Seatwork 11.1 Exploratory Data Analysis for Machine Learning

Name: John Rome A. Belocora
Section: CPE22S3
Date: 04/26/2024
Teacher: Engr. Roman Richard

```
pip install ucimlrepo
```

```
Collecting ucimlrepo
  Downloading ucimlrepo-0.0.6-py3-none-any.whl (8.0 kB)
Installing collected packages: ucimlrepo
Successfully installed ucimlrepo-0.0.6
```

```
!pip install hvplot
```

```
Collecting hvplot
  Downloading hvplot-0.9.2-py2.py3-none-any.whl (1.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 12.1 MB/s eta 0:00:00
Requirement already satisfied: bokeh>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from hvplot) (3.3.4)
Requirement already satisfied: colorcet>=2 in /usr/local/lib/python3.10/dist-packages (from hvplot) (3.1.0)
Requirement already satisfied: holoviews>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from hvplot) (1.17.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from hvplot) (2.0.3)
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.10/dist-packages (from hvplot) (1.25.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from hvplot) (24.0)
Requirement already satisfied: panel>=0.11.0 in /usr/local/lib/python3.10/dist-packages (from hvplot) (1.3.8)
Requirement already satisfied: param<3.0,>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from hvplot) (2.1.0)
Requirement already satisfied: Jinja2>=2.9 in /usr/local/lib/python3.10/dist-packages (from bokeh>=1.0.0->hvplot) (3.1.3)
Requirement already satisfied: contourpy>=1 in /usr/local/lib/python3.10/dist-packages (from bokeh>=1.0.0->hvplot) (1.2.1)
Requirement already satisfied: pillow>=7.1.0 in /usr/local/lib/python3.10/dist-packages (from bokeh>=1.0.0->hvplot) (9.4.0)
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.10/dist-packages (from bokeh>=1.0.0->hvplot) (6.0.1)
Requirement already satisfied: tornado>=5.1 in /usr/local/lib/python3.10/dist-packages (from bokeh>=1.0.0->hvplot) (6.3.3)
Requirement already satisfied: xyzservices>=2021.09.1 in /usr/local/lib/python3.10/dist-packages (from bokeh>=1.0.0->hvplot) (2024.4.0)
Requirement already satisfied: pyviz-comms>=0.7.4 in /usr/local/lib/python3.10/dist-packages (from holoviews>=1.11.0->hvplot) (3.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas->hvplot) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->hvplot) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas->hvplot) (2024.1)
Requirement already satisfied: markdown in /usr/local/lib/python3.10/dist-packages (from panel>=0.11.0->hvplot) (3.6)
Requirement already satisfied: markdown-it-py in /usr/local/lib/python3.10/dist-packages (from panel>=0.11.0->hvplot) (3.0.0)
Requirement already satisfied: linkify-it-py in /usr/local/lib/python3.10/dist-packages (from panel>=0.11.0->hvplot) (2.0.3)
Requirement already satisfied: mdit-py-plugins in /usr/local/lib/python3.10/dist-packages (from panel>=0.11.0->hvplot) (0.4.0)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from panel>=0.11.0->hvplot) (2.31.0)
Requirement already satisfied: tqdm>=4.48.0 in /usr/local/lib/python3.10/dist-packages (from panel>=0.11.0->hvplot) (4.66.2)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from panel>=0.11.0->hvplot) (6.1.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from panel>=0.11.0->hvplot) (4.11.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from Jinja2>=2.9->bokeh>=1.0.0->hvplot) (2.1.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas->hvplot) (1.16.0
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->panel>=0.11.0->hvplot) (0.5.1)
Requirement already satisfied: uc-micro-py in /usr/local/lib/python3.10/dist-packages (from linkify-it-py->panel>=0.11.0->hvplot) (1.0.3
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py->panel>=0.11.0->hvplot) (0.1.2
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->panel>=0.11.0->hvplot
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->panel>=0.11.0->hvplot) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->panel>=0.11.0->hvplot) (2.0
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->panel>=0.11.0->hvplot) (202
Installing collected packages: hvplot
Successfully installed hvplot-0.9.2
```

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
automobile = fetch_ucirepo(id=10)

# data (as pandas dataframes)
A = automobile.data.features
B = automobile.data.targets
```

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

from sklearn.model_selection import train_test_split

from sklearn import metrics

from sklearn.linear_model import LinearRegression

%matplotlib inline
```

Linear Regression Analysis

Data Wrangling

```
#Concatenating
amdata = pd.concat([A, B])
amdata
```

	price	highway-mpg	city-mpg	peak-rpm	horsepower	compression-ratio	stroke	bore	fuel-system	en
0	13495.0	27.0	21.0	5000.0	111.0	9.0	2.68	3.47	mpfi	
1	16500.0	27.0	21.0	5000.0	111.0	9.0	2.68	3.47	mpfi	
2	16500.0	26.0	19.0	5000.0	154.0	9.0	3.47	2.68	mpfi	
3	13950.0	30.0	24.0	5500.0	102.0	10.0	3.40	3.19	mpfi	
4	17450.0	22.0	18.0	5500.0	115.0	8.0	3.40	3.19	mpfi	
...	...	...	...	...	...	...	...	...	...	
200	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
201	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
202	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
203	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
204	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

```
#Identifying missing values in each column
amdata.isnull().sum()
```

```
price                209
highway-mpg          205
city-mpg             205
peak-rpm             207
horsepower           207
compression-ratio    205
stroke               209
bore                 209
fuel-system          205
engine-size          205
num-of-cylinders     205
engine-type          205
curb-weight          205
height               205
width                205
length               205
wheel-base           205
engine-location      205
drive-wheels         205
body-style           205
```

```
num-of-doors         207
aspiration           205
fuel-type            205
make                 205
normalized-losses    246
symboling            205
dtype: int64
```

```
#Identifying datatypes in each column
amdata.dtypes
```

```
price                float64
highway-mpg          float64
city-mpg             float64
peak-rpm             float64
horsepower           float64
compression-ratio    float64
stroke               float64
bore                 float64
fuel-system           object
engine-size          float64
num-of-cylinders     float64
engine-type           object
curb-weight          float64
height               float64
width                float64
length               float64
wheel-base           float64
engine-location       object
drive-wheels          object
body-style            object
num-of-doors         float64
aspiration            object
fuel-type             object
make                  object
normalized-losses    float64
symboling            float64
dtype: object
```

```
# We can notice here that the 209 values in the column price has missing values
amdata.price.isnull().value_counts()
```

```
price
True     209
False    201
Name: count, dtype: int64
```

To do Linear Regression we have to remove object data types in order to predict the value of a variable based on the value of another variable

```
#Creating new dataframe
new_df = amdata.copy()
new_df
```

	price	highway-mpg	city-mpg	peak-rpm	horsepower	compression-ratio	stroke	bore	fuel-system	en
0	13495.0	27.0	21.0	5000.0	111.0	9.0	2.68	3.47	mpfi	
1	16500.0	27.0	21.0	5000.0	111.0	9.0	2.68	3.47	mpfi	
2	16500.0	26.0	19.0	5000.0	154.0	9.0	3.47	2.68	mpfi	
3	13950.0	30.0	24.0	5500.0	102.0	10.0	3.40	3.19	mpfi	
4	17450.0	22.0	18.0	5500.0	115.0	8.0	3.40	3.19	mpfi	
...	...	...	...	...	...	...	...	...	...	
200	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
201	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
202	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
203	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
204	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

```
#Dropping object datatypes and fill missing values with mean
def Doafmv(data):
    for column in data.columns:
        #Identifying the column if it is a object datatype
        if data[column].dtype == 'object':
            #If the column is a object, the column will get dropped
            data = data.drop(column, axis=1)
        else:
            #If the column is not a object datatype, it will get the mean
            column_mean = data[column].mean()
            #Filling missing values with the collected mean value
            data[column] = data[column].fillna(column_mean)
    return data
```

```
#Output of the new datatypes result
new_df = Doafmv(new_df)
new_df.dtypes
```

```
price                float64
highway-mpg          float64
city-mpg             float64
peak-rpm             float64
horsepower           float64
compression-ratio    float64
stroke               float64
bore                 float64
engine-size          float64
num-of-cylinders     float64
curb-weight          float64
height               float64
width                float64
length               float64
wheel-base           float64
num-of-doors         float64
normalized-losses    float64
symboling            float64
dtype: object
```

```
#Output of the new results of number of missing values
new_df.isnull().sum()
```

```
price                0
highway-mpg          0
city-mpg             0
peak-rpm             0
horsepower           0
compression-ratio    0
stroke               0
bore                 0
engine-size          0
num-of-cylinders     0
```

```
curb-weight          0
height               0
width                0
length               0
wheel-base           0
num-of-doors         0
normalized-losses    0
symboling            0
dtype: int64
```

Correlation

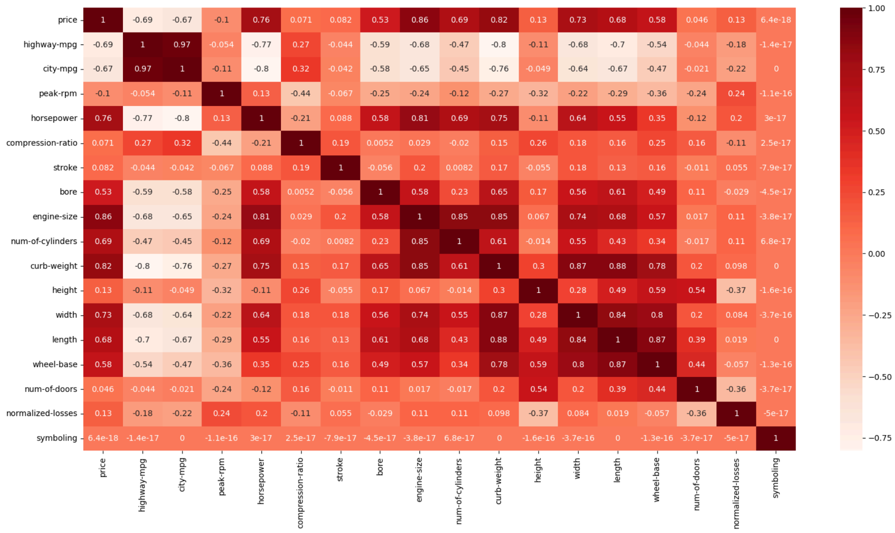
```
# Dataframe correlation
new_df.corr()
```

	price	highway-mpg	city-mpg	peak-rpm	horsepower	compressi ra
price	1.000000e+00	-6.905257e-01	-0.667449	-1.008541e-01	7.579170e-01	7.099036e
highway-mpg	-6.905257e-01	1.000000e+00	0.971337	-5.425672e-02	-7.709027e-01	2.652014e
city-mpg	-6.674493e-01	9.713370e-01	1.000000	-1.137229e-01	-8.031621e-01	3.247014e
peak-rpm	-1.008541e-01	-5.425672e-02	-0.113723	1.000000e+00	1.309708e-01	-4.359359e
horsepower	7.579170e-01	-7.709027e-01	-0.803162	1.309708e-01	1.000000e+00	-2.057397e
compression-ratio	7.099036e-02	2.652014e-01	0.324701	-4.359359e-01	-2.057397e-01	1.000000e
stroke	8.209530e-02	-4.396069e-02	-0.042179	-6.684439e-02	8.826363e-02	1.861052e
bore	5.323000e-01	-5.869915e-01	-0.584508	-2.547613e-01	5.757374e-01	5.200705e
engine-size	8.617522e-01	-6.774699e-01	-0.653658	-2.445994e-01	8.107125e-01	2.897136e
num-of-cylinders	6.877698e-01	-4.666659e-01	-0.445837	-1.243575e-01	6.912082e-01	-2.000185e
curb-weight	8.208247e-01	-7.974648e-01	-0.757414	-2.662830e-01	7.509684e-01	1.513617e
height	1.343875e-01	-1.073576e-01	-0.048640	-3.206018e-01	-1.101370e-01	2.612142e
width	7.286988e-01	-6.772179e-01	-0.642704	-2.198592e-01	6.421954e-01	1.811286e
length	6.829863e-01	-7.046616e-01	-0.670909	-2.870306e-01	5.544341e-01	1.584137e
wheel-base	5.831681e-01	-5.440819e-01	-0.470414	-3.607037e-01	3.519573e-01	2.497858e
num-of-doors	4.600081e-02	-4.421287e-02	-0.020671	-2.402947e-01	-1.240007e-01	1.615024e
normalized-losses	1.339987e-01	-1.782209e-01	-0.218749	2.377476e-01	2.034339e-01	-1.145246e
symboling	6.412808e-18	-1.431166e-17	0.000000	-1.057927e-16	2.992664e-17	2.481255e

Heatmap Correlation

```
# Heatmap correlation of the Dataframe
plt.figure(figsize=[20,10])
sns.heatmap(new_df.corr(), annot=True,cmap='Reds')
```

```
<Axes: >
```



Train a Linear Regression Model

```
#Using the highway-mpg and city-mpg as a Train test
highway = new_df['highway-mpg']
city = new_df['city-mpg']
```

```
highway.shape
```

```
(410,)
```

Train Test Split

```
#Train Test Split
highway_train, highway_test, city_train, city_test = train_test_split(highway, city, test_size=0.3, random_state = 100)
```

```
highway_train.shape
```

```
(287,)
```

```
highway_test.shape
```

```
(123,)
```

```
city_train.shape
```

```
(287,)
```

```
city_test.shape
```

```
(123,)
```

Linear Regression

```
model = LinearRegression()
```

```
#Reshaping the 1D array into a 2D array for model.fit standards
city_train_reshaped = np.reshape(city_train, (-1, 1))
highway_train_reshaped = np.reshape(highway_train, (-1, 1))

city_test_reshaped = np.reshape(city_test, (-1, 1))
highway_test_reshaped = np.reshape(highway_test, (-1, 1))
```

```
model.fit(highway_train_reshaped, city_train_reshaped)
```

```
▾ LinearRegression
LinearRegression()
```

Model Coefficient

```
model.coef_
```

```
array([[0.90745472]])
```

Predictions from our Model

```
highway_pred = model.predict(highway_test_reshaped)
highway_pred
```

```
       [36.32104129],
       [25.20582758],
       [20.89431098],
       [22.70922043],
       [19.98685626],
       [24.52412988],
       [25.20582758],
       [25.20582758],
       [25.20582758],
       [25.20582758],
       [25.20582758],
       [25.20582758],
       [25.20582758],
       [34.50613185],
       [42.67322436],
       [19.98685626],
       [25.20582758],
       [25.20582758],
       [22.70922043],
       [25.20582758],
       [31.78376767],
       [24.52412988],
       [36.32104129],
       [36.32104129],
       [25.20582758],
       [17.26449209],
       [25.20582758],
       [26.33903933],
       [25.20582758],
       [19.07940154],
       [25.20582758]])
```

```
city_pred = model.predict(city_test_reshaped)
city_pred
```

```
       [31.78376767],
       [31.78376767],
       [20.18605364],
       [11.81976374],
       [20.18605364],
       [20.89431098],
       [20.18605364],
       [14.54212792],
       [20.18605364]])
```

Regression Evaluation Metrics

```
MAE = metrics.mean_absolute_error(highway_test_reshaped, highway_pred)
MSE = metrics.mean_squared_error(highway_test_reshaped, highway_pred)
RMSE= np.sqrt(MSE)
```

```
#Mean Absolute Error
MAE
```

```
5.589537666261423
```

```
#Mean Squared Error
MSE
```

```
31.479419933629654
```

```
#Root Mean Squared Error
RMSE
```

```
5.610652362571544
```

```
new_df['highway-mpg'].mean()
```

```
30.75121951219512
```

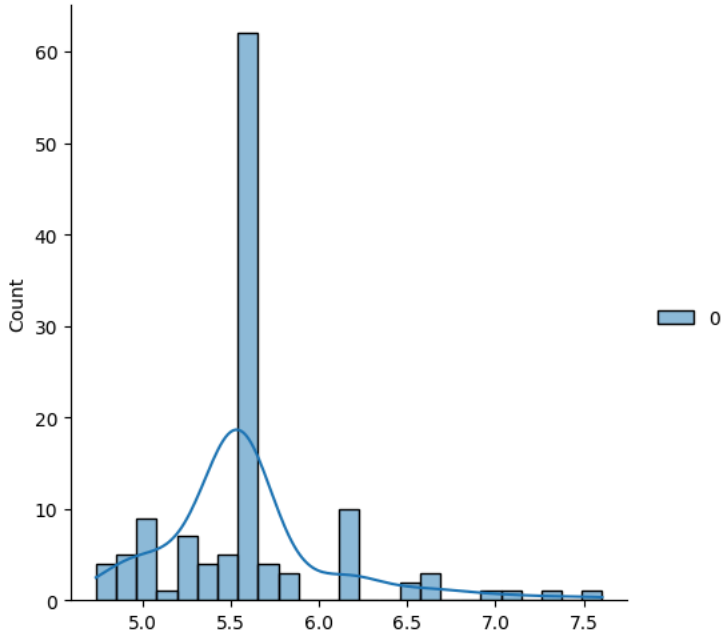
Residual Histograms

```
#Declaration of test residual variables
test_residual_highway = highway_test_reshaped - highway_pred
test_residual_city = city_test_reshaped - city_pred
```

Using Residual Histograms to check whether the variance is normally distributed

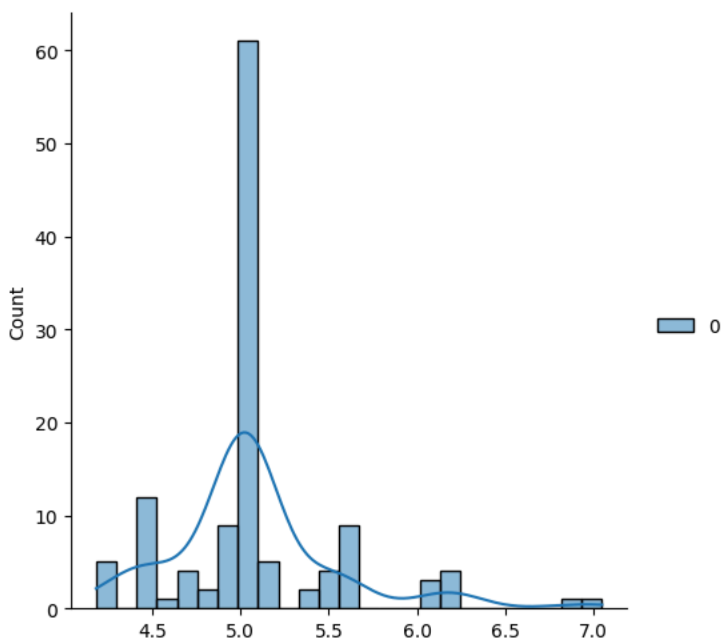
```
sns.displot(test_residual_highway, bins=25, kde=True)
```

<seaborn.axisgrid.FacetGrid at 0x7d56eef668f0>




```
sns.displot(test_residual_city, bins=25, kde=True)
```

<seaborn.axisgrid.FacetGrid at 0x7d56ef8f7610>



Logistic Regression Analysis

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

```
import warnings

warnings.filterwarnings('ignore')
```

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
wine = fetch_ucirepo(id=109)

# data (as pandas dataframes)
X = wine.data.features
y = wine.data.targets


wine_data = pd.concat([X, y])
wine_data
```

	Alcohol	Malicacid	Ash	Alcalinity_of_ash	Magnesium	Total_phenols	Flavanoids	I
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	
...	...	...	...	...	...	...	...	
173	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
174	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
175	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
176	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
177	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

356 rows × 14 columns

Next steps: View recommended plots

Exploratory Data Analysis

```
wine_data.shape
```

```
(356, 14)
```

```
wine_data.columns
```

```
Index(['Alcohol', 'Malicacid', 'Ash', 'Alcalinity_of_ash', 'Magnesium',
       'Total_phenols', 'Flavanoids', 'Nonflavanoid_phenols',
       'Proanthocyanins', 'Color_intensity', 'Hue',
       '0D280_0D315_of_diluted_wines', 'Proline', 'class'],
      dtype='object')
```

```
wine_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 356 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Alcohol                       178 non-null    float64
 1   Malicacid                     178 non-null    float64
 2   Ash                           178 non-null    float64
 3   Alcalinity_of_ash             178 non-null    float64
 4   Magnesium                     178 non-null    float64
 5   Total_phenols                 178 non-null    float64
 6   Flavanoids                    178 non-null    float64
 7   Nonflavanoid_phenols          178 non-null    float64
 8   Proanthocyanins               178 non-null    float64
 9   Color_intensity               178 non-null    float64
 10  Hue                           178 non-null    float64
 11  0D280_0D315_of_diluted_wines  178 non-null    float64
 12  Proline                       178 non-null    float64
 13  class                         178 non-null    float64
dtypes: float64(14)
memory usage: 41.7 KB
```

```
new_df2 = wine_data.copy()
new_df2
```

	Alcohol	Malicacid	Ash	Alcalinity_of_ash	Magnesium	Total_phenols	Flavanoids
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69
...	...	...	...	...	...	...	...
173	NaN	NaN	NaN	NaN	NaN	NaN	NaN
174	NaN	NaN	NaN	NaN	NaN	NaN	NaN
175	NaN	NaN	NaN	NaN	NaN	NaN	NaN
176	NaN	NaN	NaN	NaN	NaN	NaN	NaN
177	NaN	NaN	NaN	NaN	NaN	NaN	NaN

356 rows × 14 columns

Next steps: View recommended plots

```
new_df2 = Doafmv(new_df2)
new_df2
```

	Alcohol	Malicacid	Ash	Alcalinity_of_ash	Magnesium	Total_phenols	Flavan
0	14.230000	1.710000	2.430000	15.600000	127.000000	2.800000	3.0
1	13.200000	1.780000	2.140000	11.200000	100.000000	2.650000	2.7
2	13.160000	2.360000	2.670000	18.600000	101.000000	2.800000	3.2
3	14.370000	1.950000	2.500000	16.800000	113.000000	3.850000	3.4
4	13.240000	2.590000	2.870000	21.000000	118.000000	2.800000	2.6
...	...	...	...	...	...	...	
173	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.0
174	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.0
175	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.0
176	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.0
177	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.0

356 rows × 14 columns

Next steps: View recommended plots

```
new_df2.rename(columns={'Flavanoids':'Flavonoids', 'Nonflavanoid_phenols':'Nonflavonoid_phenols'}, inplace=True)
new_df2
```

	Alcohol	Malicacid	Ash	Alcalinity_of_ash	Magnesium	Total_phenols	Flavon
0	14.230000	1.710000	2.430000	15.600000	127.000000	2.800000	3.0
1	13.200000	1.780000	2.140000	11.200000	100.000000	2.650000	2.7
2	13.160000	2.360000	2.670000	18.600000	101.000000	2.800000	3.2
3	14.370000	1.950000	2.500000	16.800000	113.000000	3.850000	3.4
4	13.240000	2.590000	2.870000	21.000000	118.000000	2.800000	2.6
...	...	...	...	...	...	...	
173	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.0
174	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.0
175	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.0
176	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.0
177	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.0

356 rows × 14 columns

Next steps: View recommended plots

Identifying Outliers

Outliers in Numerical Variables

```
print(round(new_df2.describe()),2)
```

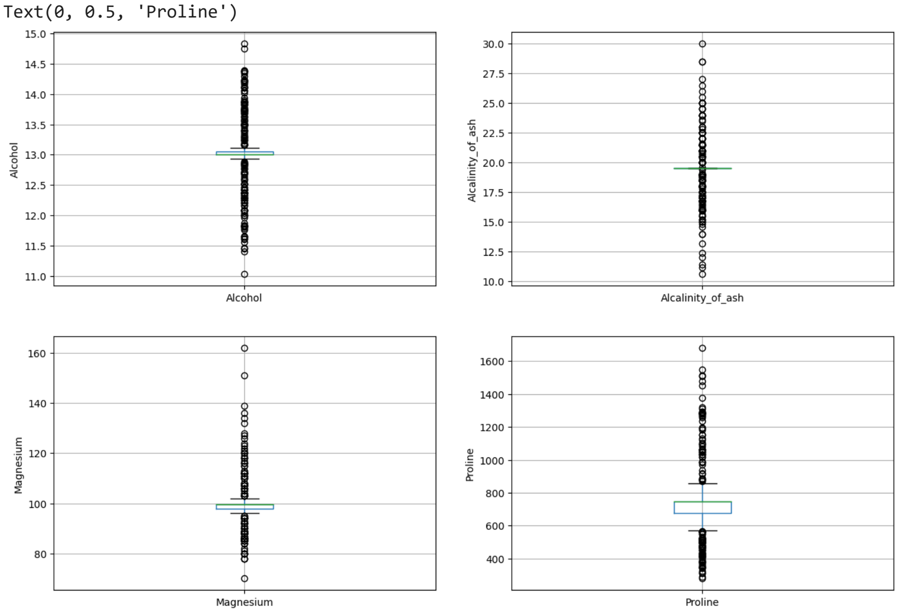
```
       Alcohol  Malicacid    Ash  Alcalinity_of_ash  Magnesium  Total_phenols  \
count    356.0      356.0  356.0              356.0      356.0          356.0
mean      13.0        2.0    2.0               19.0      100.0            2.0
std        1.0        1.0    0.0                2.0       10.0            0.0
min       11.0        1.0    1.0               11.0       70.0            1.0
25%       13.0        2.0    2.0               19.0       98.0            2.0
50%       13.0        2.0    2.0               19.0      100.0            2.0
75%       13.0        2.0    2.0               20.0      100.0            2.0
max       15.0        6.0    3.0               30.0      162.0            4.0

       Flavonoids  Nonflavonoid_phenols  Proanthocyanins  Color_intensity  \
count       356.0                 356.0            356.0            356.0
mean          2.0                   0.0              2.0              5.0
std           1.0                   0.0              0.0              2.0
min           0.0                   0.0              0.0              1.0
25%           2.0                   0.0              2.0              5.0
50%           2.0                   0.0              2.0              5.0
75%           2.0                   0.0              2.0              5.0
max           5.0                   1.0              4.0             13.0

         Hue  0D280_0D315_of_diluted_wines  Proline  class
count  356.0                         356.0    356.0  356.0
mean     1.0                           3.0    747.0    2.0
std      0.0                           1.0    222.0    1.0
min      0.0                           1.0    278.0    1.0
25%      1.0                           3.0    674.0    2.0
50%      1.0                           3.0    747.0    2.0
75%      1.0                           3.0    747.0    2.0
max      2.0                           4.0   1680.0    3.0   2
```

Regarding in the results above, we can notice that Alcohol, Alcalinity_of_ash, Magnesium, and Proline columns may contain outliers

Using boxplots to visualize outliers

```
#Creating 4 subplots for the box plots of the 4 columns
plt.figure(figsize=(15,10))

plt.subplot(2, 2, 1)
fig = new_df2.boxplot(column='Alcohol')
fig.set_title('')
fig.set_ylabel('Alcohol')

plt.subplot(2, 2, 2)
fig = new_df2.boxplot(column='Alcalinity_of_ash')
fig.set_title('')
fig.set_ylabel('Alcalinity_of_ash')

plt.subplot(2, 2, 3)
fig = new_df2.boxplot(column='Magnesium')
fig.set_title('')
fig.set_ylabel('Magnesium')

plt.subplot(2, 2, 4)
fig = new_df2.boxplot(column='Proline')
fig.set_title('')
fig.set_ylabel('Proline')
```

```
Text(0, 0.5, 'Proline')
```



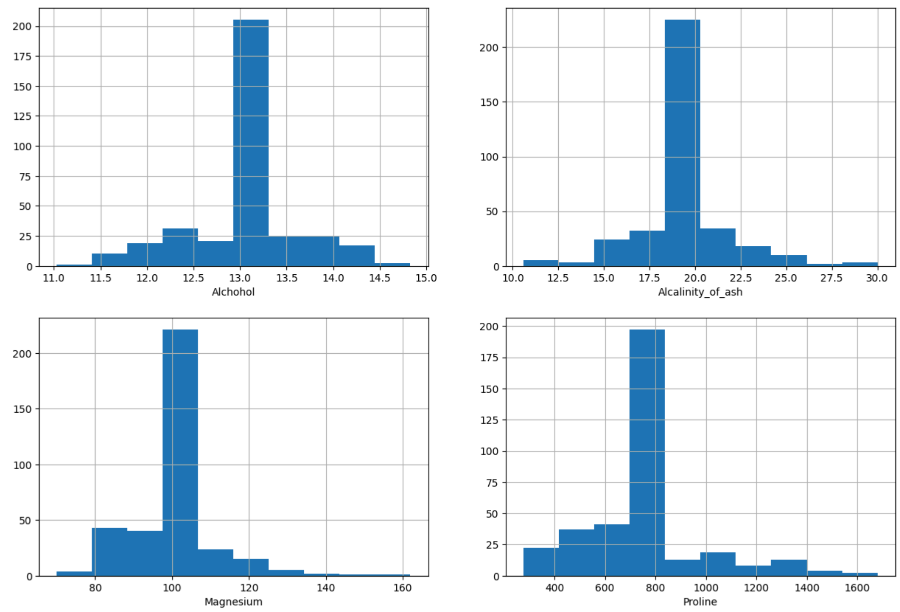
As seen in the result we can say that the boxplots confirm that there are a lot of outliers in the 4 given columns

Checking the Distribution of variables

Using histograms we can check if the distributions are normal or skewed

```
#Using plot histogram to check distribution
plt.figure(figsize=(15,10))

plt.subplot(2, 2, 1)
fig = new_df2.Alcohol.hist(bins=10)
fig.set_xlabel('Alchohol')

plt.subplot(2, 2, 2)
fig = new_df2.Alcalinity_of_ash.hist(bins=10)
fig.set_xlabel('Alcalinity_of_ash')

plt.subplot(2, 2, 3)
fig = new_df2.Magnesium.hist(bins=10)
fig.set_xlabel('Magnesium')

plt.subplot(2, 2, 4)
fig = new_df2.Proline.hist(bins=10)
fig.set_xlabel('Proline')
```

```
Text(0.5, 0, 'Proline')
```



It is noticable that the distributions of our given columns are skewed because of their visual distribution

```
new_df2.head()
```

	Alcohol	Malicacid	Ash	Alcalinity_of_ash	Magnesium	Total_phenols	Flavonoids	Nor
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	

Next steps: View recommended plots

Training a Regression Model

```
X = new_df2.drop(['Alcohol'], axis=1)
y = new_df2['Alcohol']
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=100)
```

```
X_train.shape, X_test.shape
```

```
((284, 13), (72, 13))
```

```
X_train.dtypes
```

```
Malicacid                       float64
Ash                             float64
Alcalinity_of_ash               float64
Magnesium                       float64
Total_phenols                   float64
Flavonoids                      float64
Nonflavonoid_phenols            float64
Proanthocyanins                 float64
Color_intensity                 float64
Hue                             float64
0D280_0D315_of_diluted_wines    float64
Proline                         float64
class                           float64
dtype: object
```

Double-click (or enter) to edit

```
#Identifying columns that are not Object types
numerical = [col for col in X_train.columns if X_train[col].dtypes != '0']
numerical
```

```
['Malicacid',
 'Ash',
 'Alcalinity_of_ash',
 'Magnesium',
 'Total_phenols',
 'Flavonoids',
 'Nonflavonoid_phenols',
 'Proanthocyanins',
 'Color_intensity',
 'Hue',
 '0D280_0D315_of_diluted_wines',
 'Proline',
 'class']
```

```
#Checking if there is a missing values in the columns
X_test[numerical].isnull().sum()
```

```
Malicacid                       0
Ash                             0
Alcalinity_of_ash               0
Magnesium                       0
Total_phenols                   0
Flavonoids                      0
```

```
Nonflavonoid_phenols            0
Proanthocyanins                 0
Color_intensity                 0
Hue                             0
0D280_0D315_of_diluted_wines    0
Proline                         0
class                           0
dtype: int64
```

```
pip install category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.6.3-py2.py3-none-any.whl (81 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81.9/81.9 kB 1.9 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.25.2)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.2.2)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.11.4)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.14.2)
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (2.0.3)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.5.6)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders) (2024.1
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_encoders) (
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_enco
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.9.0->category_encoders) (
Installing collected packages: category_encoders
Successfully installed category_encoders-2.6.3
```

```
X_train[numerical].columns
```

```
Index(['Malicacid', 'Ash', 'Alcalinity_of_ash', 'Magnesium', 'Total_phenols',
       'Flavonoids', 'Nonflavonoid_phenols', 'Proanthocyanins',
       'Color_intensity', 'Hue', '0D280_0D315_of_diluted_wines', 'Proline',
       'class'],
      dtype='object')
```

```
import category_encoders as ce

encoder = ce.BinaryEncoder(cols=['Malicacid'])
X_train = encoder.fit_transform(X_train)
X_test = encoder.transform(X_test)
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
cols = X_train.columns
```

```
X_train = pd.DataFrame(X_train, columns=[cols])
X_test = pd.DataFrame(X_test, columns=[cols])
```

Feature Scaling

```
X_train.describe()
```

	Malicacid	Ash	Alcalinity_of_ash	Magnesium	Total_phenols	Flavonoids
count	284.000000	284.000000	284.000000	284.000000	284.000000	284.000000

```
for new_df2 in [X_train, X_test]:
    new_df2[numerical] = new_df2[numerical].fillna(new_df2[numerical].median())

#Check for missing values in X_train after filling
print(X_train[numerical].isnull().sum())
```

```
Malicacid                       0
Ash                             0
Alcalinity_of_ash               0
Magnesium                       0
Total_phenols                   0
Flavonoids                      0
Nonflavonoid_phenols            0
Proanthocyanins                 0
Color_intensity                 0
Hue                             0
0D280_0D315_of_diluted_wines    0
Proline                         0
class                           0
dtype: int64
```

```
new_df2.isnull().sum()
```

```
Malicacid                       0
Ash                             0
Alcalinity_of_ash               0
Magnesium                       0
Total_phenols                   0
Flavonoids                      0
Nonflavonoid_phenols            0
Proanthocyanins                 0
Color_intensity                 0
Hue                             0
0D280_0D315_of_diluted_wines    0
Proline                         0
class                           0
dtype: int64
```

Model Training

```
linreg = LinearRegression()
linreg.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

Predict Results

```
y_pred_test = logreg.predict(X_test)

y_pred_test
```

```
array([ 47,  47,  47,  47,  47,  47,  47,  47,  47,  47,  47,  47,  47,
        47,  47,  47,  47,  47,  47,  21,  47,  47,  47,  47,  47,  47,
        47,  47,  47,  47,  47,  47,  47,  47,  47,  47,  47,  47,  47,
        47,  47, 102,  47,  47,  47,  47,  47,  47,  47,  47,  81,  47,
        47,  47,  47,  24,  47,  47,  24,  47,  47,  47,  47,  47,  47,
        47,  47,  47,  47,  47,  47,  47])
```

```
#Probability of getting output as 0
logreg.predict_proba(X_test)[:,0]
```

```
array([2.28377355e-03, 1.98028487e-04, 1.28994228e-04, 1.82656514e-02,
```