jupyter **Practical 1** Last Checkpoint: a minute ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
df=pd.read_csv('tiatanic.csv')
df
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [3]:
```
df.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [5]:
```
df.tail()
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

In [6]:
```
df.describe()
```

Out[6]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [10]:
```
df.shape
```

Out[10]:
```
(891, 12)
```

In [9]:
```
df['Age'].isnull().sum()
```

Out[9]:
```
177
```

In [10]:
```
df['Cabin'].isnull().sum()
```

Out[10]:
```
687
```

In [11]:
```
df.describe()
```

Out[11]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [13]:
```
df.dtypes
```

Out[13]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [14]:
```
df.shape
```

Out[14]:
```
(891, 12)
```

In [15]:
```
def impute_age(cols):
    Age = cols[0]
    Pclass = cols[1]
    if pd.isnull(Age):
        if Pclass == 1:
            return 37
        elif Pclass == 2:
            return 29
        else:
            return 24

    else:
        return Age
```

In [16]:
```
df['Age']=df[['Age','Pclass']].apply(impute_age,axis=1)
```

In [17]:
```
df.drop('Cabin',axis=1,inplace=True)
```

In [18]:
```
df.dropna(inplace=True)
```

In [19]:
```
df
```

Out[19]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S
...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	24.0	1	2	W./C. 6607	23.4500	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	Q

889 rows × 11 columns

jupyter Practicle2 Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

In [45]:
```
import pandas as pd
import numpy as np

# Generate sample data
np.random.seed(0)
data = {
    'Student_ID': range(1, 101),
    'Math': np.random.randint(50, 100, 100).astype(object),
    'Physics': np.random.randint(40, 95, 100),
    'English': np.random.randint(30, 90, 100).astype(object),
    'Attendance': np.random.uniform(0.7, 1, 100).astype(object),
    'GPA': np.random.uniform(2.5, 4, 100)
}

# Introduce some missing values and inconsistencies
data['Math'][10] = np.nan
data['Physics'][20] = 200  # introducing an outlier
data['English'][30] = 'A'  # introducing inconsistency

# Create DataFrame
df = pd.DataFrame(data)
```

In [46]:
```
df
```

Out[46]:

	Student_ID	Math	Physics	English	Attendance	GPA
0	1	94	45	63	0.983236	3.573342
1	2	97	81	70	0.882476	3.229239
2	3	50	75	62	0.878997	3.562822
3	4	53	40	66	0.935093	3.247210
4	5	53	71	89	0.850008	3.766825
...	...	...	...	...	...	...
95	96	65	65	30	0.881943	3.574366
96	97	63	56	41	0.765521	2.540603
97	98	71	94	64	0.836551	3.597096
98	99	98	93	35	0.963661	3.650445
99	100	99	59	83	0.84768	2.514650

100 rows × 6 columns

In [47]:
```
missing_values = df.isnull().sum()
print("Missing Values:\n", missing_values)
```

```
Missing Values:
 Student_ID    0
Math           1
Physics        0
English        0
Attendance     0
GPA            0
dtype: int64
```

In [48]:
```
# Check for inconsistencies in 'English_Score'
inconsistent_data = pd.to_numeric(df['English'], errors='coerce').isnull().sum()
print("Inconsistencies in English_Score:", inconsistent_data)
```

```
Inconsistencies in English_Score: 1
```

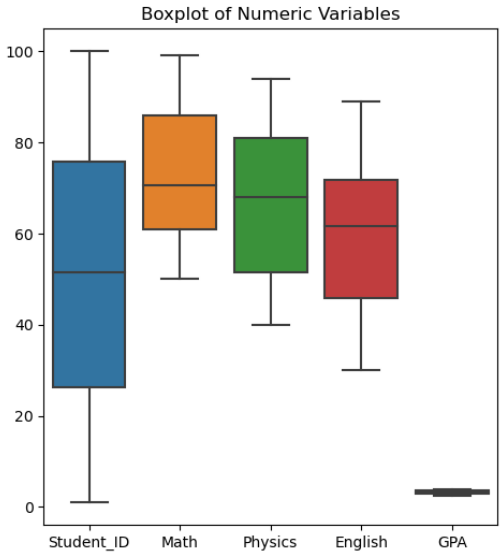
In [49]:
```
# Dealing with missing values and inconsistencies
# For missing values, we can either drop rows or fill them with a suitable value
df['Math'].fillna(df['Math'].mean(), inplace=True)
df['Physics'] = np.where(df['Physics'] > 100, np.nan, df['Physics'])  # replacing outlier with NaN
df['English'] = pd.to_numeric(df['English'], errors='coerce')  # Convert to numeric, converting inconsistencies to NaN
df.dropna(inplace=True)  # Drop rows with NaN values
```

In [50]:
```
# Check if missing values and inconsistencies are handled
print("After handling missing values and inconsistencies:")
print(df.isnull().sum())
```

```
After handling missing values and inconsistencies:
Student_ID    0
Math          0
Physics       0
English       0
Attendance    0
GPA           0
dtype: int64
```

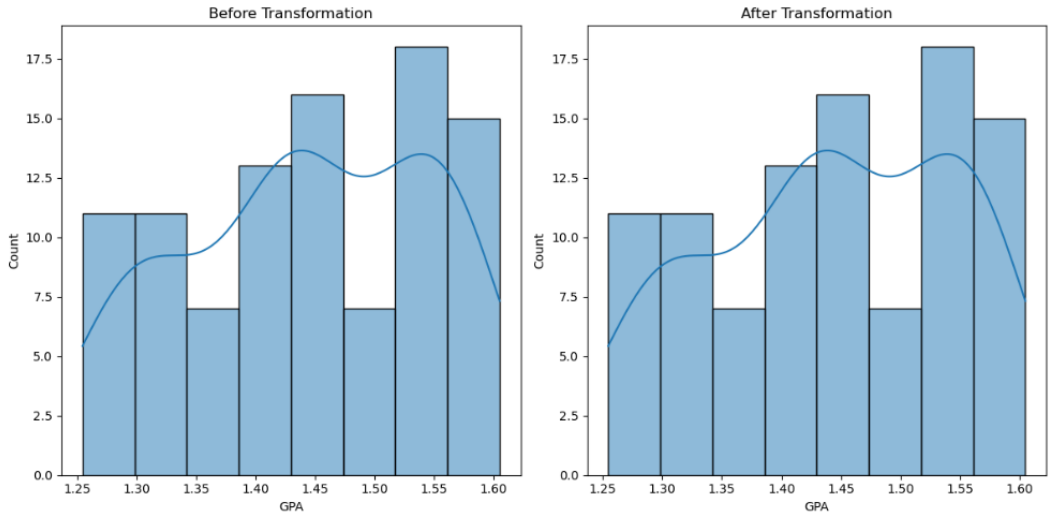
In [51]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import zscore
numeric_cols = df.select_dtypes(include=np.number).columns.tolist()
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.boxplot(data=df[numeric_cols])
plt.title('Boxplot of Numeric Variables')
```

Out[51]:
```
Text(0.5, 1.0, 'Boxplot of Numeric Variables')
```



In [52]:
```
# Data transformation on 'GPA' variable
# Log transformation to decrease skewness
z_scores = np.abs(zscore(df[numeric_cols]))
threshold = 3
outlier_indices = np.where(z_scores > threshold)[0]
df_cleaned = df.drop(outlier_indices)
df_cleaned.dropna(subset=['GPA'], inplace=True)
df_cleaned['GPA'] = np.log1p(df_cleaned['GPA'])
```

In [53]:
```
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(df_cleaned['GPA'], kde=True)
plt.title('Before Transformation')

plt.subplot(1, 2, 2)
sns.histplot(np.log1p(df['GPA']), kde=True)
plt.title('After Transformation')

plt.tight_layout()
plt.show()
```



jupyter Practical 3 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn import datasets
```

In [2]:
```
df = pd.read_csv('Iris.csv')
df.head()
```

Out[2]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [5]:
```
df = df.drop(columns= ['Id'])
df.head()
```

Out[5]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [6]:
```
df.describe()
```

Out[6]:

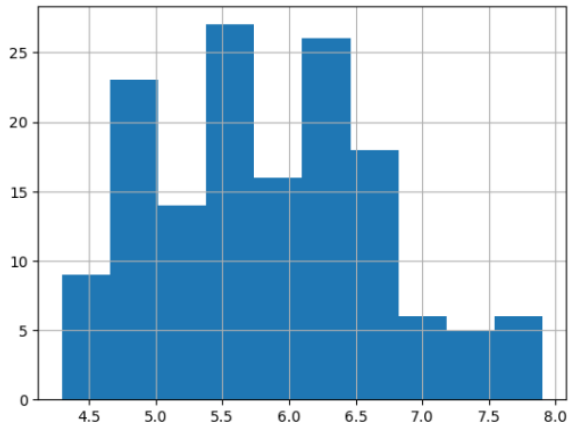
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [7]:
```
df['Species'].value_counts()
```

Out[7]:
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

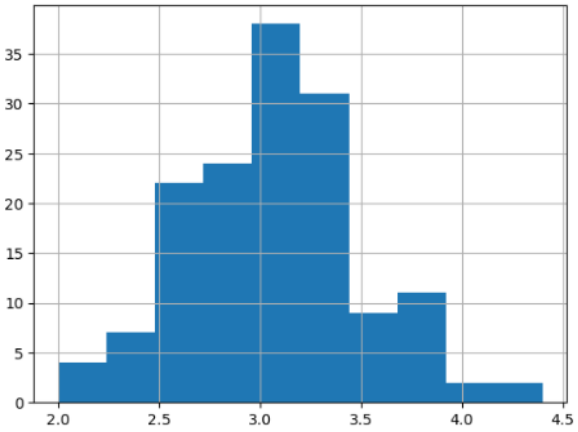
In [8]:
```
df['SepalLengthCm'].hist()
```

Out[8]: <Axes: >



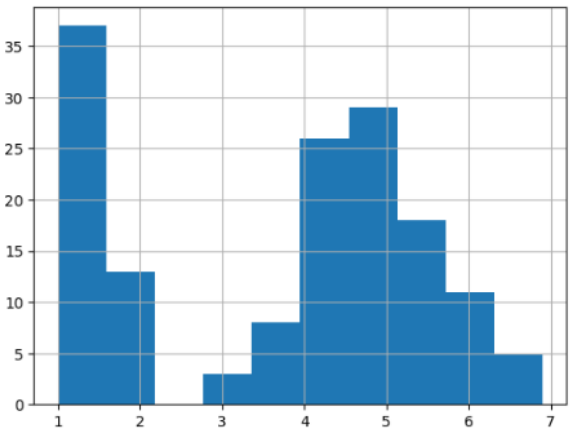
In [9]:
```
df['SepalWidthCm'].hist()
```

Out[9]: <Axes: >



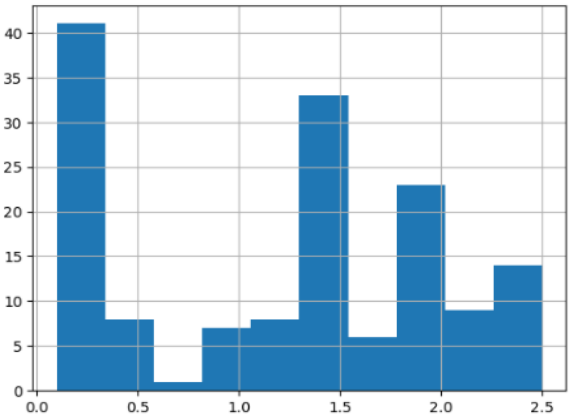
In [10]:
```
df['PetalLengthCm'].hist()
```

Out[10]: <Axes: >



In [11]:
```
df['PetalWidthCm'].hist()
```

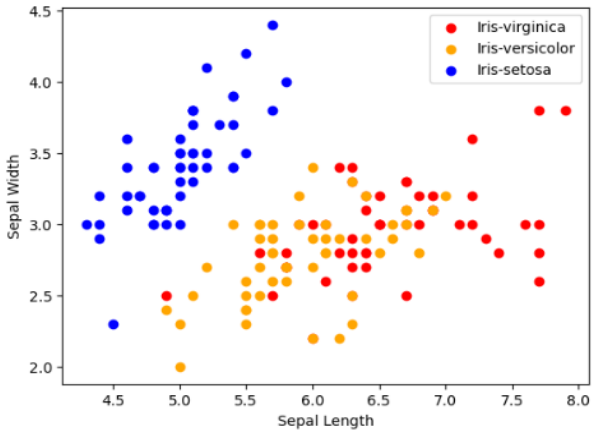
Out[11]: <Axes: >



In [12]:
```
colors = ['red', 'orange', 'blue']
species = ['Iris-virginica','Iris-versicolor','Iris-setosa']
```

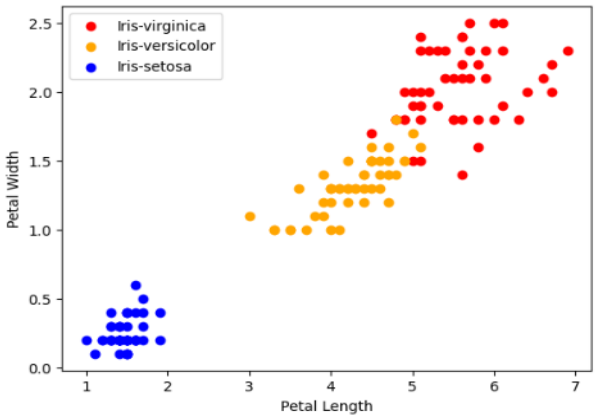
In [13]:
```
for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['SepalLengthCm'], x['SepalWidthCm'], c = colors[i], label=species[i])
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.legend()
```

Out[13]: <matplotlib.legend.Legend at 0x1c1b32a4390>



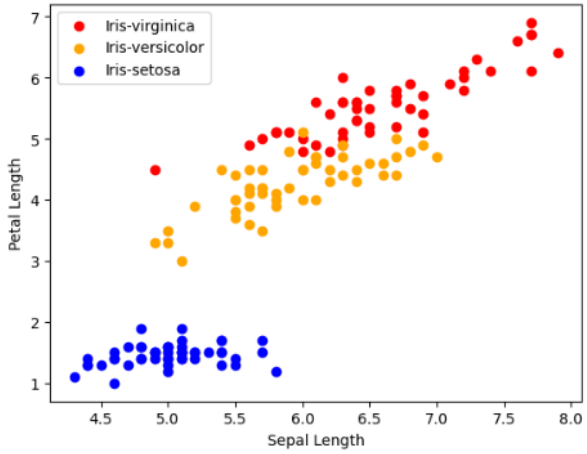
In [14]:
```
for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['PetalLengthCm'], x['PetalWidthCm'], c = colors[i], label=species[i])
plt.xlabel("Petal Length")
plt.ylabel("Petal Width")
plt.legend()
```

Out[14]: <matplotlib.legend.Legend at 0x1c1b336fa10>



In [15]:
```
for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['SepalLengthCm'], x['PetalLengthCm'], c = colors[i], label=species[i])
plt.xlabel("Sepal Length")
plt.ylabel("Petal Length")
plt.legend()
```

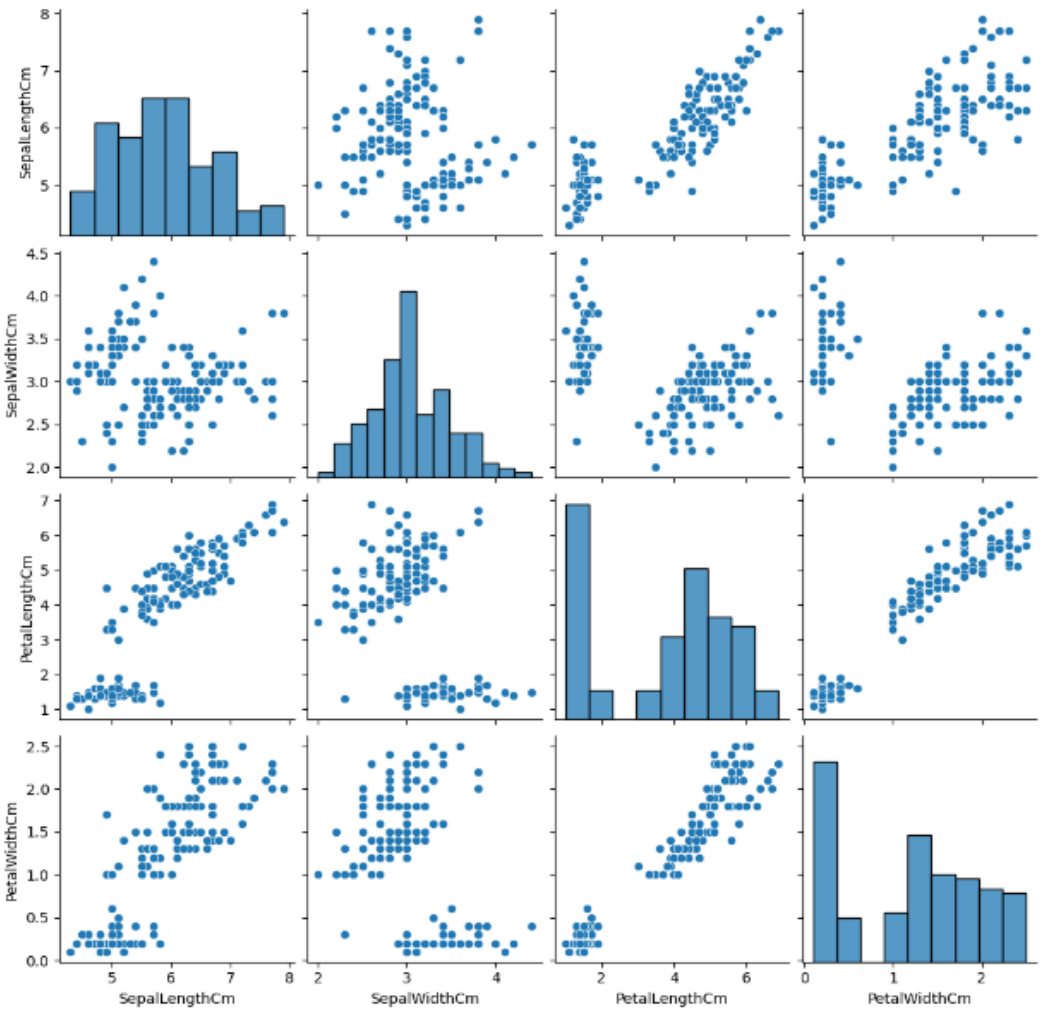
Out[15]: <matplotlib.legend.Legend at 0x1c1b2fdf6d0>



In [16]:
```
sns.pairplot(df)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[16]: <seaborn.axisgrid.PairGrid at 0x1c1b30fbe90>



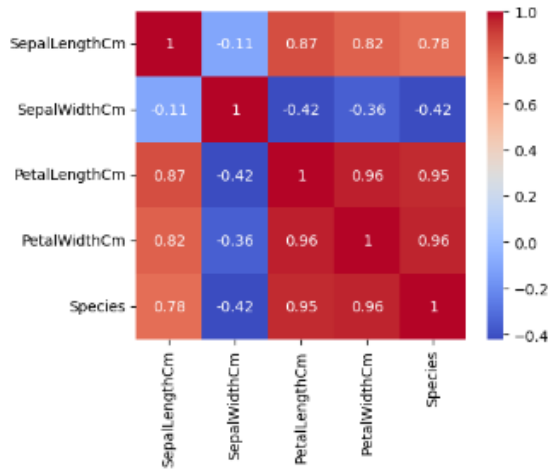
In [28]:
```
df.corr()
```

Out[28]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954	0.782561
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544	-0.419446
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757	0.949043
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000	0.956464
Species	0.782561	-0.419446	0.949043	0.956464	1.000000

In [27]:
```
corr = df.corr()
fig, ax = plt.subplots(figsize=(5,4))
sns.heatmap(corr, annot=True, ax=ax, cmap = 'coolwarm')
```

Out[27]: <Axes: >



In [21]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [22]:
```
df['Species'] = le.fit_transform(df['Species'])
df.head()
```

Out[22]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [23]:
```
from sklearn.model_selection import train_test_split
# train - 70
# test - 30
X = df.drop(columns=['Species'])
Y = df['Species']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.30)
```

In [24]:
```
# knn - k-nearest neighbours
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
```

In [25]:
```
model.fit(x_train, y_train)
```

Out[25]:
```
KNeighborsClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [26]:
```
print("Accuracy: ",model.score(x_test, y_test) * 100)
```

```
Accuracy:  97.77777777777777
```

jupyter Practical 4 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

import pandas as pd
import seaborn as sns

%matplotlib inline
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\__init__.py:155: UserWarning: A NumPy version >=1.18.5 and <1.25.0 is required
for this version of SciPy (detected version 1.26.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [3]:
```
from sklearn.datasets import load_boston
boston_dataset = load_boston()

boston_dataset.keys()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function load_boston is deprecated;
`load_boston` is deprecated in 1.0 and will be removed in 1.2.

    The Boston housing prices dataset has an ethical problem. You can refer to
    the documentation of this function for further details.

    The scikit-learn maintainers therefore strongly discourage the use of this
    dataset unless the purpose of the code is to study and educate about
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the original
    source::

        import pandas as pd
        import numpy as np


        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
        data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames housing
    dataset. You can load the datasets as follows::
```

Out[3]:
```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename', 'data_module'])
```

In [4]:
```
boston = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)
boston.head()
```

Out[4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

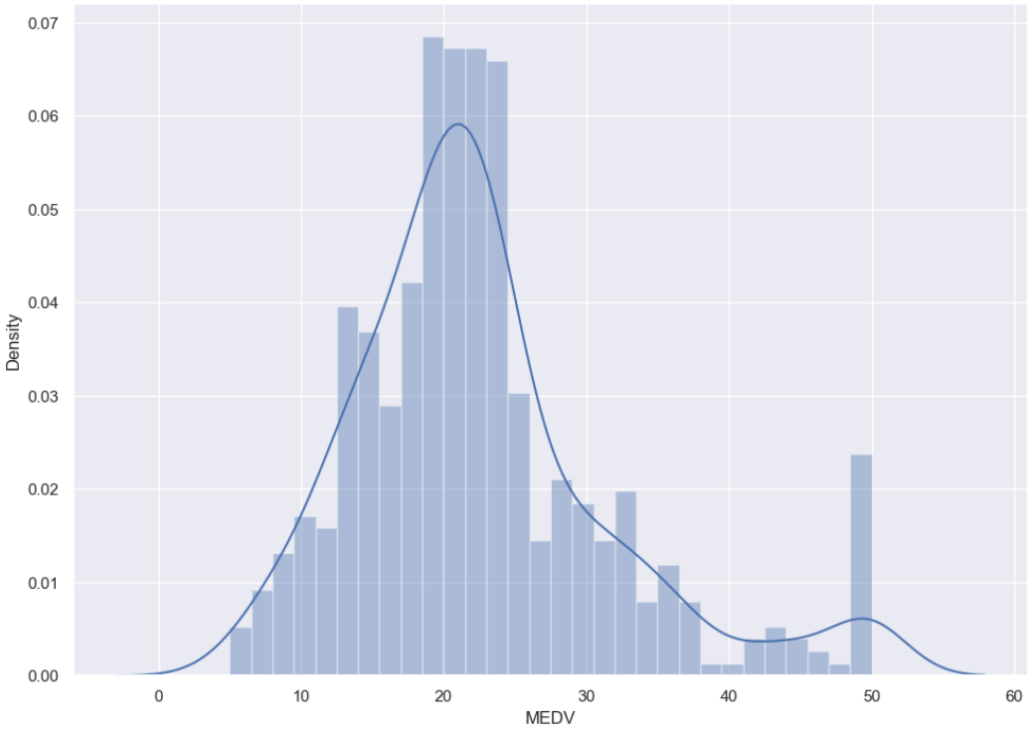
In [5]:
```
boston['MEDV'] = boston_dataset.target
```

In [6]:
```
boston.isnull().sum()
```

Out[6]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

In [7]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})

sns.distplot(boston['MEDV'], bins=30)
plt.show()
```

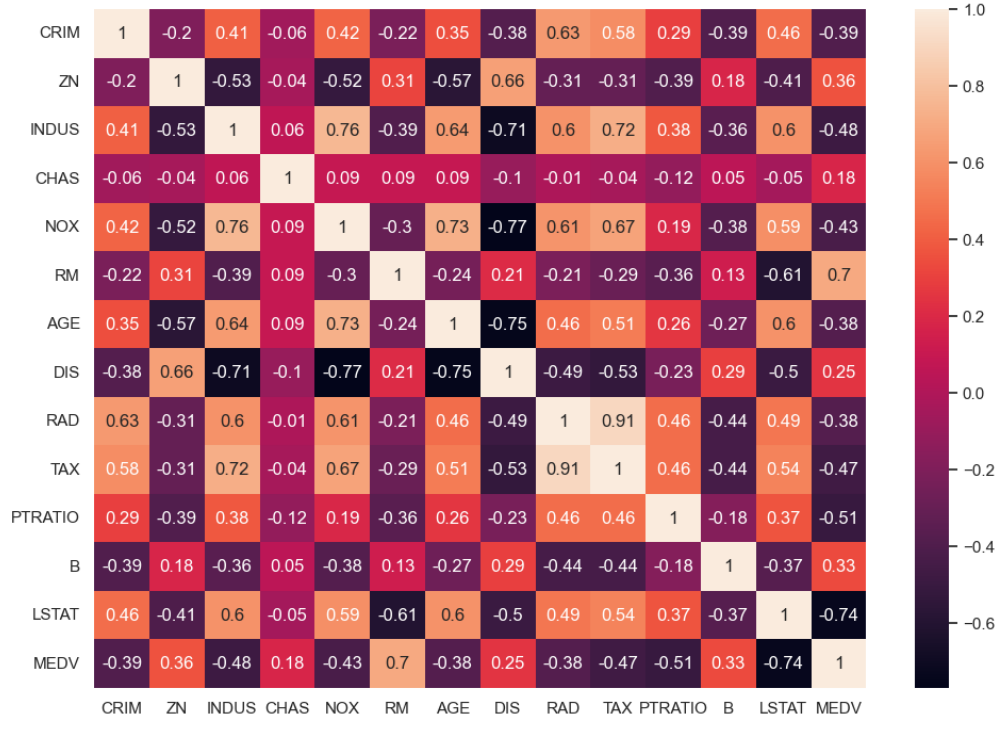
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



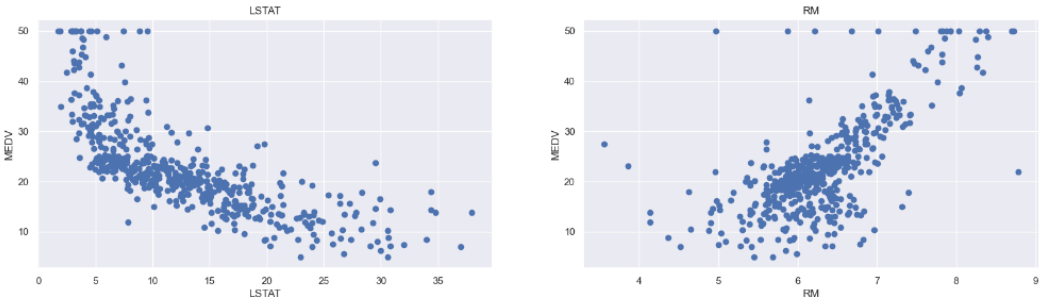
In [9]:
```
correlation_matrix = boston.corr().round(2)
```

In [10]:
```
sns.heatmap(data = correlation_matrix, annot=True)
```

Out[10]: <AxesSubplot:>



In [15]:
```
plt.figure(figsize=(20,5))

features = ["LSTAT", 'RM']
target = boston['MEDV']

for i, col in enumerate(features):
    plt.subplot(1,len(features) , i+1)
    x = boston[col]
    y = target
    plt.scatter(x,y,marker='o')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('MEDV')
```



In [19]:
```
X = pd.DataFrame(np.c_[boston['LSTAT'],boston['RM']], columns = ['LSTAT','RM'])
Y = boston['MEDV']
```

In [21]:
```
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=20)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(404, 2)
(102, 2)
(404,)
(102,)
```

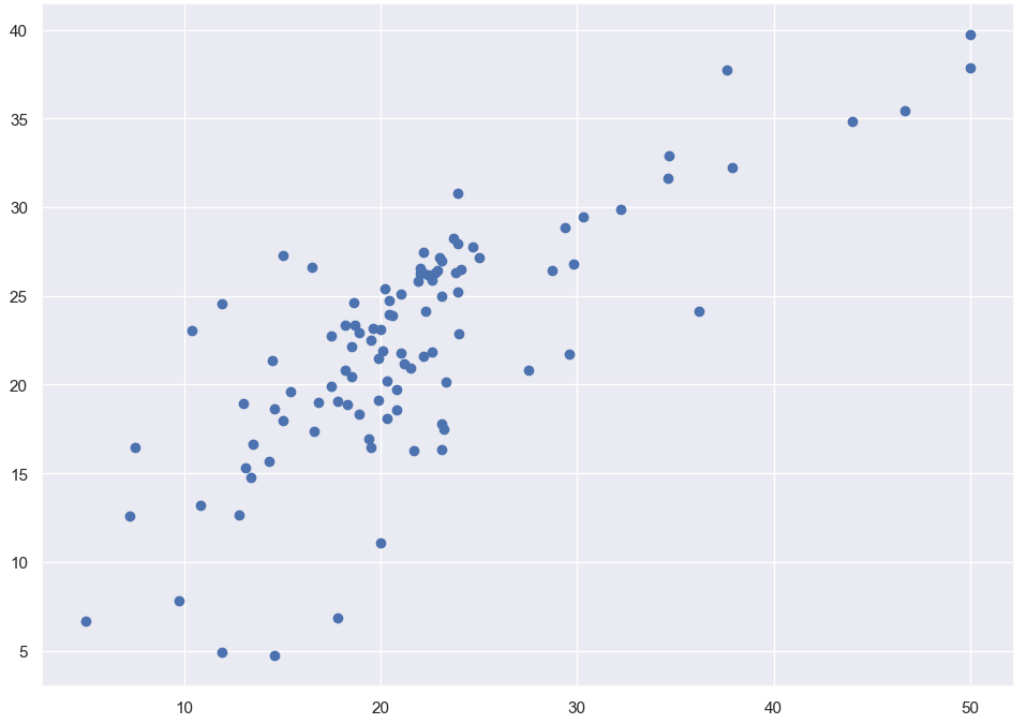
In [23]:
```
#Train the model using sklearnLinearRegression

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score

lin_model = LinearRegression()
lin_model.fit(X_train,Y_train)
```

Out[23]:
```
LinearRegression()
```

In [25]:
```
y_train_predict = lin_model.predict(X_train)
rmse = (np.sqrt(mean_squared_error(Y_train,y_train_predict)))
r2 = r2_score(Y_train,y_train_predict)

print("The model performance for testing set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))


y_test_predict = lin_model.predict(X_test)

rmse = (np.sqrt(mean_squared_error(Y_test,y_test_predict)))

r2 = r2_score(Y_test, y_test_predict)

print("The model performance for testing set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
```

```
The model performance for testing set
--------------------------------------
RMSE is 5.612505753798557
R2 score is 0.6468915821243122
The model performance for testing set
--------------------------------------
RMSE is 5.175217627561771
R2 score is 0.5841519194311253
```

In [26]:
```
plt.scatter(Y_test,y_test_predict)
plt.show()
```



In [ ]:

jupyter Practicle5 Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay,classification_report,accuracy_score, precision_score, recall
```

In [2]:
```
data =  pd.read_csv('Social_Network_Ads.csv')
data.head(5)
```

Out[2]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

In [4]:
```
data.describe()
```

Out[4]:

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

In [5]:
```
data.isnull().sum()
```

Out[5]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

In [6]:
```
data.shape
```

Out[6]:
```
(400, 5)
```

In [7]:
```
x = data.iloc[:,2:4]
y = data.iloc[:,4]
```

In [8]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42)
```

In [9]:
```
scale = StandardScaler()
x_train = scale.fit_transform(x_train)
x_test = scale.transform(x_test)
```

In [10]:
```
lr = LogisticRegression(random_state = 0,solver = 'lbfgs')
lr.fit(x_train,y_train)
pred = lr.predict(x_test)

print(x_test[:10])
print('-'*15)
print(pred[:10])
```

```
[[ 0.812419   -1.39920777]
 [ 2.0889839   0.52871943]
 [-0.95513241 -0.75656537]
 [ 1.0088136   0.76240757]
 [-0.85693511 -1.22394166]
 [-0.75873781 -0.23076704]
 [ 0.9106163   1.08372877]
 [-0.85693511  0.38266434]
 [ 0.2232352   0.14897619]
 [ 0.4196298  -0.14313399]]
---------------
[0 1 0 1 0 0 1 0 0 0]
```

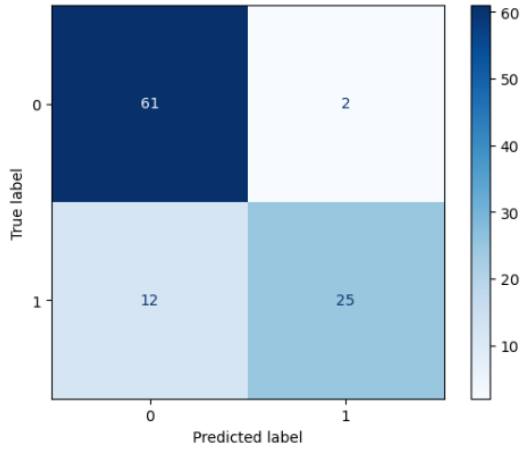
In [11]:
```
print('Expected Output:',pred[:10])
print('-'*15)
print('Predicted Output:\n',y_test[:10])
```

```
Expected Output: [0 1 0 1 0 0 1 0 0 0]
---------------
Predicted Output:
 209    0
280    1
33     0
210    1
93     0
84     0
329    1
94     0
266    0
126    0
Name: Purchased, dtype: int64
```

In [12]:
```
matrix = confusion_matrix(y_test,pred,labels = lr.classes_)
print(matrix)

tp, fn, fp, tn = confusion_matrix(y_test,pred,labels=[1,0]).reshape(-1)
```

```
[[61  2]
 [12 25]]
```

In [13]:
```
conf_matrix = ConfusionMatrixDisplay(confusion_matrix=matrix,display_labels=lr.classes_)
conf_matrix.plot(cmap=plt.cm.Blues)
plt.show()
```



In [14]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.97      0.90        63
           1       0.93      0.68      0.78        37

    accuracy                           0.86       100
   macro avg       0.88      0.82      0.84       100
weighted avg       0.87      0.86      0.85       100
```

In [15]:
```
print('\nAccuracy: {:.2f}'.format(accuracy_score(y_test,pred)))
print('Error Rate: ',(fp+fn)/(tp+tn+fn+fp))
print('Sensitivity (Recall or True positive rate) :',tp/(tp+fn))
print('Specificity (True negative rate) :',tn/(fp+tn))
print('Precision (Positive predictive value) :',tp/(tp+fp))
print('False Positive Rate :',fp/(tn+fp))
```

```
Accuracy: 0.86
Error Rate:  0.14
Sensitivity (Recall or True positive rate) : 0.6756756756756757
Specificity (True negative rate) : 0.9682539682539683
Precision (Positive predictive value) : 0.9259259259259259
False Positive Rate : 0.031746031746031744
```

In [ ]:

jupyter Practicle6 Last Checkpoint: a few seconds ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Checkpoint created: 00:45:24 | Trusted | Python 3 (ipykernel)

Run | Code

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay,classification_report,accuracy_score, precision_score, recall
from sklearn.preprocessing import LabelEncoder
```

In [2]:
```
data =  pd.read_csv('Iris.csv')
data.head(5)
```

Out[2]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [3]:
```
data.describe(include = 'all')
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	150.000000	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	NaN	Iris-setosa
freq	NaN	NaN	NaN	NaN	NaN	50
mean	75.500000	5.843333	3.054000	3.758667	1.198667	NaN
std	43.445368	0.828066	0.433594	1.764420	0.763161	NaN
min	1.000000	4.300000	2.000000	1.000000	0.100000	NaN
25%	38.250000	5.100000	2.800000	1.600000	0.300000	NaN
50%	75.500000	5.800000	3.000000	4.350000	1.300000	NaN
75%	112.750000	6.400000	3.300000	5.100000	1.800000	NaN
max	150.000000	7.900000	4.400000	6.900000	2.500000	NaN

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [5]:
```
print(data.shape)
data['Species'].unique()
```

```
(150, 6)
```

Out[5]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [7]:
```
x = data.iloc[:,1:5]
y = data.iloc[:,5:]
```

In [8]:
```
encode = LabelEncoder()
y = encode.fit_transform(y)
```

```
C:\Users\ghans\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.py:114: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [9]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,random_state = 0)
```

In [10]:
```
naive_bayes = GaussianNB()
naive_bayes.fit(x_train,y_train)
pred = naive_bayes.predict(x_test)
```

In [11]:
```
pred
```

Out[11]:
```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0, 2, 1, 0, 2, 2, 1, 0, 1, 1, 1, 2, 0, 2, 0,
       0])
```

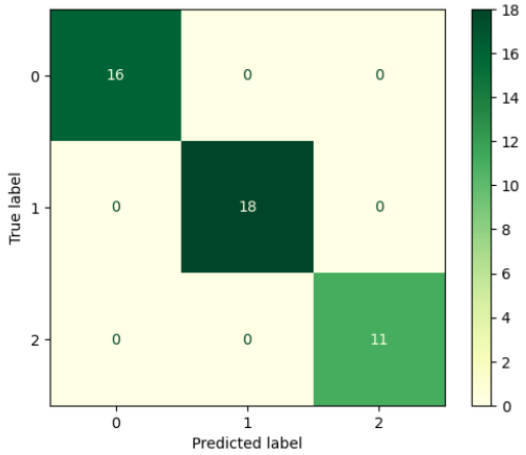
In [12]:
```
y_test
```

Out[12]:
```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0, 2, 1, 0, 2, 2, 1, 0, 1, 1, 1, 2, 0, 2, 0,
       0])
```

In [13]:
```
matrix =  confusion_matrix(y_test,pred,labels = naive_bayes.classes_)
print(matrix)

tp, fn, fp, tn = confusion_matrix(y_test,pred,labels=[1,0]).reshape(-1)
```

```
[[16  0  0]
 [ 0 18  0]
 [ 0  0 11]]
```

In [14]:
```
conf_matrix = ConfusionMatrixDisplay(confusion_matrix=matrix,display_labels=naive_bayes.classes_)
conf_matrix.plot(cmap=plt.cm.YlGn)
plt.show()
```



In [15]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      1.00      1.00        18
           2       1.00      1.00      1.00        11

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

In [16]:
```
print('\nAccuracy: {:.2f}'.format(accuracy_score(y_test,pred)))
print('Error Rate: ',(fp+fn)/(tp+tn+fn+fp))
print('Sensitivity (Recall or True positive rate) :',tp/(tp+fn))
print('Specificity (True negative rate) :',tn/(fp+tn))
print('Precision (Positive predictive value) :',tp/(tp+fp))
print('False Positive Rate :',fp/(tn+fp))
```

```
Accuracy: 1.00
Error Rate:  0.0
Sensitivity (Recall or True positive rate) : 1.0
Specificity (True negative rate) : 1.0
Precision (Positive predictive value) : 1.0
False Positive Rate : 0.0
```

In [ ]:

jupyter Practicle7 Last Checkpoint: a few seconds ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
!pip install nltk
```

```
Collecting nltk
  Obtaining dependency information for nltk from https://files.pythonhosted.org/packages/a6/0a/0d20d2c0f16be91b9fa32a77b76c60f9baf6eba419e5ef5deca17af9c582/nltk-3.8.1-py3-none-any.whl.metadata
  Downloading nltk-3.8.1-py3-none-any.whl.metadata (2.8 kB)
Requirement already satisfied: click in c:\users\ghans\appdata\local\programs\python\python311\lib\site-packages (from nltk) (8.1.3)

Installing collected packages: tqdm, regex, joblib, nltk
Successfully installed joblib-1.4.0 nltk-3.8.1 regex-2023.12.25 tqdm-4.66.2


[notice] A new release of pip is available: 23.2.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [2]:
```
import nltk
```

In [6]:
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\ghans\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

Out[6]:
```
True
```

In [7]:
```
from nltk.tokenize import sent_tokenize
text="""Hello Mr.smith,how are you doing today? The weather is great, and city is awesome. The sky is pinkish-blue. You shouldn't
tokenized_text = sent_tokenize(text)
print(tokenized_text)
```

```
['Hello Mr.smith,how are you doing today?', 'The weather is great, and city is awesome.', 'The sky is pinkish-blue.', "You shou
ldn't eat cardboard"]
```

In [8]:
```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['Hello', 'Mr.smith', ',', 'how', 'are', 'you', 'doing', 'today', '?', 'The', 'weather', 'is', 'great', ',', 'and', 'city', 'i
s', 'awesome', '.', 'The', 'sky', 'is', 'pinkish-blue', '.', 'You', 'should', "n't", 'eat', 'cardboard']
```

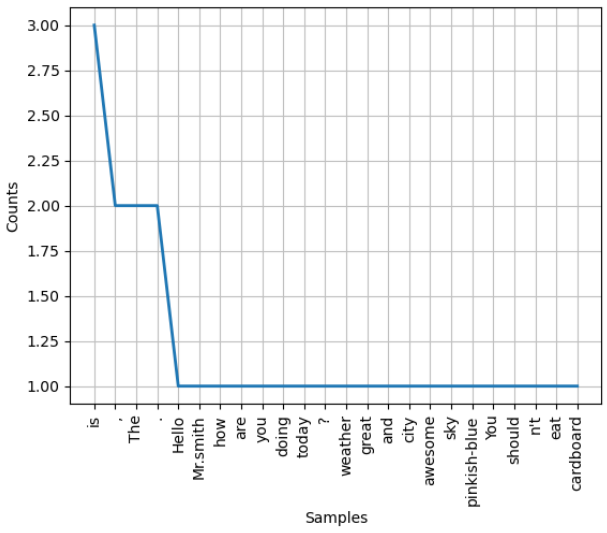
In [9]:
```
from nltk.probability import FreqDist
fdist = FreqDist(tokenized_word)
print(fdist)
```

```
<FreqDist with 24 samples and 29 outcomes>
```

In [10]:
```
fdist.most_common(2)
```

Out[10]:
```
[('is', 3), (',', 2)]
```

In [11]:
```
import matplotlib.pyplot as plt
fdist.plot(30,cumulative=False)
plt.show()
```



In [12]:
```
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words("english"))
print(stop_words)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ghans\AppData\Roaming\nltk_data...
```

```
{"that'll", 'does', 'shan', 'are', 'has', 'your', 'wasn', "you've", "she's", 'they', 'some', 'mightn', 'over', 'myself', 'own',
'themselves', 'ma', "won't", 'couldn', 't', 'haven', 've', 'i', 'under', 'he', 'only', 'just', 'yours', 'very', 'into', "was
n't", 'yourself', 'again', 'nor', 'now', 'our', 'there', 'when', 'am', 'on', 'to', 'than', "it's", 'how', 'was', 'and', 'in',
'other', 'weren', 'then', 'herself', 'had', 'more', 'whom', 'be', 'didn', "isn't", 'her', 'the', 'through', 'is', 'against', 'a
fter', 'once', 'y', 'hasn', 'wouldn', 'both', 'their', 'most', "mightn't", 'off', 'too', 'those', 'about', 'who', 'doing', 'bu
t', 'which', 's', 'ain', 'this', 'isn', 'out', 'for', 'here', 'we', 'them', 'until', 'what', 'm', 'a', 'why', 'she', "needn't",
'can', 'down', "aren't", 'should', "doesn't", 'd', 'don', "shouldn't", 'himself', 'further', 'by', 'that', 'mustn', "you'll",
'you', 'below', 'up', 'between', 'were', 'above', 'll', 'at', 'each', 'its', "mustn't", "wouldn't", 'where', "you'd", 'itself',
'these', "weren't", "should've", "hadn't", 'as', 'my', 'his', 'being', 'if', 'such', 'aren', 'no', 'from', "don't", 'did', 'hav
ing', 'not', 'will', 're', 'have', 'been', 'him', 'few', "you're", 'because', 'ours', 'before', "haven't", 'during', 'so', 'al
l', "shan't", "couldn't", 'hadn', 'or', 'me', 'an', 'shouldn', 'hers', 'yourselves', 'it', 'same', 'o', 'doesn', 'won', 'with',
'theirs', "didn't", 'while', 'ourselves', "hasn't", 'of', 'needn', 'do', 'any'}
```

```
[nltk_data]   Unzipping corpora\stopwords.zip.
```

In [13]:
```
from nltk.tokenize import word_tokenize
text1="""Hello Mr.smith,how are you doing today?"""
tokenized_sent=word_tokenize(text1)
print(tokenized_sent)
filtered_sent=[]
for w in tokenized_sent:
    if w not in stop_words:
        filtered_sent.append(w)
print("Tokenized Sentences:",tokenized_sent)
print("Filtered Sentence:",filtered_sent)
```

```
['Hello', 'Mr.smith', ',', 'how', 'are', 'you', 'doing', 'today', '?']
Tokenized Sentences: ['Hello', 'Mr.smith', ',', 'how', 'are', 'you', 'doing', 'today', '?']
Filtered Sentence: ['Hello', 'Mr.smith', ',', 'today', '?']
```

In [14]:
```
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize

ps = PorterStemmer()

stemmed_words=[]
for w in filtered_sent:
    stemmed_words.append(ps.stem(w))

print("Filtered Sentence:",filtered_sent)
print("Stemmed Sentence:",stemmed_words)
```

```
Filtered Sentence: ['Hello', 'Mr.smith', ',', 'today', '?']
Stemmed Sentence: ['hello', 'mr.smith', ',', 'today', '?']
```

In [16]:
```
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()

from nltk.stem.porter import PorterStemmer
stem = PorterStemmer()

word = "flying"
print("Lemmenized word:",lem.lemmatize(word,"v"))
print("Stemmed word:",stem.stem(word))
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\ghans\AppData\Roaming\nltk_data...
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\ghans\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

```
Lemmenized word: fly
Stemmed word: fli
```

In [17]:
```
sent = "Albert Einstein was born in Ulm,Germant in 1879."
```

In [18]:
```
tokens=nltk.word_tokenize(sent)
print(tokens)
```

```
['Albert', 'Einstein', 'was', 'born', 'in', 'Ulm', ',', 'Germant', 'in', '1879', '.']
```

In [19]:
```
nltk.download('averaged_perceptron_tagger')
nltk.pos_tag(tokens)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\ghans\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
```

Out[19]:
```
[('Albert', 'NNP'),
 ('Einstein', 'NNP'),
 ('was', 'VBD'),
 ('born', 'VBN'),
 ('in', 'IN'),
 ('Ulm', 'NNP'),
 (',', ','),
 ('Germant', 'NNP'),
 ('in', 'IN'),
 ('1879', 'CD'),
 ('.', '.')]
```

In [20]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [21]:
```
corpus = [
    "Sachin was the GOAT of the previous generation",
    "Virat is the GOAT of the this generation",
    "Shubman will be the GOAT of the next generation"
]
vectorizer = TfidfVectorizer()
```

In [22]:
```
matrix = vectorizer.fit(corpus)
matrix.vocabulary_
```

Out[22]:
```
{'sachin': 7,
 'was': 12,
 'the': 9,
 'goat': 2,
 'of': 5,
 'previous': 6,
 'generation': 1,
 'virat': 11,
 'is': 3,
 'this': 10,
 'shubman': 8,
 'will': 13,
 'be': 0,
 'next': 4}
```

In [23]:
```
tfidf_matrix = vectorizer.transform(corpus)
print(tfidf_matrix)
```

```
  (0, 12)       0.4286758743128819
  (0, 9)        0.5063657539459899
  (0, 7)        0.4286758743128819
  (0, 6)        0.4286758743128819
  (0, 5)        0.25318287697299496
  (0, 2)        0.25318287697299496
  (0, 1)        0.25318287697299496
  (1, 11)       0.4286758743128819
  (1, 10)       0.4286758743128819
  (1, 9)        0.5063657539459899
  (1, 5)        0.25318287697299496
  (1, 3)        0.4286758743128819
  (1, 2)        0.25318287697299496
  (1, 1)        0.25318287697299496
  (2, 13)       0.39400039808922477
  (2, 9)        0.4654059642457353
  (2, 8)        0.39400039808922477
  (2, 5)        0.23270298212286766
  (2, 4)        0.39400039808922477
  (2, 2)        0.23270298212286766
  (2, 1)        0.23270298212286766
  (2, 0)        0.39400039808922477
```

In [24]:
```
print(vectorizer.get_feature_names_out())
```

```
['be' 'generation' 'goat' 'is' 'next' 'of' 'previous' 'sachin' 'shubman'
 'the' 'this' 'virat' 'was' 'will']
```

In [ ]:

jupyter Practicle8 Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df=pd.read_csv('tiatanic.csv')
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [4]:
```
df.describe()
```

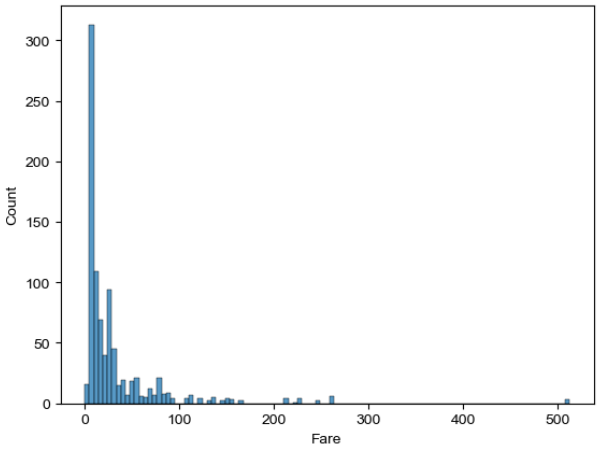
Out[4]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

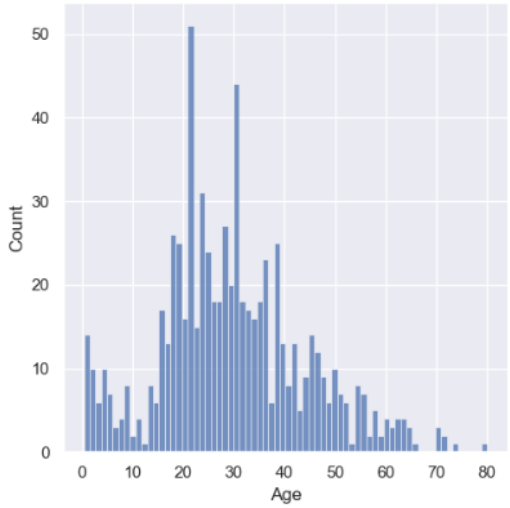
In [5]:
```
df.shape
```

Out[5]:
```
(891, 12)
```

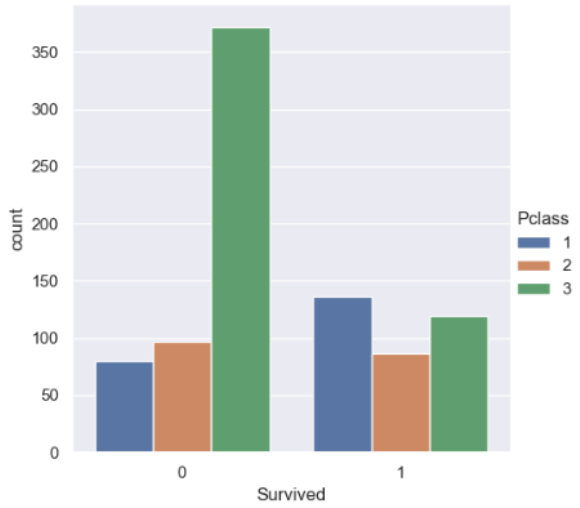
In [6]:
```
sns.histplot(x='Fare',data=df)
sns.set(rc={'figure.figsize':(3,3)})
```



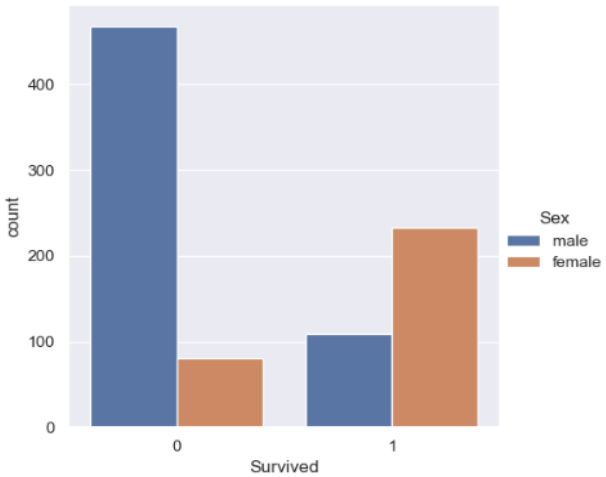
In [7]:
```
sns.displot(x='Age',data=df,bins=70)
sns.set(rc={'figure.figsize':(5,5)})
```



In [8]:
```
sns.catplot(x='Survived', data=df, kind='count', hue='Pclass')
sns.set(rc={'figure.figsize':(5,5)})
```



In [9]:
```
sns.catplot(x='Survived',data=df, kind='count',hue='Sex')
sns.set(rc={'figure.figsize':(5,5)})
```



In [ ]:

jupyter Practicle9 Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
import seaborn as sns
```

In [2]:
```
df = sns.load_dataset('titanic')
df
```

Out[2]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

891 rows × 15 columns

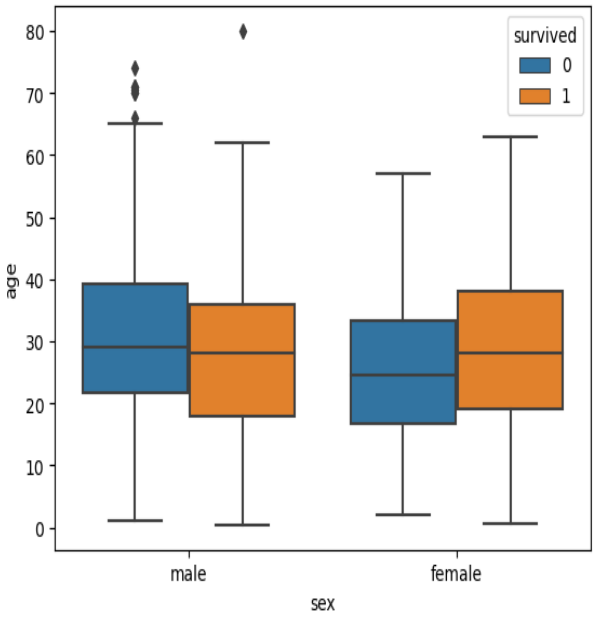
In [3]:
```
df.head()
```

Out[3]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

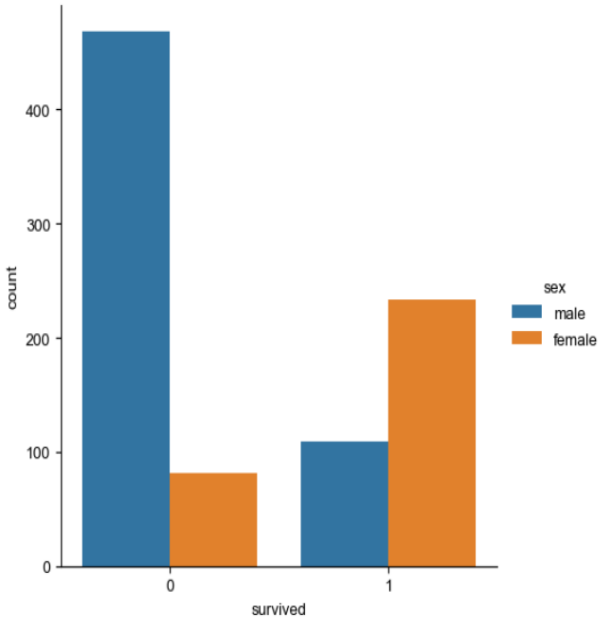
In [6]:
```
sns.boxplot(x='sex', y='age', data=df, hue='survived')
```

Out[6]: <Axes: xlabel='sex', ylabel='age'>



In [9]:
```
sns.catplot(x='survived',data=df, kind='count',hue='sex')
sns.set(rc={'figure.figsize':(5,5)})
```

```
C:\Users\ghans\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```




```
# Observations
We created a box plot of variables 'age & 'sex' & used survival as the hue
Ther we visualized three variables Age, Sex & Survival. Two out of these are categorical and one is numerical
Now in addition to the information about the age of each gender, you can also see the distribution of passengers who survived
For instance, we can see that among the male passengers, on average more younger people survived as compared to older ones
Also, we can see that among the survived passangers, more female survived as complared to male.
```

jupyter **Practical 10** Last Checkpoint: a minute ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help — Trusted — Python 3 (ipykernel)

Run — Code

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [2]:
```
dataset = sns.load_dataset('iris')
```

In [4]:
```
dataset
```

Out[4]:

	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

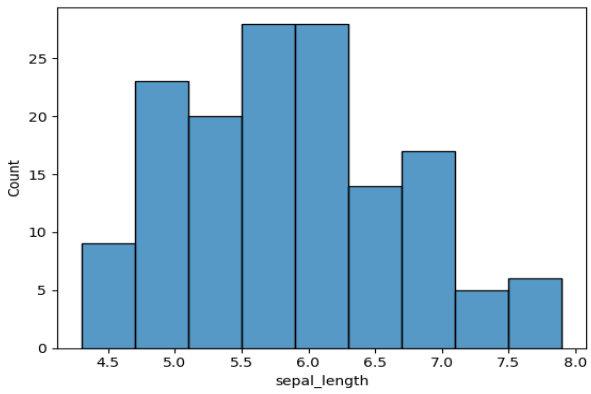
150 rows × 5 columns

In [5]:
```
dataset.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
```

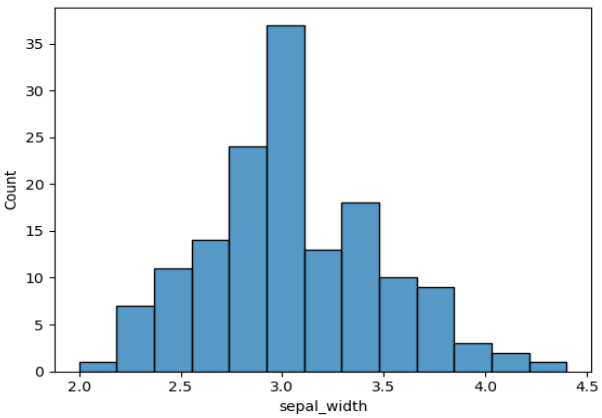
In [8]:
```
sns.histplot(x='sepal_length', data=dataset)
```

Out[8]: <Axes: xlabel='sepal_length', ylabel='Count'>



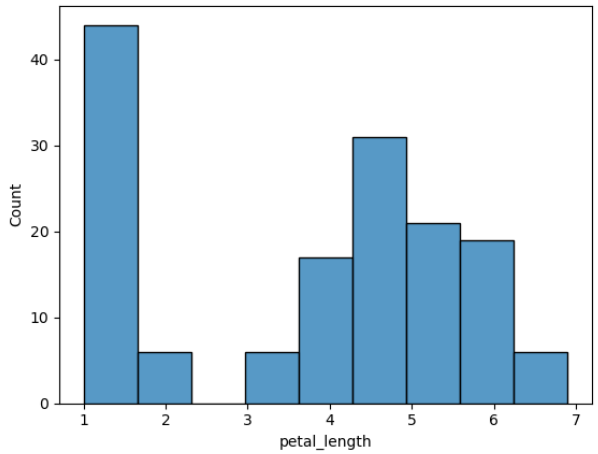
In [9]:
```
sns.histplot(x='sepal_width', data=dataset)
```

Out[9]: <Axes: xlabel='sepal_width', ylabel='Count'>



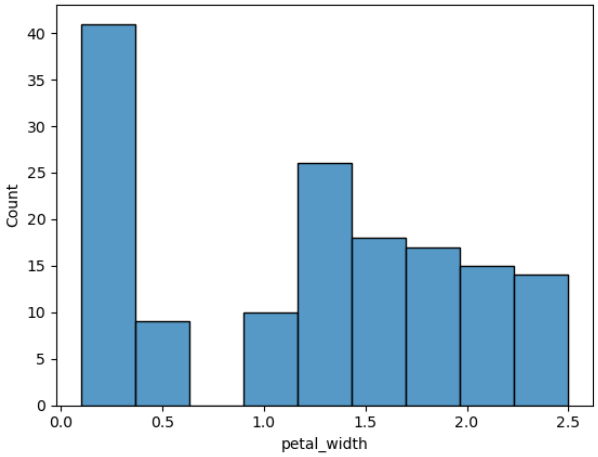
In [10]:
```
sns.histplot(x='petal_length', data=dataset)
```

Out[10]: <Axes: xlabel='petal_length', ylabel='Count'>



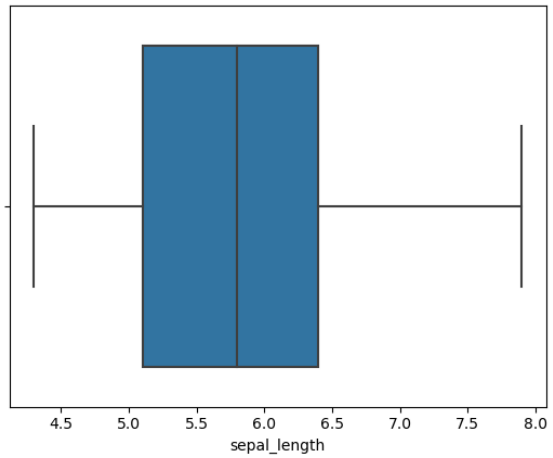
In [11]:
```
sns.histplot(x='petal_width', data=dataset)
```

Out[11]: <Axes: xlabel='petal_width', ylabel='Count'>



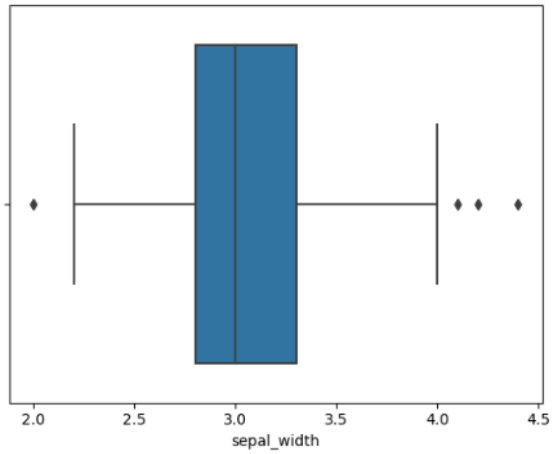
In [12]:
```
sns.boxplot(x='sepal_length', data=dataset)
```

Out[12]: <Axes: xlabel='sepal_length'>



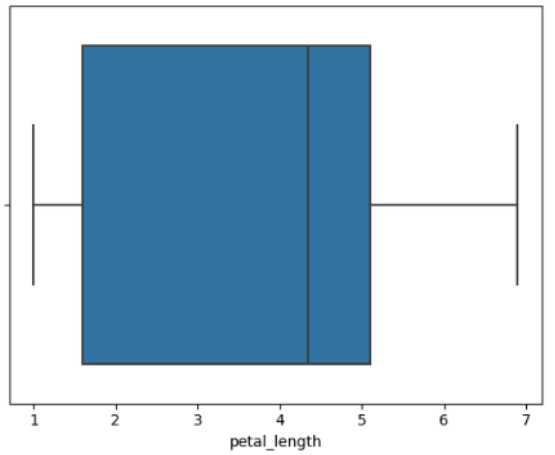
In [14]:
```
sns.boxplot(x='sepal_width', data=dataset)
```

Out[14]: <Axes: xlabel='sepal_width'>



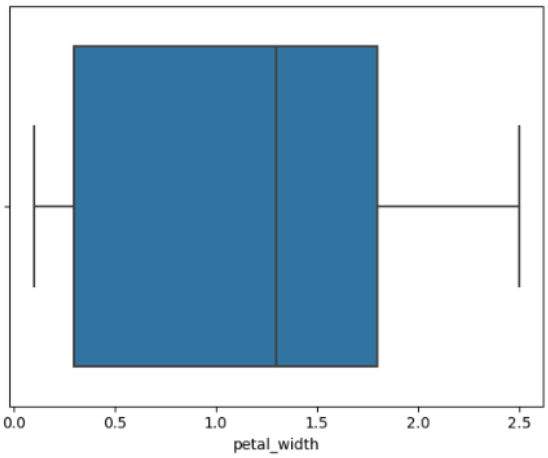
In [15]:
```
sns.boxplot(x='petal_length', data=dataset)
```

Out[15]: <Axes: xlabel='petal_length'>



In [16]:
```
sns.boxplot(x='petal_width', data=dataset)
```

Out[16]: <Axes: xlabel='petal_width'>



In [ ]:
```

```