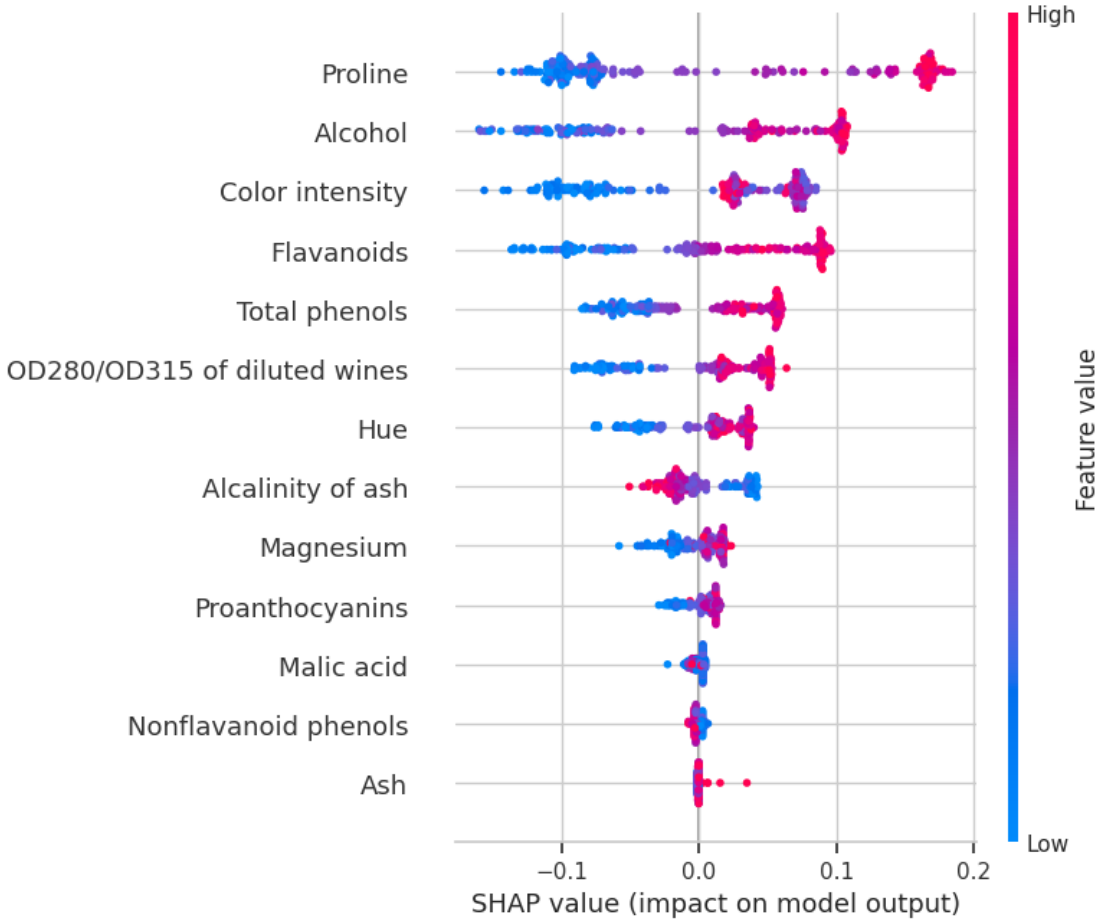
[46]:
```
# importing shap
import shap
# instantiating a TreeExplainer object
explainer = shap.TreeExplainer(clf)
```

[47]:
```
# obtaining shapely values of the data
shap_values = explainer.shap_values(x)
```

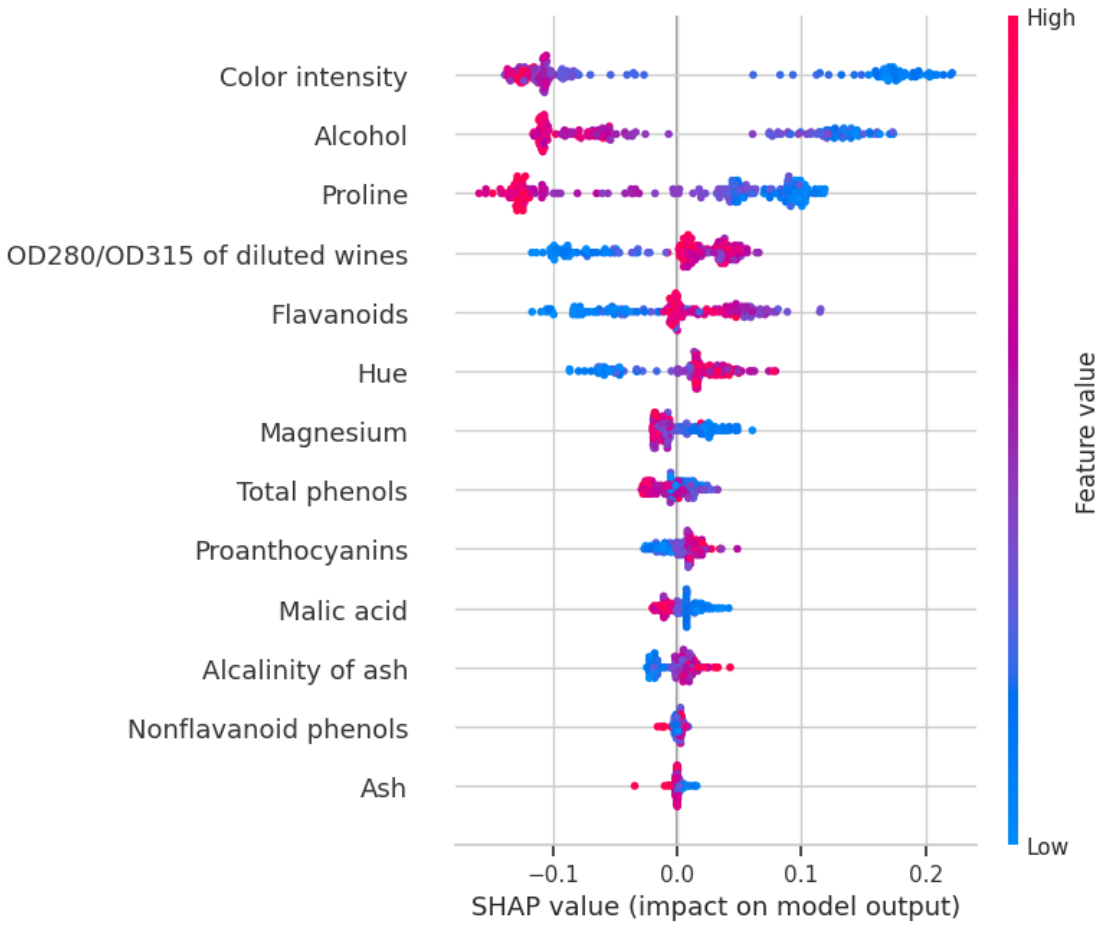
[56]:
```
# plotting the dependence of shapely values on alcohol
shap.dependence_plot('Alcohol', shap_values[0], x)
```



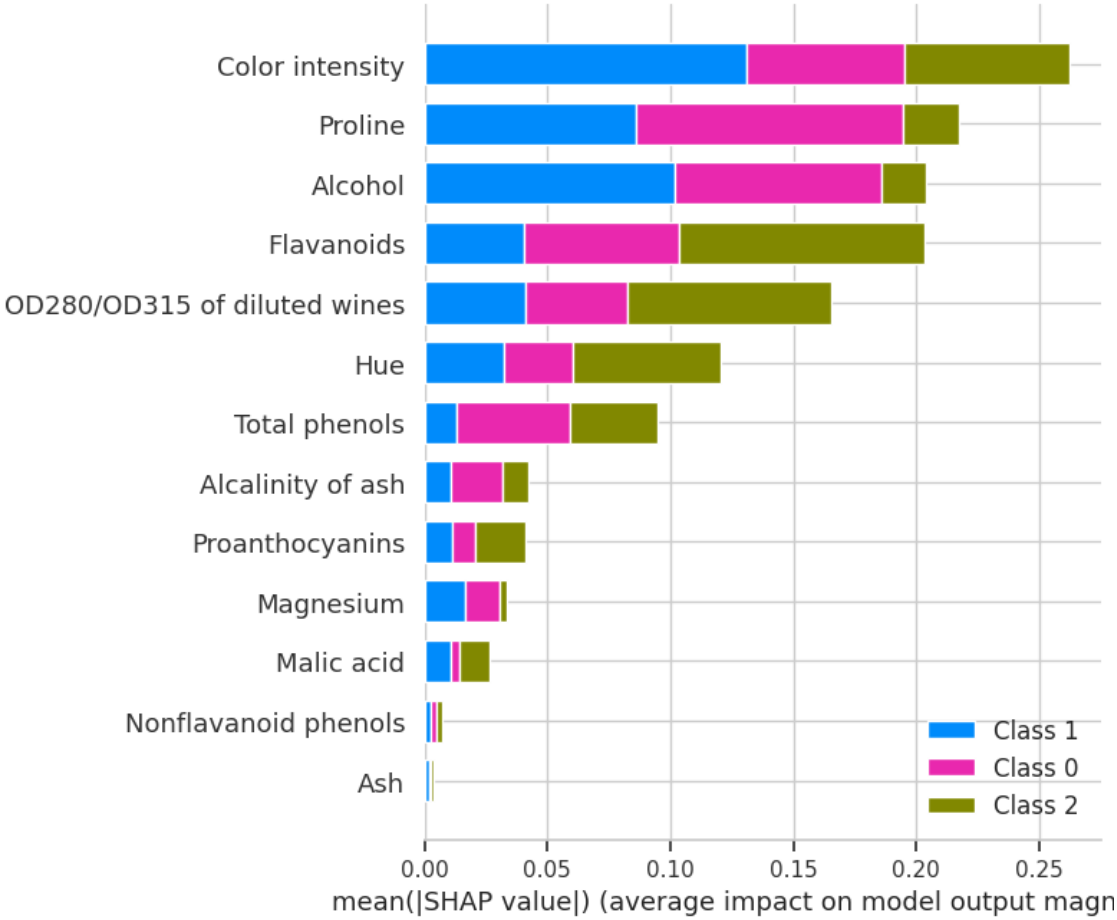
[53]:
```
# plotting the dependence of shapely values on all the features
shap.summary_plot(shap_values[0], x)
```



[58]:
```
shap.summary_plot(shap_values[1], x)
```



[59]:
```
shap.summary_plot(shap_values, x_train, plot_type="bar")
```


Color intensity
Proline
Alcohol
Flavanoids
OD280/OD315 of diluted wines
Hue
Total phenols
Alcalinity of ash
Proanthocyanins
Magnesium
Malic acid
Nonflavanoid phenols
Ash
Class 1
Class 0
Class 2
0.00
0.05
0.10
0.15
0.20
0.25
mean(|SHAP value|) (average impact on model output magn