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

```
warnings.filterwarnings('ignore')
```

```
df = sns.load_dataset('iris')
```

```
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
df.shape
```

```
(150, 5)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
df.isnull().sum()
```

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

```
df.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Converting species column categorical values to numerical values

```
df['species'].value_counts()
```

```
setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
```

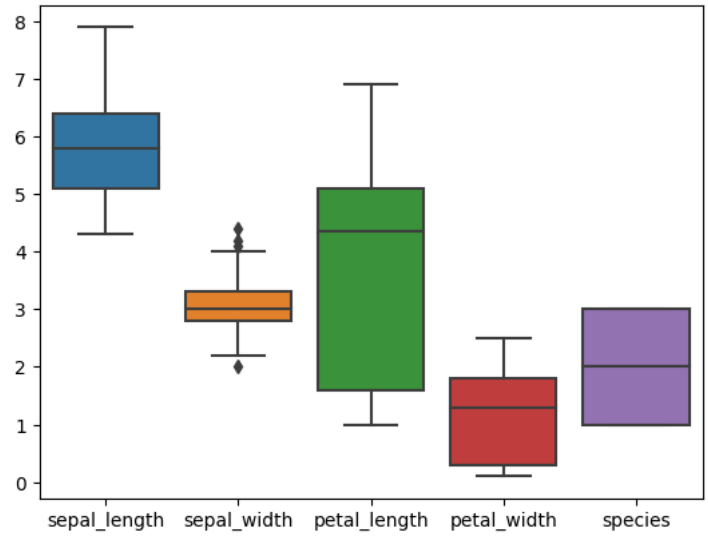
```
df.replace(['setosa', 'versicolor', 'virginica'], [1, 2, 3], inplace = True)
```

```
df.dtypes['species']
```

```
dtype('int64')
```

EDA (Exploratory Data Analysis)

```
sns.boxplot(data = df)
plt.show()
```

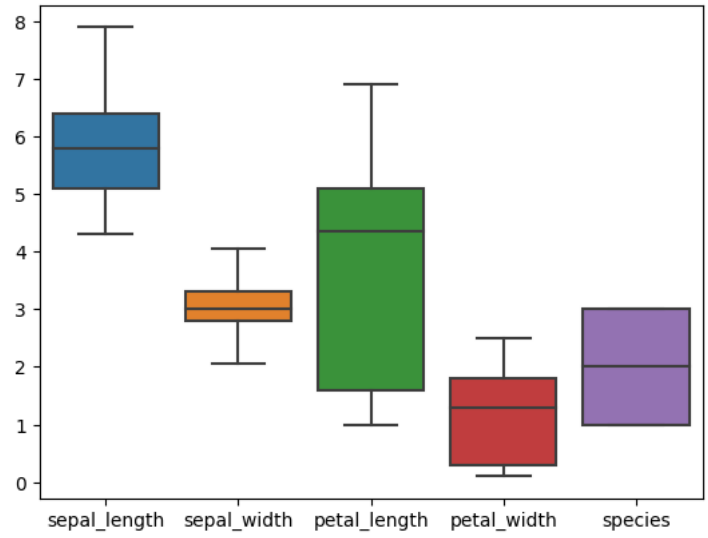



```
# function for Outlier replacement

def Replace(col):
  Q1 = df[col].quantile(0.25)
  Q3 = df[col].quantile(0.75)
  IQR = Q3 - Q1
  upper = Q3 + (1.5*IQR)
  lower = Q1 - (1.5*IQR)
  np.clip(df[col], lower, upper, inplace=True)
```

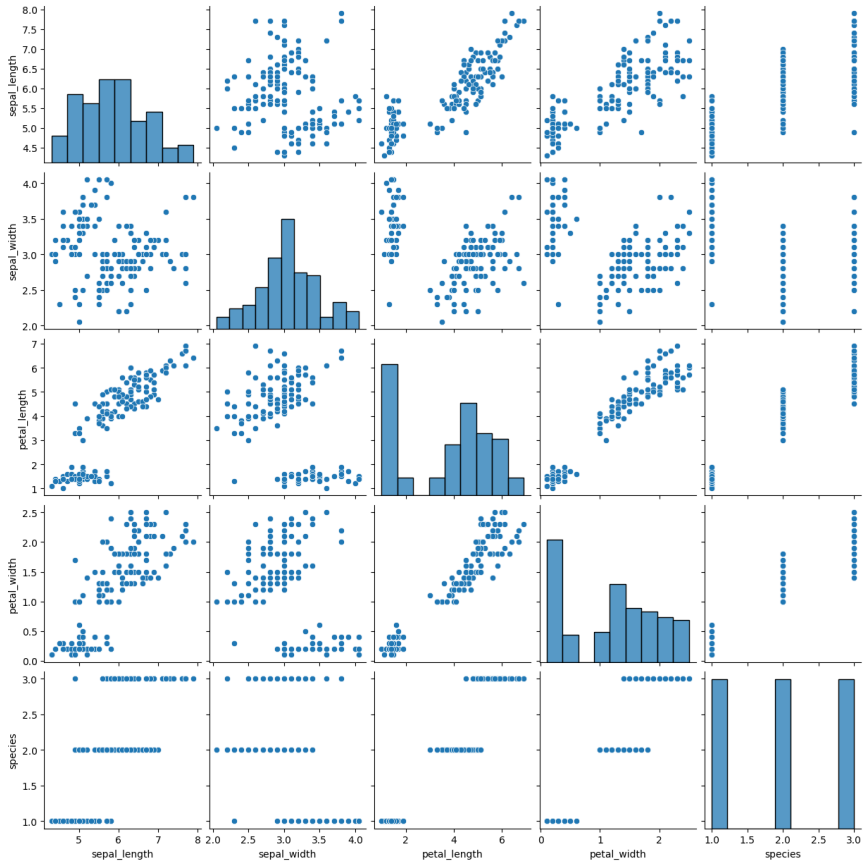
```
Replace('sepal_width')
```

```
sns.boxplot(data=df)
plt.show()
```



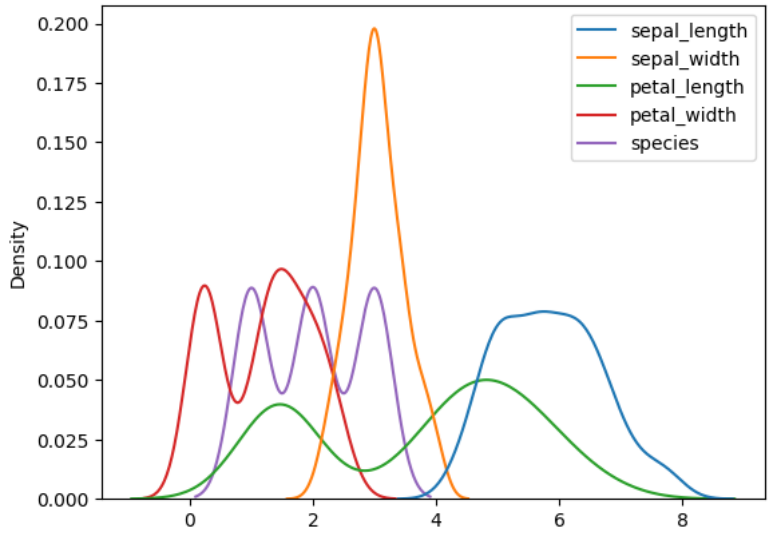
```
sns.pairplot(data=df)
```

```
<seaborn.axisgrid.PairGrid at 0x7f904f197bb0>
```



```
sns.kdeplot(data=df)
```

```
<Axes: ylabel='Density'>
```




```
# Navie Bayes Classification Model
x = df.iloc[ : , :-1]
y = df.iloc[ : ,-1]
```

```
x
```

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

```
y
```

```
0      1
1      1
2      1
3      1
4      1
      ..
145    3
146    3
147    3
148    3
149    3
Name: species, Length: 150, dtype: int64
```

```
# Training and Testing x and y
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8, random_state=0)
```

```
x_train.shape
```

```
(120, 4)
```

```
y_train.shape
```

```
(120,)
```

```
x_test.shape
```

```
(30, 4)
```

```
y_test.shape
```

```
(30,)
```

```
# Implementing Naive Bayes Model

from sklearn.naive_bayes import GaussianNB


classify = GaussianNB()


# Training model
classify.fit(x_train, y_train)
```

```
▾ GaussianNB
GaussianNB()
```

```
y_pred = classify.predict(x_test)


y_pred.shape
```

```
(30,)
```

```
# Evaluating the model
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score


confusion_matrix(y_test, y_pred)
```

```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  1,  5]])
```

```
accuracy_score(y_test, y_pred)
```

```
0.9666666666666667
```

```
precision_score(y_test, y_pred, average=None)
```

```
array([1.        , 0.92857143, 1.        ])
```

```
recall_score(y_test, y_pred, average=None)
```

```
array([1.        , 1.        , 0.83333333])
```