## For Linear Regression Analysis:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
wine_columns = [
    'Class', 'Alcohol', 'Malic Acid', 'Ash', 'Alcalinity of Ash', 'Magnesium',
    'Total Phenols', 'Flavanoids', 'Nonflavanoid Phenols', 'Proanthocyanins',
    'Color Intensity', 'Hue', 'OD280/OD315 of Diluted Wines', 'Proline'
]

data_file_path = '/content/wine.data'
wine_df = pd.read_csv(data_file_path, header=None, names=wine_columns)
```

```
print(wine_data.head())
```

```
   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
```

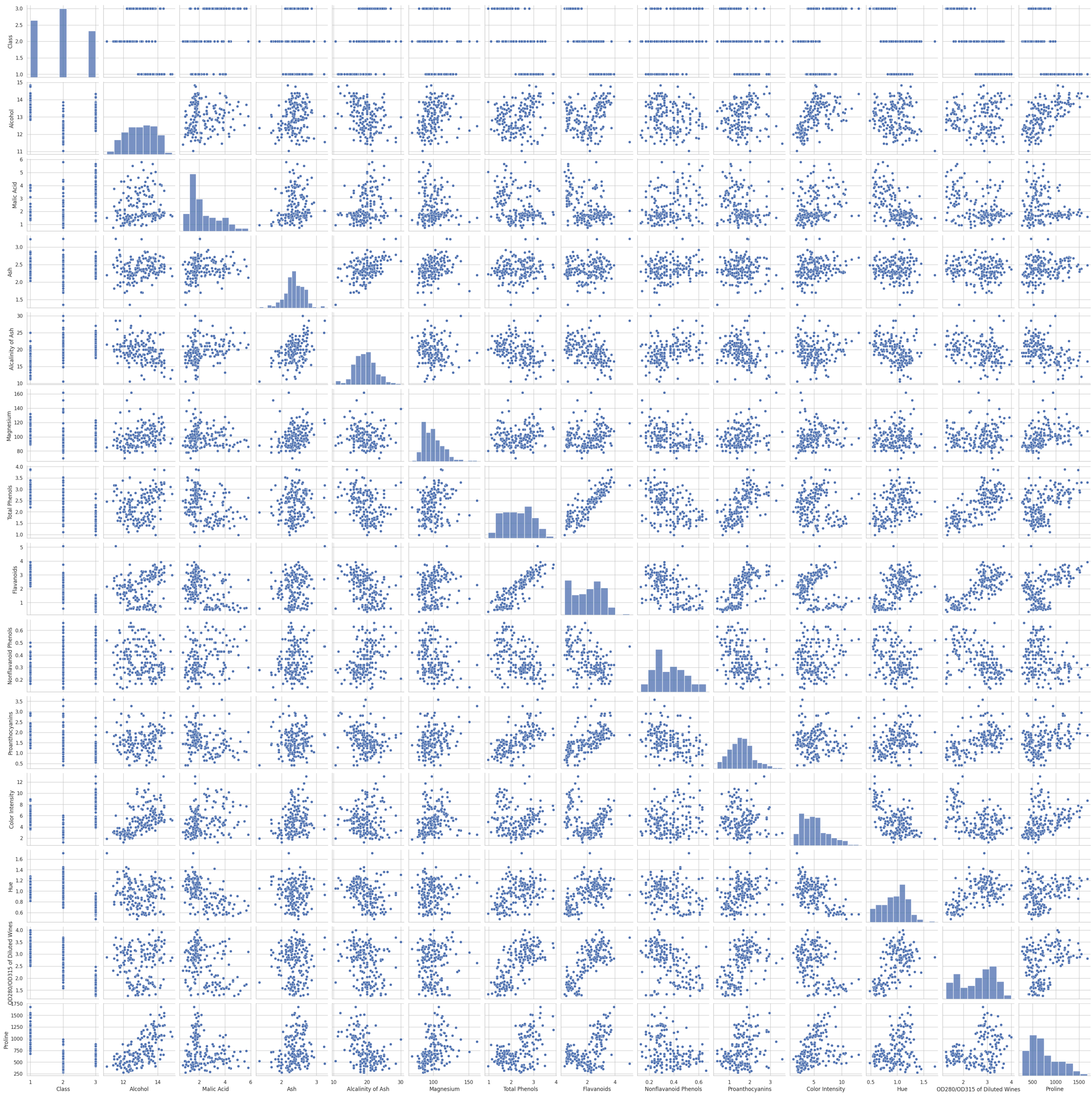
```
print(wine_df.info())
```

```
<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
None
```

```
print(wine_df.describe())
```

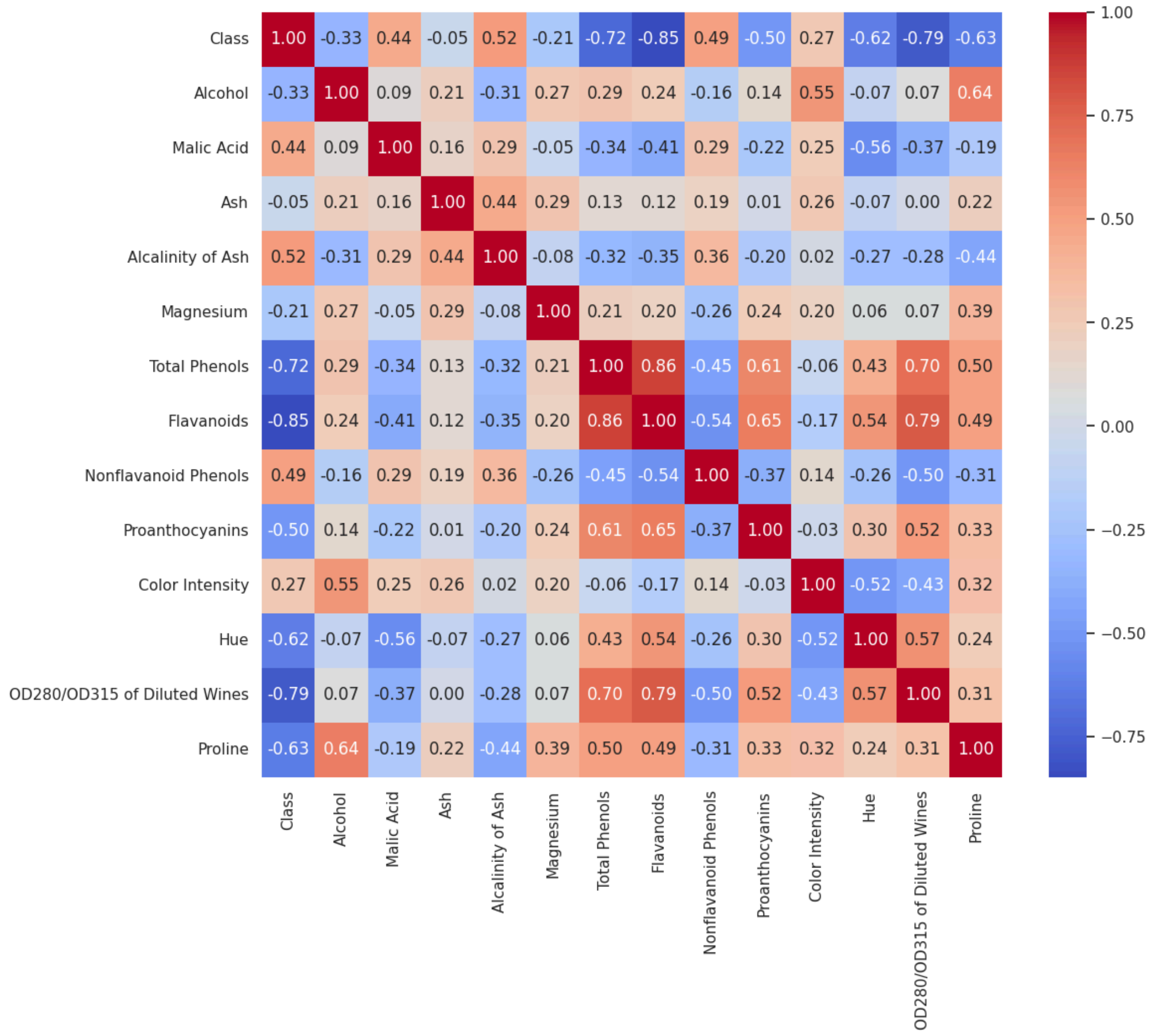
```
            Class     Alcohol  Malic Acid         Ash  Alcalinity of Ash  \
count  178.000000  178.000000  178.000000  178.000000         178.000000
mean     1.938202   13.000618    2.336348    2.366517          19.494944
std      0.775035    0.811827    1.117146    0.274344           3.339564
min      1.000000   11.030000    0.740000    1.360000          10.600000
25%      1.000000   12.362500    1.602500    2.210000          17.200000
50%      2.000000   13.050000    1.865000    2.360000          19.500000
75%      3.000000   13.677500    3.082500    2.557500          21.500000
max      3.000000   14.830000    5.800000    3.230000          30.000000

        Magnesium  Total Phenols  Flavanoids  Nonflavanoid Phenols  \
count  178.000000     178.000000  178.000000            178.000000
mean    99.741573       2.295112    2.029270              0.361854
std     14.282484       0.625851    0.998859              0.124453
min     70.000000       0.980000    0.340000              0.130000
25%     88.000000       1.742500    1.205000              0.270000
50%     98.000000       2.355000    2.135000              0.340000
75%    107.000000       2.800000    2.875000              0.437500
max    162.000000       3.880000    5.080000              0.660000

       Proanthocyanins  Color Intensity         Hue  \
count       178.000000       178.000000  178.000000
mean          1.590899         5.058090    0.957449
std           0.572359         2.318286    0.228572
min           0.410000         1.280000    0.480000
25%           1.250000         3.220000    0.782500
50%           1.555000         4.690000    0.965000
75%           1.950000         6.200000    1.120000
max           3.580000        13.000000    1.710000

       OD280/OD315 of Diluted Wines      Proline
count                    178.000000   178.000000
mean                       2.611685   746.893258
std                        0.709990   314.907474
min                        1.270000   278.000000
25%                        1.937500   500.500000
50%                        2.780000   673.500000
75%                        3.170000   985.000000
max                        4.000000  1680.000000
```

```
sns.pairplot(wine_df)
plt.show()
```




```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Prepare the data
X = wine_df_scaled.drop(['Alcohol', 'Class'], axis=1)
y = wine_df_scaled['Alcohol']

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Linear Regression Model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict and evaluate the model
predictions = model.predict(X_test)
print('Mean Squared Error:', mean_squared_error(y_test, predictions))
print('R^2 Score:', r2_score(y_test, predictions))
```

```
plt.figure(figsize=(12, 10))
sns.heatmap(wine_df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.show()
```



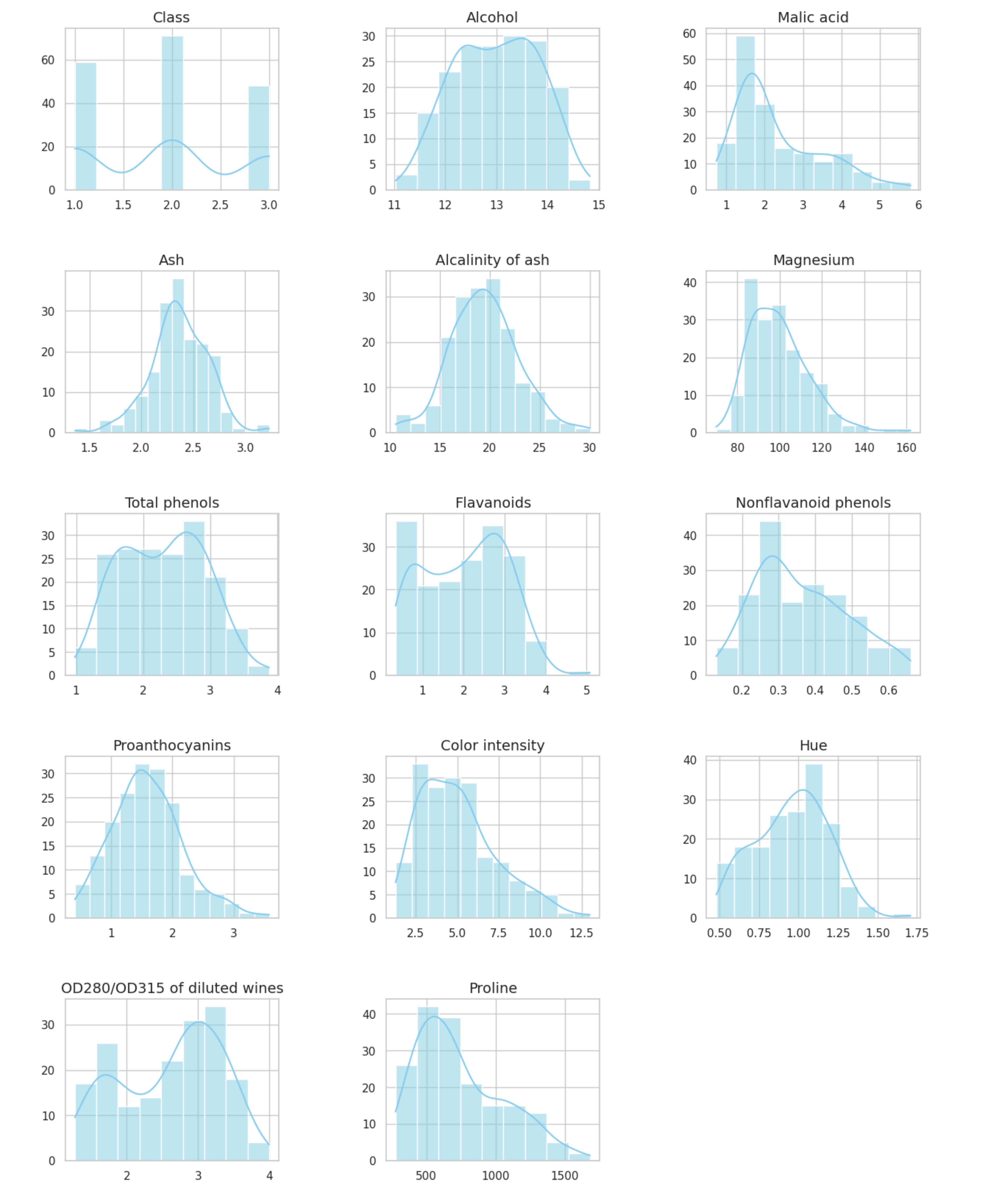

```
missing_values = wine_data.isnull().sum()
print("Missing values in each column:\n", missing_values)
```

```
Missing values in each column:
 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
```

```
summary_statistics = wine_data.describe()
print("\nSummary Statistics:\n", summary_statistics)
```

```
Summary Statistics:
             Class     Alcohol  Malic acid         Ash  Alcalinity of ash  \
count  178.000000  178.000000  178.000000  178.000000         178.000000   
mean     1.938202   13.000618    2.336348    2.366517          19.494944   
std      0.775035    0.811827    1.117146    0.274344           3.339564   
min      1.000000   11.030000    0.740000    1.360000          10.600000   
25%      1.000000   12.362500    1.602500    2.210000          17.200000   
50%      2.000000   13.050000    1.865000    2.360000          19.500000   
75%      3.000000   13.677500    3.082500    2.557500          21.500000   
max      3.000000   14.830000    5.800000    3.230000          30.000000   

        Magnesium  Total phenols  Flavanoids  Nonflavanoid phenols  \
count  178.000000     178.000000  178.000000            178.000000   
mean    99.741573       2.295112    2.029270              0.361854   
std     14.282484       0.625851    0.998859              0.124453   
min     70.000000       0.980000    0.340000              0.130000   
25%     88.000000       1.742500    1.205000              0.270000   
50%     98.000000       2.355000    2.135000              0.340000   
75%    107.000000       2.800000    2.875000              0.437500   
max    162.000000       3.880000    5.080000              0.660000   

       Proanthocyanins  Color intensity         Hue  \
count       178.000000       178.000000  178.000000   
mean          1.590899         5.058090    0.957449   
std           0.572359         2.318286    0.228572   
min           0.410000         1.280000    0.480000   
25%           1.250000         3.220000    0.782500   
50%           1.555000         4.690000    0.965000   
75%           1.950000         6.200000    1.120000   
max           3.580000        13.000000    1.710000   

       OD280/OD315 of diluted wines      Proline  
count                    178.000000   178.000000  
mean                       2.611685   746.893258  
std                        0.709990   314.907474  
min                        1.270000   278.000000  
25%                        1.937500   500.500000  
50%                        2.780000   673.500000  
75%                        3.170000   985.000000  
max                        4.000000  1680.000000  
```

```
sns.set(style="whitegrid")
fig, axes = plt.subplots(nrows=5, ncols=3, figsize=(15, 20))
fig.subplots_adjust(hspace=0.5, wspace=0.5)
axes = axes.flatten()


for i, col in enumerate(wine_data.columns):
    sns.histplot(wine_data[col], kde=True, ax=axes[i], color='skyblue')
    axes[i].set_title(col, fontsize=14)
    axes[i].set_xlabel('')
    axes[i].set_ylabel('')

# Remove any empty plots
for ax in axes[len(wine_data.columns):]:
    fig.delaxes(ax)

plt.show()
```




```
# Check for missing values (if any)
print(wine_df.isnull().sum())

# Feature Scaling - Standardization (because features have different ranges)
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
wine_df_scaled = pd.DataFrame(scaler.fit_transform(wine_df.drop(['Class'], axis=1)), columns=wine_df.columns[1:])
wine_df_scaled['Class'] = wine_df['Class']  # Add the non-scaled 'Class' column back

print(wine_df_scaled.head())
```

```
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
    Alcohol  Malic Acid       Ash  Alcalinity of Ash  Magnesium  \
0  1.518613   -0.562250  0.232053          -1.169593   1.913905
1  0.246290   -0.499413 -0.827996          -2.490847   0.018145
2  0.196879    0.021231  1.109334          -0.268738   0.088358
3  1.691550   -0.346811  0.487926          -0.809251   0.930918
4  0.295700    0.227694  1.840403           0.451946   1.281985

   Total Phenols  Flavanoids  Nonflavanoid Phenols  Proanthocyanins  \
0       0.808997    1.034819             -0.659563         1.224884
1       0.568648    0.733629             -0.820719        -0.544721
2       0.808997    1.215533             -0.498407         2.135968
3       2.491446    1.466525             -0.981875         1.032155
4       0.808997    0.663351              0.226796         0.401404

   Color Intensity       Hue  OD280/OD315 of Diluted Wines   Proline  Class
0         0.251717  0.362177                      1.847920  1.013009      1
1        -0.293321  0.406051                      1.113449  0.965242      1
2         0.269020  0.318304                      0.788587  1.395148      1
3         1.186068 -0.427544                      1.184071  2.334574      1
4        -0.319276  0.362177                      0.449601 -0.037874      1
```

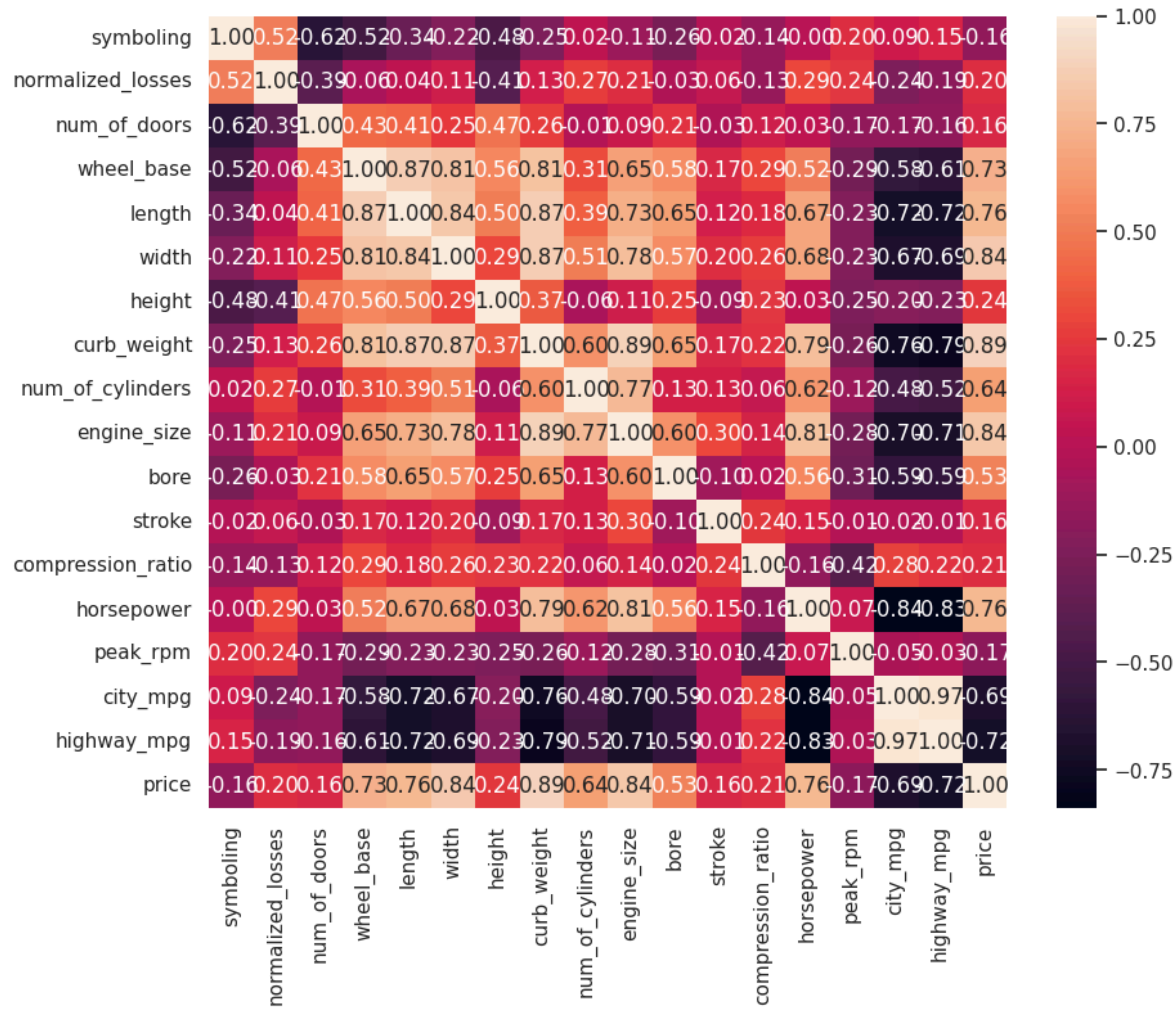
## For Logistic Regression Analysis

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

column_names = [
    "symboling", "normalized_losses", "make", "fuel_type", "aspiration",
    "num_of_doors", "body_style", "drive_wheels", "engine_location",
    "wheel_base", "length", "width", "height", "curb_weight", "engine_type",
    "num_of_cylinders", "engine_size", "fuel_system", "bore", "stroke",
    "compression_ratio", "horsepower", "peak_rpm", "city_mpg", "highway_mpg", "price"
]
data = pd.read_csv('/content/imports-85.data', names=column_names)

print(data.head())
```

```
   symboling normalized_losses         make fuel_type aspiration num_of_doors  \
0          3                 ?  alfa-romero       gas        std          two
1          3                 ?  alfa-romero       gas        std          two
2          1                 ?  alfa-romero       gas        std          two
3          2               164         audi       gas        std         four
4          2               164         audi       gas        std         four

    body_style drive_wheels engine_location  wheel_base  ...  engine_size  \
0  convertible          rwd           front        88.6  ...          130
1  convertible          rwd           front        88.6  ...          130
2    hatchback          rwd           front        94.5  ...          152
3        sedan          fwd           front        99.8  ...          109
4        sedan          4wd           front        99.4  ...          136

   fuel_system  bore  stroke compression_ratio horsepower  peak_rpm city_mpg  \
0         mpfi  3.47    2.68               9.0        111      5000       21
1         mpfi  3.47    2.68               9.0        111      5000       21
2         mpfi  2.68    3.47               9.0        154      5000       19
3         mpfi  3.19    3.40              10.0        102      5500       24
4         mpfi  3.19    3.40               8.0        115      5500       18

  highway_mpg  price
0          27  13495
1          27  16500
2          26  16500
3          30  13950
4          22  17450

[5 rows x 26 columns]
```

```
data.replace('?', np.nan, inplace=True)
data['normalized_losses'] = pd.to_numeric(data['normalized_losses'], errors='coerce')
data['price'] = pd.to_numeric(data['price'], errors='coerce')
numeric_columns = ['bore', 'stroke', 'horsepower', 'peak_rpm']
data[numeric_columns] = data[numeric_columns].apply(pd.to_numeric, errors='coerce')

data.dropna(subset=['price'], inplace=True)
for column in numeric_columns:
    data[column].fillna(data[column].mean(), inplace=True)

categorical_columns = ['make', 'fuel_type', 'aspiration', 'body_style', 'drive_wheels', 'engine_location']
data = pd.get_dummies(data, columns=categorical_columns)
```

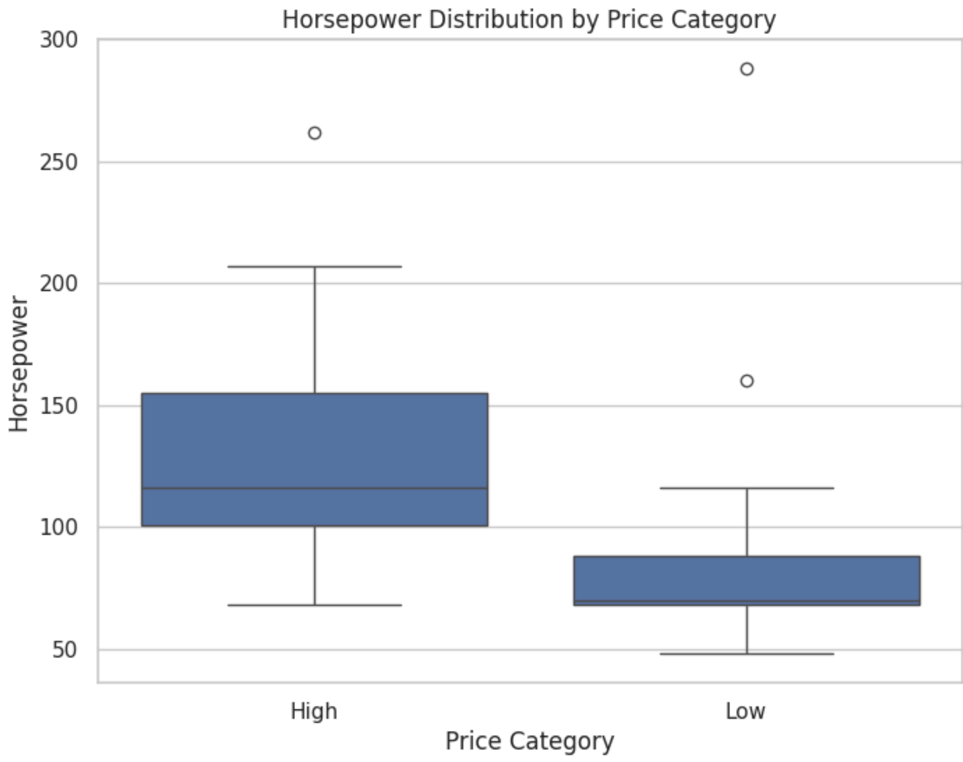
```
data['normalized_losses'] = pd.to_numeric(data['normalized_losses'], errors='coerce')
data['bore'] = pd.to_numeric(data['bore'], errors='coerce')
data['stroke'] = pd.to_numeric(data['stroke'], errors='coerce')
data['horsepower'] = pd.to_numeric(data['horsepower'], errors='coerce')
data['peak_rpm'] = pd.to_numeric(data['peak_rpm'], errors='coerce')
data['price'] = pd.to_numeric(data['price'], errors='coerce')

numeric_data = data.select_dtypes(include=[np.number])

numeric_data.dropna(inplace=True)

plt.figure(figsize=(10, 8))
sns.heatmap(numeric_data.corr(), annot=True, fmt=".2f")
plt.show()
```



```
selected_columns = ['horsepower', 'city_mpg', 'highway_mpg', 'price', 'high_price']
sub_data = data[selected_columns]

sub_data['high_price'] = sub_data['high_price'].map({0: 'Low', 1: 'High'})

sns.pairplot(sub_data, hue='high_price', diag_kind='kde', markers=["o", "s"],
             plot_kws={'alpha': 0.6, 's': 80, 'edgecolor': 'k'},
             height=3)

plt.show()
```

```
<ipython-input-88-3470c4cf172e>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
  sub_data['high_price'] = sub_data['high_price'].map({0: 'Low', 1: 'High'})
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import pandas as pd

data['high_price'] = data['high_price'].map({0: 'Low', 1: 'High'})

plt.figure(figsize=(8, 6))
sns.boxplot(x='high_price', y='horsepower', data=data)

plt.title('Horsepower Distribution by Price Category')
plt.xlabel('Price Category')
plt.ylabel('Horsepower')

plt.show()
```




```
# Prepare data
X = data.drop(['price', 'high_price'], axis=1)
y = data['high_price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a pipeline with imputation, scaling, and logistic regression
pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler()),  # This will scale each feature to mean 0 and standard deviation 1
    ('logistic_regression', LogisticRegression(max_iter=10000))  # Increased max_iter
])
```

```
# Train the model
pipeline.fit(X_train, y_train)
```

```
▸ Pipeline
  ▸ SimpleImputer
```