```
print("Good morning")
```

```
Good morning
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

Import Dataset

```
#Import data
import pandas as pd
data=pd.read_csv('winequality-red .csv')
data.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	su
0	NaN	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	

```
# Count the number of missing values in each column
print(data.isnull().sum())
```

```
fixed acidity           1
volatile acidity        0
citric acid             1
residual sugar          0
chlorides               0
free sulfur dioxide     1
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

Drop the rows containing missing values

```
# Drop any rows that contain missing values
data_dropped = data.dropna()
```

```
# Verify that there are no missing values in the new dataset
print(data_dropped.isnull().sum())
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

Mean Imputation

```
#Import data
import pandas as pd
data = pd.read_csv('winequality-red.csv')
data.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	su
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	

```
print(data.isnull().sum())
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

```
# Impute missing values with the mean value of the column
data_mean = data.fillna(data.mean())
```

```
# Verify that there are no missing values in the new dataset
print(data_mean.isnull().sum())
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

Median Imputation

```
#Import data
import pandas as pd
data = pd.read_csv('winequality-red.csv')
data.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	su
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	

```
# Impute missing values with the median value of the column
data_median = data.fillna(data.median())
# Verify that there are no missing values in the new dataset
print(data_median.isnull().sum())
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
```

```
density               0
pH                    0
sulphates             0
alcohol               0
quality               0
dtype: int64
```

Simple Imputer by Scikit learn

```
#scikit-learn's
#SimpleImputer class, which provides different strategies for handling missing valu
#We will demonstrate how to use three different strategies:
#mean imputation, median imputation, and most frequent imputation.
```

```
#Import data
import pandas as pd
data = pd.read_csv('winequality-red.csv')
data.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	su
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	

```
from sklearn.impute import SimpleImputer
# Create a SimpleImputer object with the mean strategy
mean_imputer = SimpleImputer(strategy='mean')
# Create a SimpleImputer object with the median strategy
median_imputer = SimpleImputer(strategy='median')
# Create a SimpleImputer object with the most frequent strategy
mode_imputer = SimpleImputer(strategy='most_frequent')
# Separate the target variable (quality) from the predictors (features)
X = data.drop('quality', axis=1)
y = data['quality']
# Fit the imputers to the data and transform the data
X_mean_imputed = mean_imputer.fit_transform(X)
X_median_imputed = median_imputer.fit_transform(X)
X_mode_imputed = mode_imputer.fit_transform(X)
# Verify that there are no missing values in the new datasets
print(pd.DataFrame(X_mean_imputed).isnull().sum())
print(pd.DataFrame(X_median_imputed).isnull().sum())
print(pd.DataFrame(X_mode_imputed).isnull().sum())
```

```
0     0
1     0
2     0
3     0
4     0
5     0
6     0
7     0
8     0
9     0
10    0
dtype: int64
0     0
1     0
2     0
3     0
4     0
5     0
6     0
7     0
8     0
9     0
10    0
dtype: int64
0     0
1     0
2     0
3     0
4     0
```

```
5     0
6     0
7     0
8     0
9     0
10    0
dtype: int64
```

Error since the feature 'type' is categorical

```
import numpy as np
color=data['pH']
color
```

```
0       3.51
1       3.20
2       3.26
3       3.16
4       3.51
        ...
1594    3.45
1595    3.52
1596    3.42
1597    3.57
1598    3.39
Name: pH, Length: 1599, dtype: float64
```

Drop Quality

```
X.head(3)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	su
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	

Drop type

```
from sklearn.impute import SimpleImputer
# Create a SimpleImputer object with the mean strategy
mean_imputer = SimpleImputer(strategy='mean')
# Create a SimpleImputer object with the median strategy
median_imputer = SimpleImputer(strategy='median')
# Create a SimpleImputer object with the most frequent strategy
mode_imputer = SimpleImputer(strategy='most_frequent')
# Separate the target variable (quality) from the predictors (features)
#X = X.drop('type', axis=1)

y = data['quality']
# Fit the imputers to the data and transform the data
X_mean_imputed = mean_imputer.fit_transform(X)
X_median_imputed = median_imputer.fit_transform(X)
X_mode_imputed = mode_imputer.fit_transform(X)
# Verify that there are no missing values in the new datasets
print(pd.DataFrame(X_mean_imputed).isnull().sum())
print(pd.DataFrame(X_median_imputed).isnull().sum())
print(pd.DataFrame(X_mode_imputed).isnull().sum())
```

```
0     0
1     0
2     0
3     0
4     0
5     0
6     0
7     0
8     0
9     0
10    0
dtype: int64
0     0
1     0
2     0
3     0
4     0
5     0
6     0
```

```
7     0
8     0
9     0
10    0
dtype: int64
0     0
1     0
2     0
3     0
4     0
5     0
6     0
7     0
8     0
9     0
10    0
dtype: int64
```

```
X.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	NaN	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	NaN	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	NaN	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	NaN	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	NaN	11.0	34.0	0.9978	3.51	0.56	9.4

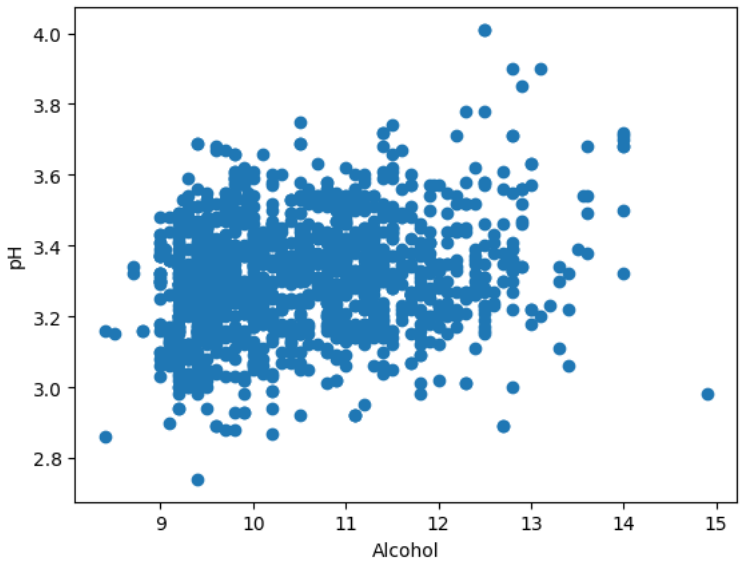
```
X['pH']=color
```

```
X.head()
```

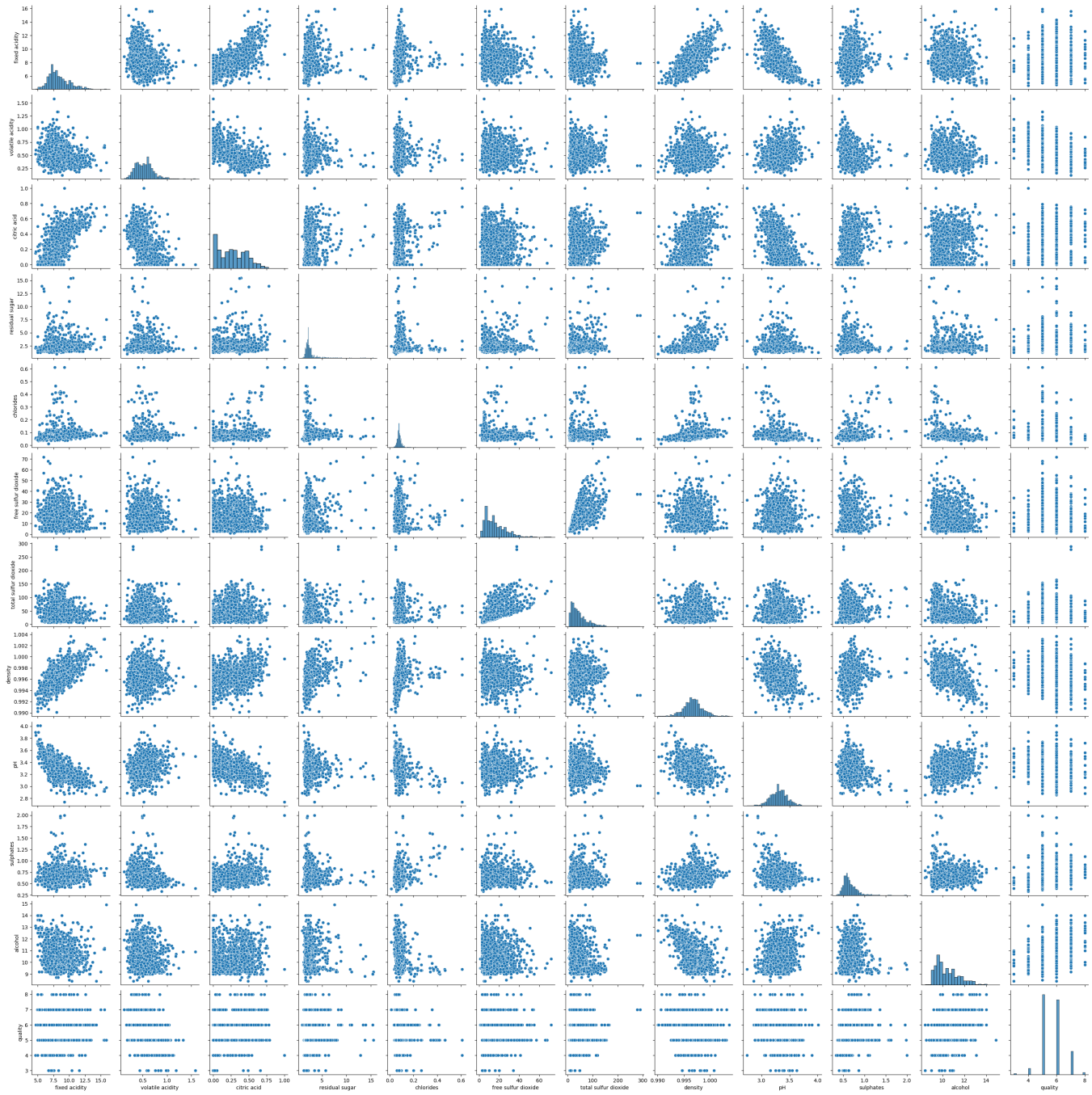
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	NaN	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	NaN	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	NaN	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	NaN	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	NaN	11.0	34.0	0.9978	3.51	0.56	9.4

Visualization

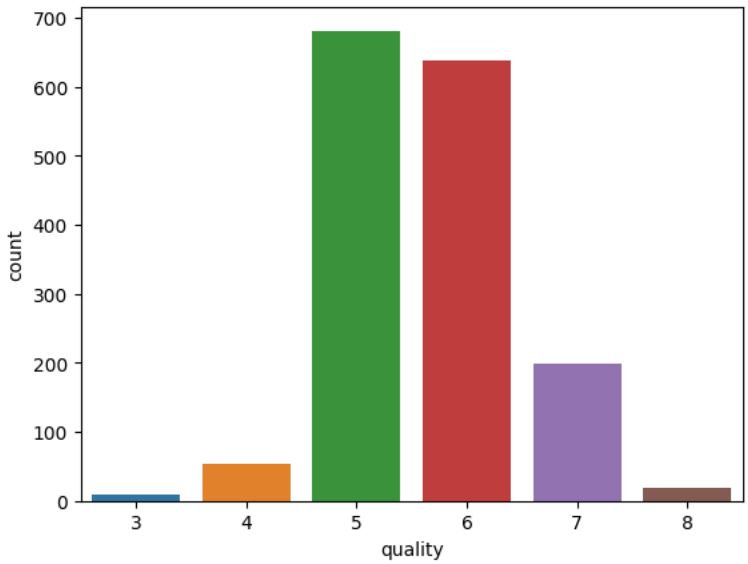
```
import matplotlib.pyplot as plt
# Create a scatter plot of alcohol content vs. pH
plt.scatter(data['alcohol'], data['pH'])
plt.xlabel('Alcohol')
plt.ylabel('pH')
plt.show()
```



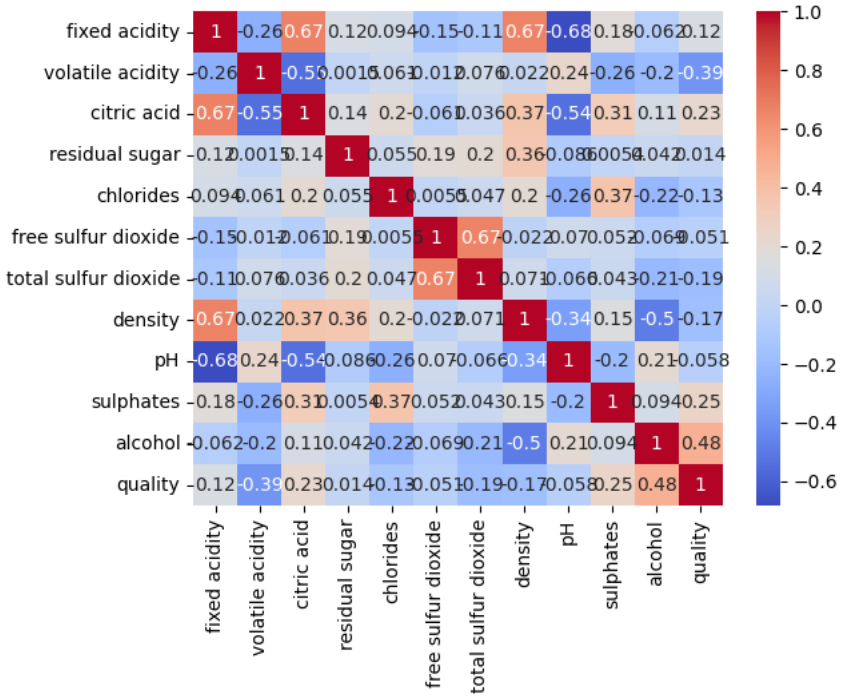
```
import seaborn as sns
# Create a pair plot of all columns in the dataset
sns.pairplot(data)
plt.show()
```



```
# Create a count plot of the "quality" column
sns.countplot(x='quality', data=data)
plt.show()
```




```
import seaborn as sns
# Create a heatmap of the correlation between all columns in the dataset
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.show()
```




```
from wordcloud import WordCloud
# Get column names of the dataset
column_names = data.columns.tolist()
# Create a word cloud plot of the column names
wordcloud = WordCloud(background_color='white').generate(' '.join(column_names))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```


```
pip install wordcloud
```

```
Collecting wordcloud
  Downloading wordcloud-1.8.2.2-cp39-cp39-win_amd64.whl (153 kB)
     ------------------------------------ 153.1/153.1 kB 2.3 MB/s eta 0:00:00
Requirement already satisfied: matplotlib in c:\users\divya\anaconda3\lib\site-packages (from wordcloud) (3.5.2)
Requirement already satisfied: pillow in c:\users\divya\anaconda3\lib\site-packages (from wordcloud) (9.2.0)
Requirement already satisfied: numpy>=1.6.1 in c:\users\divya\anaconda3\lib\site-packages (from wordcloud) (1.21.5)
Requirement already satisfied: cycler>=0.10 in c:\users\divya\anaconda3\lib\site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\divya\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.4.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\divya\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2.8
Requirement already satisfied: packaging>=20.0 in c:\users\divya\anaconda3\lib\site-packages (from matplotlib->wordcloud) (21.3)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\divya\anaconda3\lib\site-packages (from matplotlib->wordcloud) (4.25.0
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\divya\anaconda3\lib\site-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: six>=1.5 in c:\users\divya\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib->wordcl
Installing collected packages: wordcloud
Successfully installed wordcloud-1.8.2.2
Note: you may need to restart the kernel to use updated packages.
```

```
from wordcloud import WordCloud
# Get column names of the dataset
column_names = data.columns.tolist()
# Create a word cloud plot of the column names
wordcloud = WordCloud(background_color='white').generate(' '.join(column_names))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```