Data Science Lab

Student PRN:123M1H041

Student Name:DARSHAN SHASHIKANT PATHAK

Submission Date:07/07/2024

Advanced Data Science Lab - Practical Assignment: 3

---

1) Assignment Based on Data Preprocessing and Visualization using Python Librarie

a) Data Exploration Techniques, Handling Missing Values, Encoding Categorial Variables and Data Visualization

1. **Loading and displaying the first 10 rows of a dataset:**

In [1]:
```
import seaborn as sns
import pandas as pd
import numpy as np
import warnings
import os
warnings.filterwarnings("ignore")

iris = sns.load_dataset('iris')
print(iris.head(10))
```

```
   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
5           5.4          3.9           1.7          0.4  setosa
6           4.6          3.4           1.4          0.3  setosa
7           5.0          3.4           1.5          0.2  setosa
8           4.4          2.9           1.4          0.2  setosa
9           4.9          3.1           1.5          0.1  setosa
```

2. **Showing basic statistics (mean, median, standard deviation):**

In [2]:
```
# Basic statistics
basic_stats = iris.describe()
median_values = iris.median(numeric_only=True)
print(basic_stats)
print(median_values)
```

```
       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
sepal_length    5.80
sepal_width     3.00
petal_length    4.35
petal_width     1.30
dtype: float64
```

3. **Handling missing values:**

In [3]:
```
titanic = sns.load_dataset('titanic')
print(titanic.isnull().sum())
```

```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

4. **Filling missing values with median/mode:**

In [4]:
```
titanic['age'].fillna(titanic['age'].median(), inplace=True)
titanic['embarked'].fillna(titanic['embarked'].mode()[0], inplace=True)
```

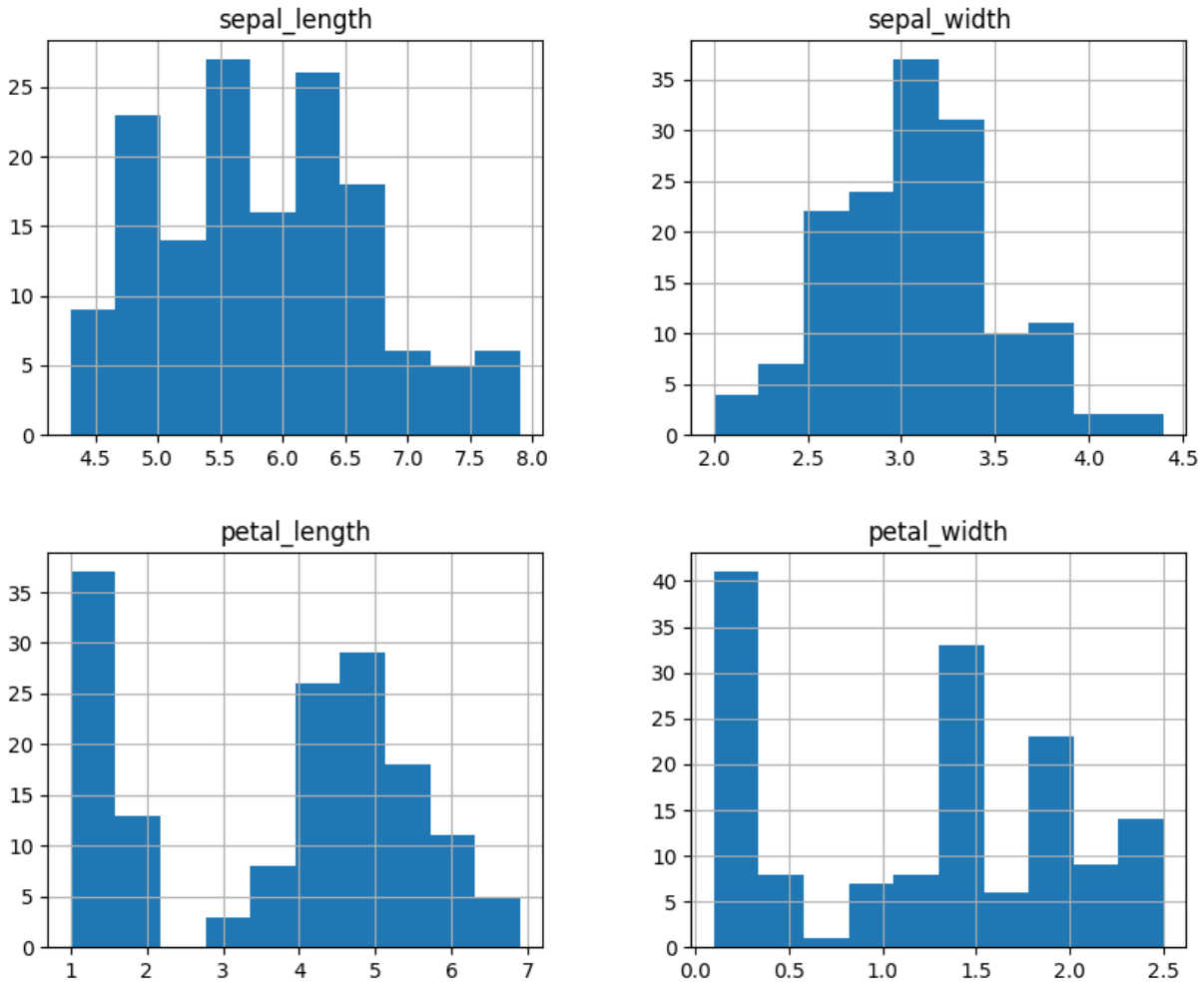
5. **Encoding categorical variables (binary and one-hot):**

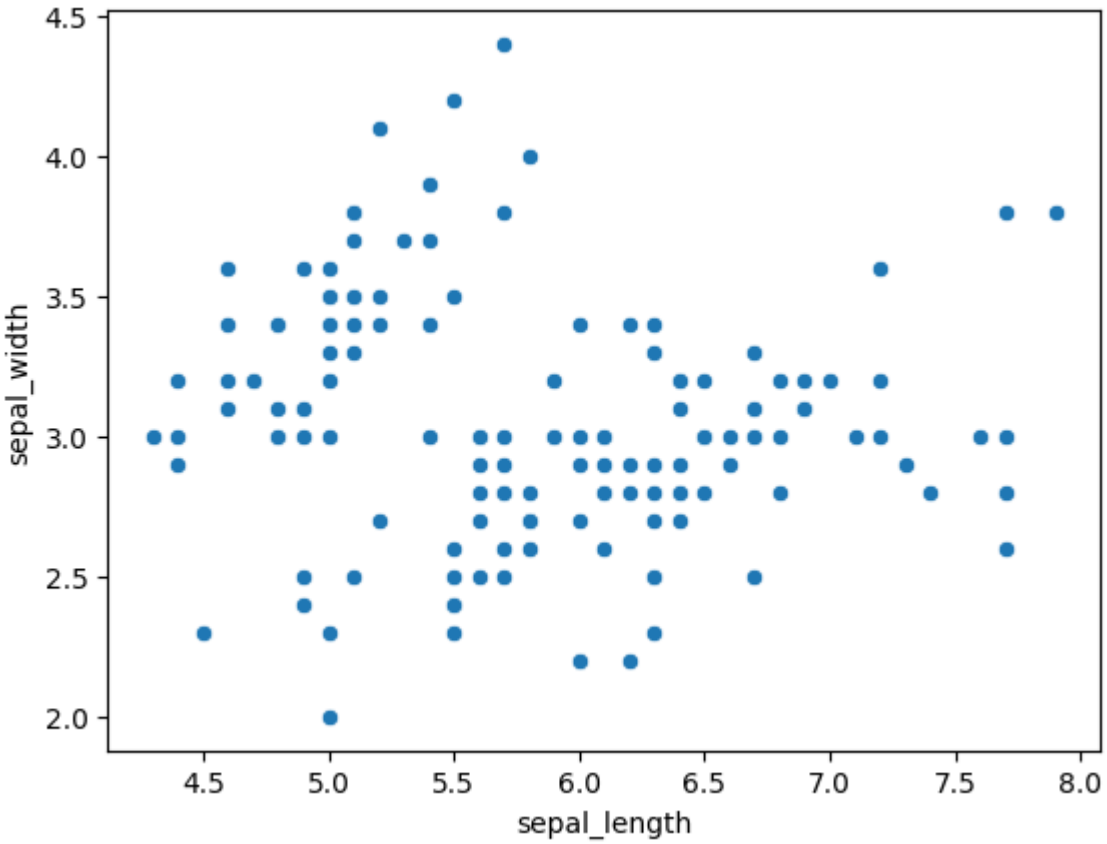
In [5]:
```
# Binary encoding for gender
titanic['gender_binary'] = titanic['sex'].map({'male': 0, 'female': 1})
```

```
# One-hot encoding for embarked column
titanic_encoded = pd.get_dummies(titanic, columns=['embarked'])
```

6. **Plotting histograms and scatter plots:**

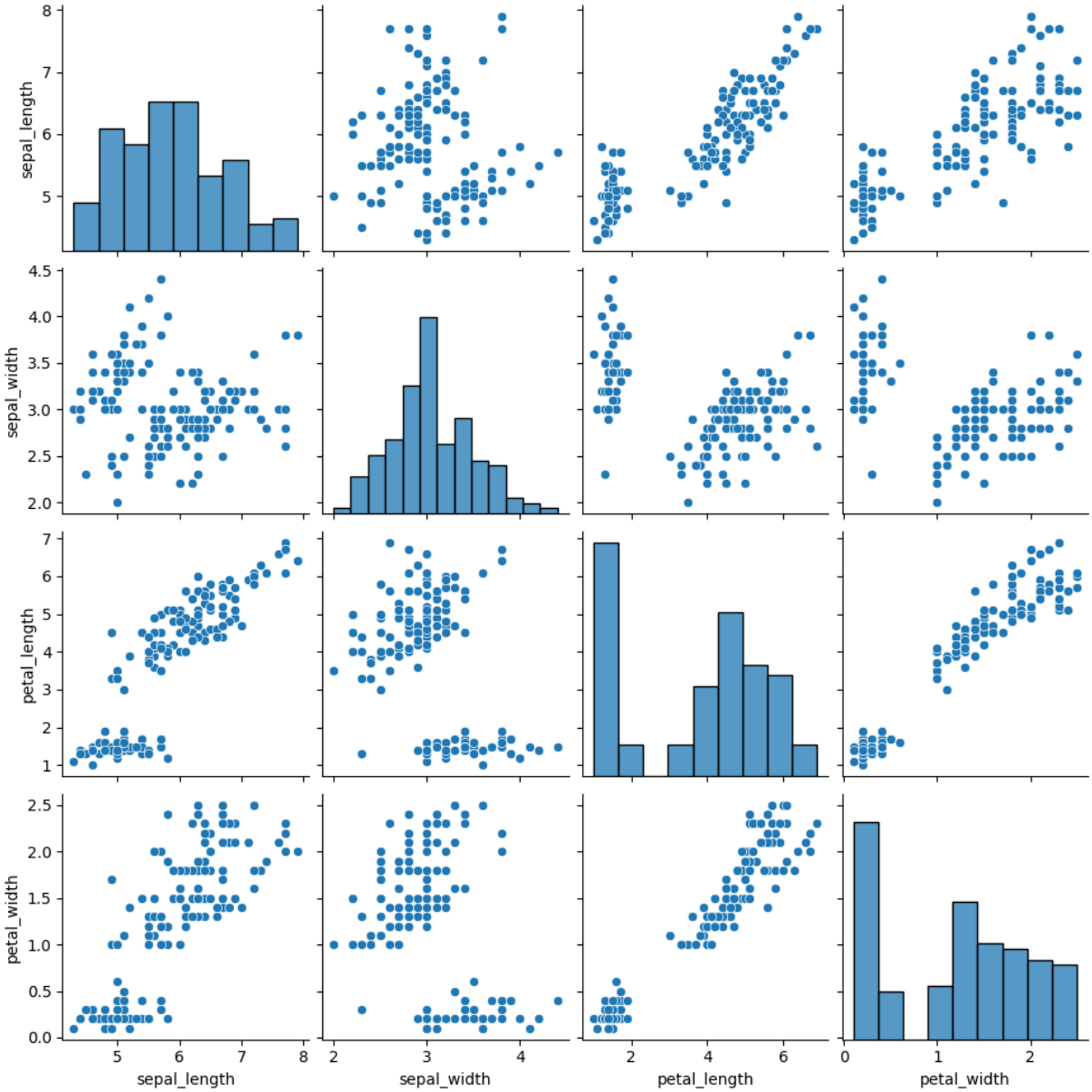
In [6]:
```
import matplotlib.pyplot as plt

# Histograms for each numerical feature
iris.hist(figsize=(10, 8))
plt.show()

# Scatter plot of sepal length vs. sepal width
sns.scatterplot(x='sepal_length', y='sepal_width', data=iris)
plt.show()
```

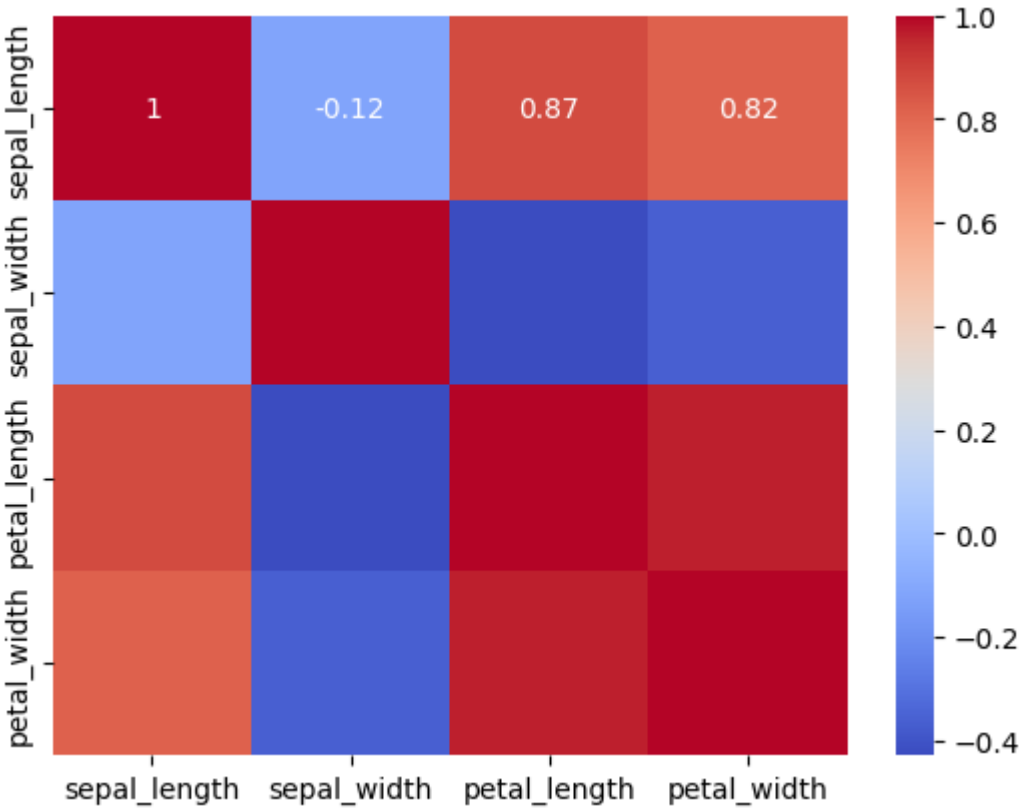




7. **Creating a pair plot and correlation heatmap:**

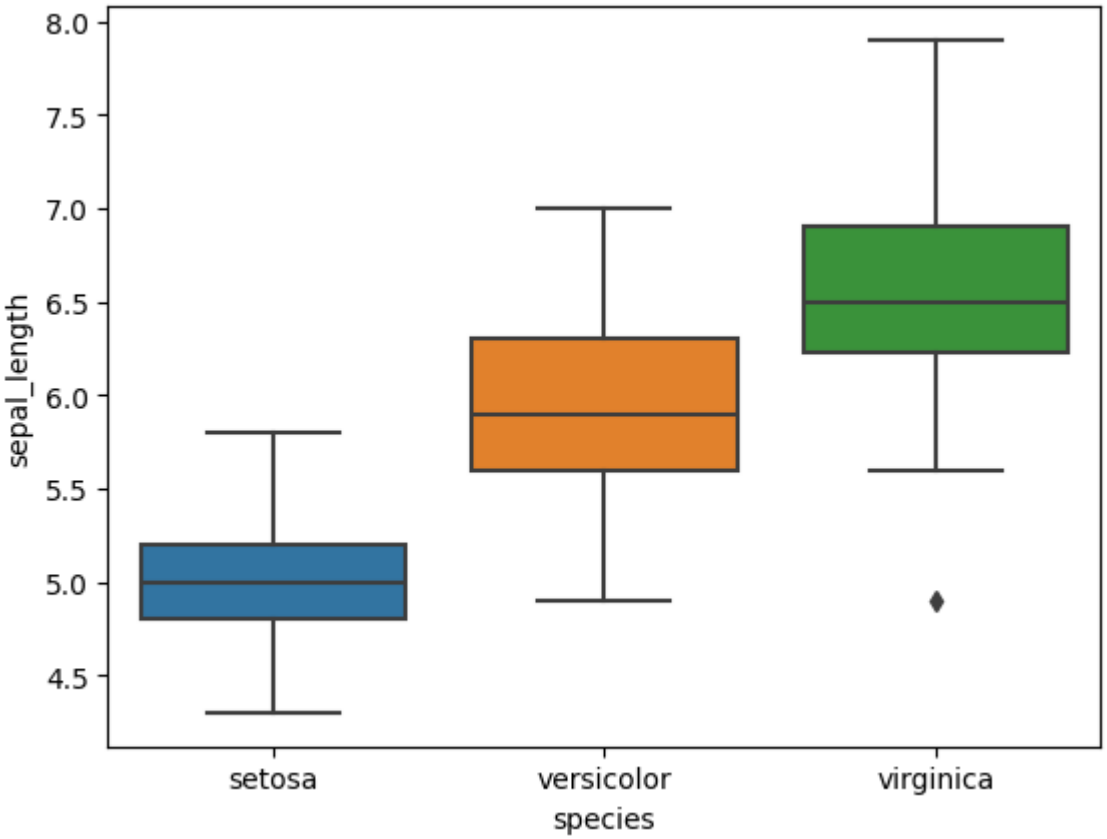
In [7]:
```
sns.pairplot(iris)
plt.show()

sns.heatmap(iris.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.show()
```


sepal_length
sepal_width
petal_length
petal_width
sepal_length
sepal_width
petal_length
petal_width



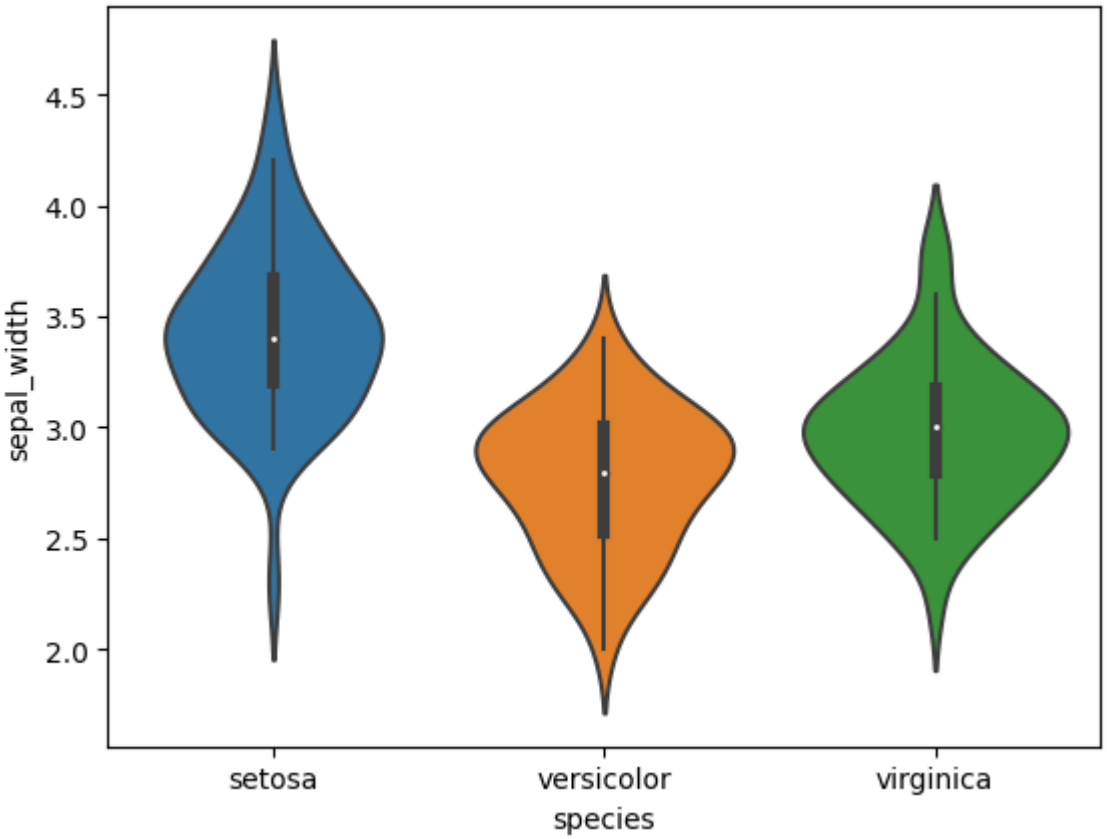
8. **Box plot comparing sepal length across species:**

In [8]:
```
sns.boxplot(x='species', y='sepal_length', data=iris)
plt.show()
```



9. **Violin plot for sepal width across species:**

In [9]:
```
sns.violinplot(x='species', y='sepal_width', data=iris)
plt.show()
```



10. **Grouping data by class and calculating average age:**

```
In [10]: class_age_avg = titanic.groupby('class')['age'].mean()
         print(class_age_avg)
```

```
class
First     36.812130
Second    29.765380
Third     25.932627
Name: age, dtype: float64
```

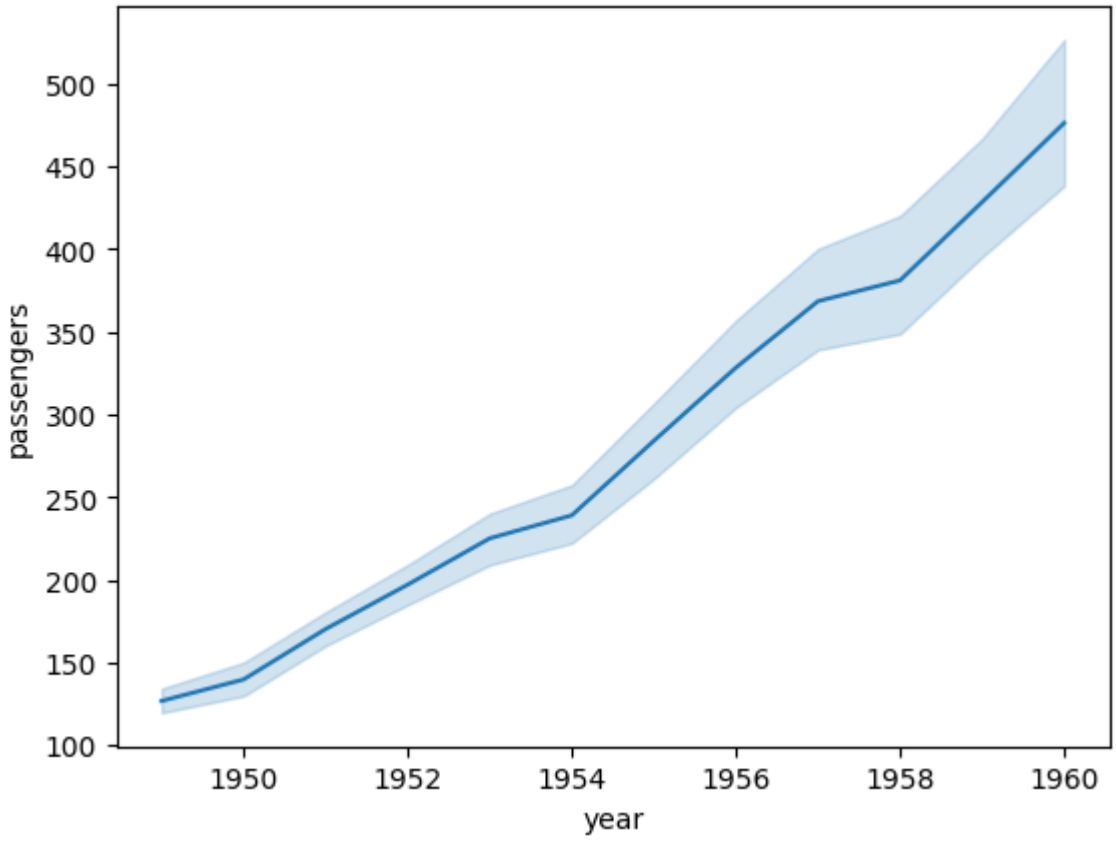
11. **Counting passengers by sex and class:**

```
In [11]: passenger_count = titanic.groupby(['sex', 'class']).size()
         print(passenger_count)
```

```
sex     class
female  First      94
        Second     76
        Third     144
male    First     122
        Second    108
        Third     347
dtype: int64
```

12. **Plotting time series data:**

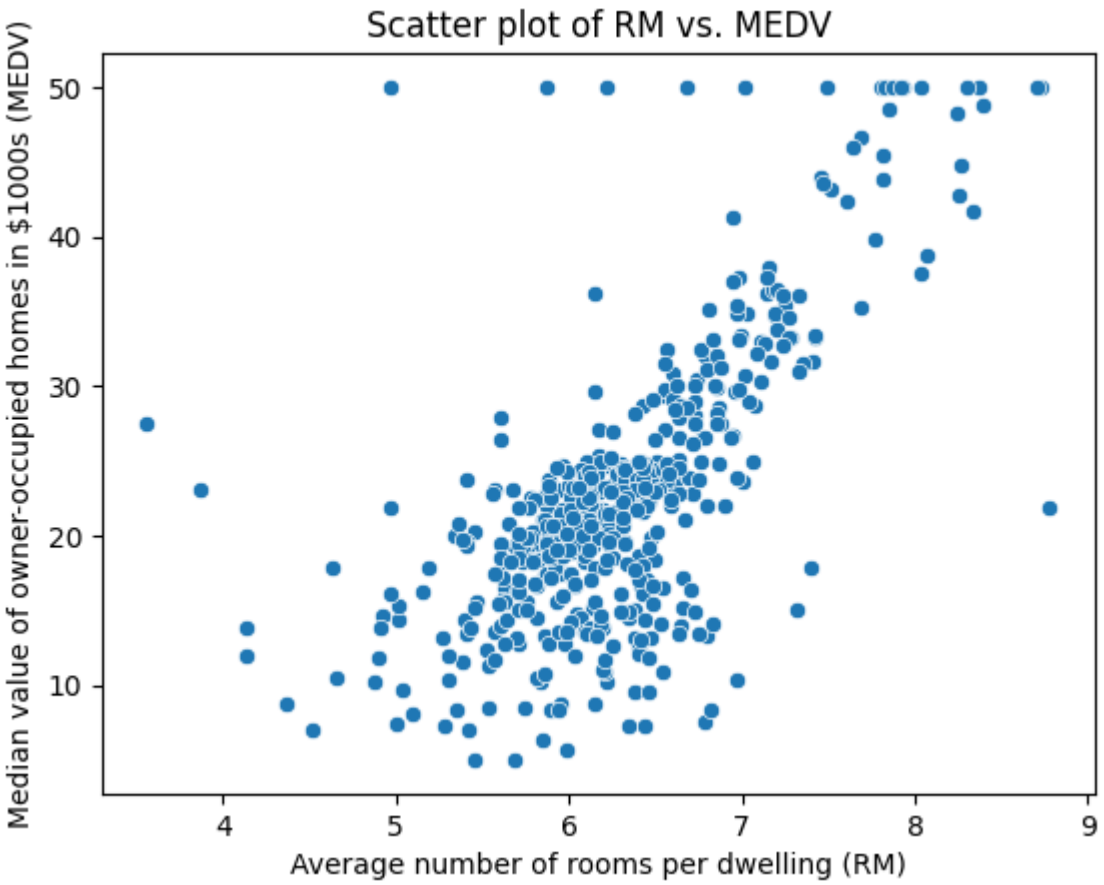
In [12]:
```
flights = sns.load_dataset('flights')

sns.lineplot(x='year', y='passengers', data=flights)
plt.show()
```



13. **Scatter plot for regression:**

In [13]:
```
# Loading the dataset from the data URL
data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)

# Separating the data and target values
data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
target = raw_df.values[1::2, 2]

# Converting to a DataFrame for easier processing
columns = [
    "CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE", "DIS", "RAD", "TAX",
    "PTRATIO", "B", "LSTAT"
]
boston_df = pd.DataFrame(data, columns=columns)
boston_df['MEDV'] = target  # Adding the target variable

# Plotting using Seaborn
sns.scatterplot(x='RM', y='MEDV', data=boston_df)
plt.xlabel("Average number of rooms per dwelling (RM)")
plt.ylabel("Median value of owner-occupied homes in $1000s (MEDV)")
```

```
plt.title("Scatter plot of RM vs. MEDV")
plt.show()
```



b) Assignment Based On Feature Engineering and Exploratory Data Analysis (EDA):

1. **Loading the Titanic dataset:**

In [14]:
```
import seaborn as sns

titanic = sns.load_dataset('titanic')
```

2. **Creating a new feature 'family_size' by combining 'sibsp' and 'parch':**

In [15]:
```
titanic['family_size'] = titanic['sibsp'] + titanic['parch']
```

3. **Creating a feature 'is_alone' which indicates whether the passenger is traveling alone:**

In [16]:
```
titanic['family_size'] = titanic['sibsp'] + titanic['parch']
```

4. **Applying Min-Max Scaling to normalize numerical features (Use Iris dataset):**

In [17]:
```
from sklearn.preprocessing import MinMaxScaler

iris = sns.load_dataset('iris')

scaler = MinMaxScaler()
iris_scaled = scaler.fit_transform(iris.drop('species', axis=1))
iris_scaled_df = pd.DataFrame(iris_scaled, columns=iris.columns[:-1])
```

5. **Applying Standardization to numerical features (Use Iris dataset):**

In [18]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
iris_standardized = scaler.fit_transform(iris.drop('species', axis=1))
iris_standardized_df = pd.DataFrame(iris_standardized, columns=iris.columns[:-1])
```

6. **Detecting outliers in the Boston housing dataset using IQR and removing them:**

In [19]:
```
Q1 = boston_df.quantile(0.25)
Q3 = boston_df.quantile(0.75)
IQR = Q3 - Q1
boston_no_outliers = boston_df[~((boston_df < (Q1 - 1.5 * IQR)) | (boston_df > (Q3
```

7. **Using correlation analysis to select features and Recursive Feature Elimination (RFE) in Iris dataset:**

In [20]:
```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

correlation_matrix = iris.corr(numeric_only = True)

X = iris.drop('species', axis=1)
y = iris['species']

model = LogisticRegression(max_iter=200)
rfe = RFE(model, n_features_to_select=2)
fit = rfe.fit(X, y)
print(fit.support_)
```

```
[False False  True  True]
```

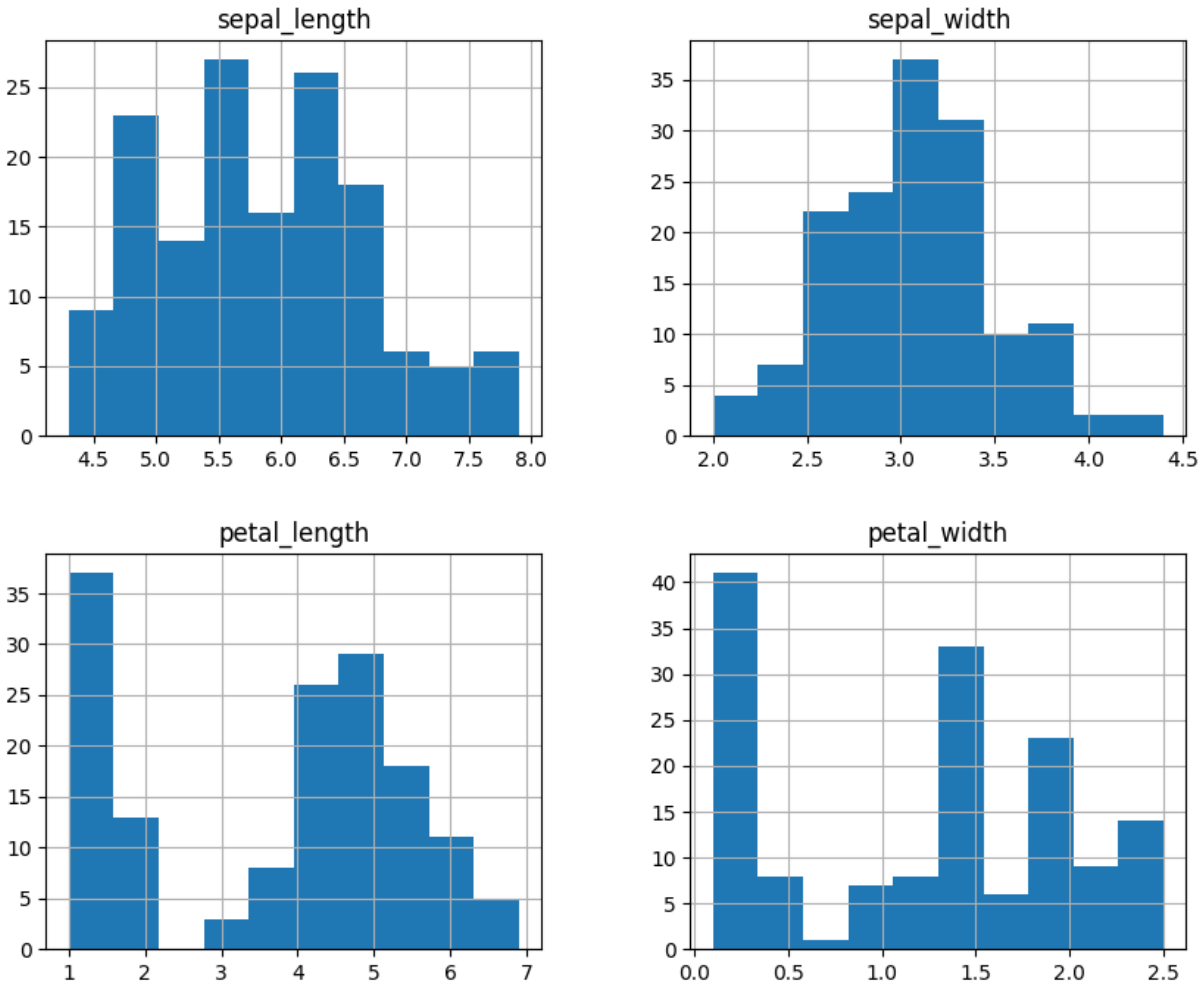
8. **Creating a new feature 'is_weekend' using a dataset with date/time:**

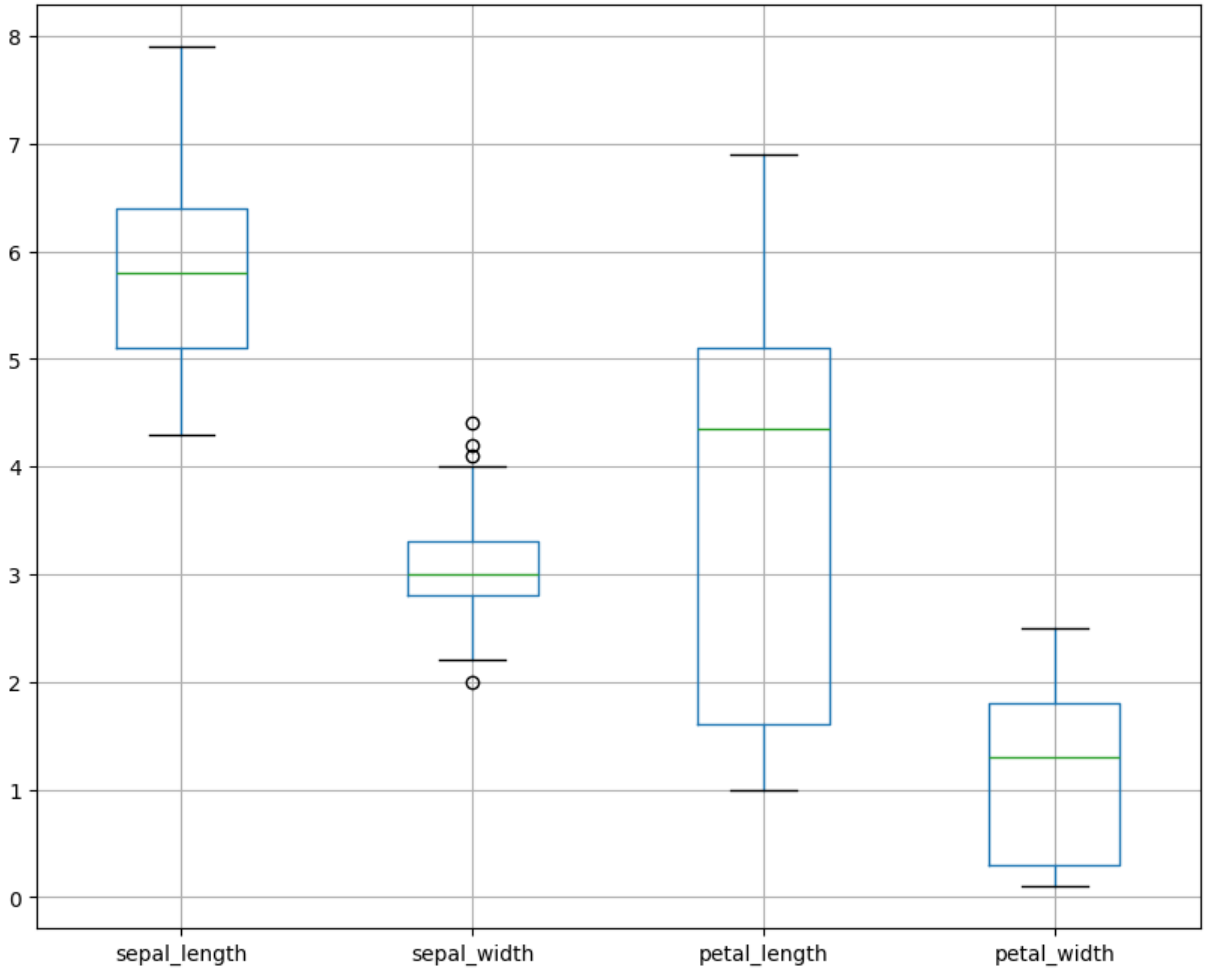
In [21]:
```
import pandas as pd

flight_data = pd.DataFrame({
    'date': pd.date_range(start='2022-01-01', periods=10, freq='D')
})

flight_data['year'] = flight_data['date'].dt.year
```

```
flight_data['month'] = flight_data['date'].dt.month
flight_data['day'] = flight_data['date'].dt.day
flight_data['day_of_week'] = flight_data['date'].dt.dayofweek
flight_data['is_weekend'] = flight_data['day_of_week'].apply(lambda x: 1 if x >= 5
```

9. **Plotting histograms and box plots for each numerical feature in Iris dataset:**

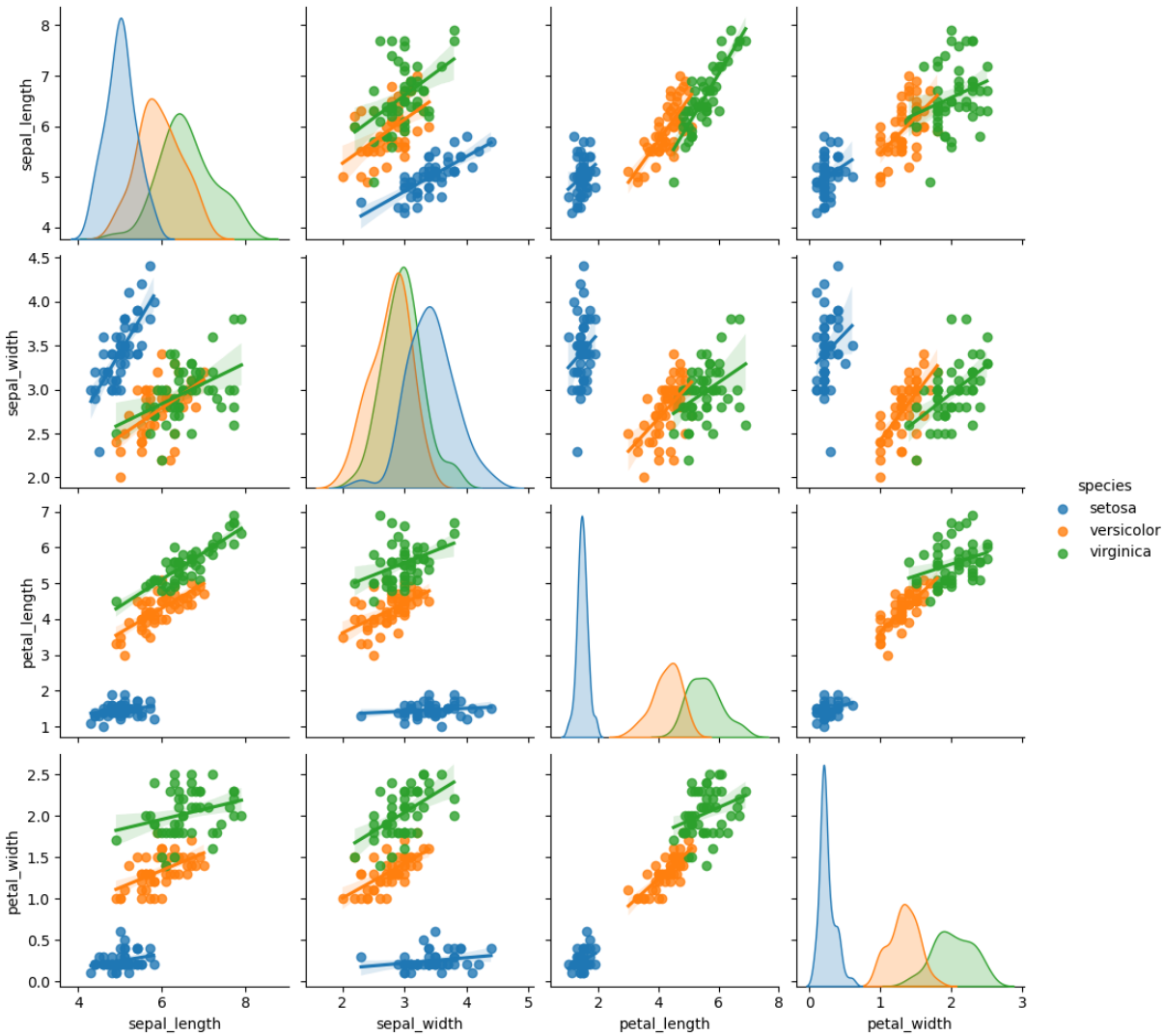
In [22]:
```
import matplotlib.pyplot as plt

iris.hist(figsize=(10, 8))
plt.show()

iris.boxplot(figsize=(10, 8))
plt.show()
```





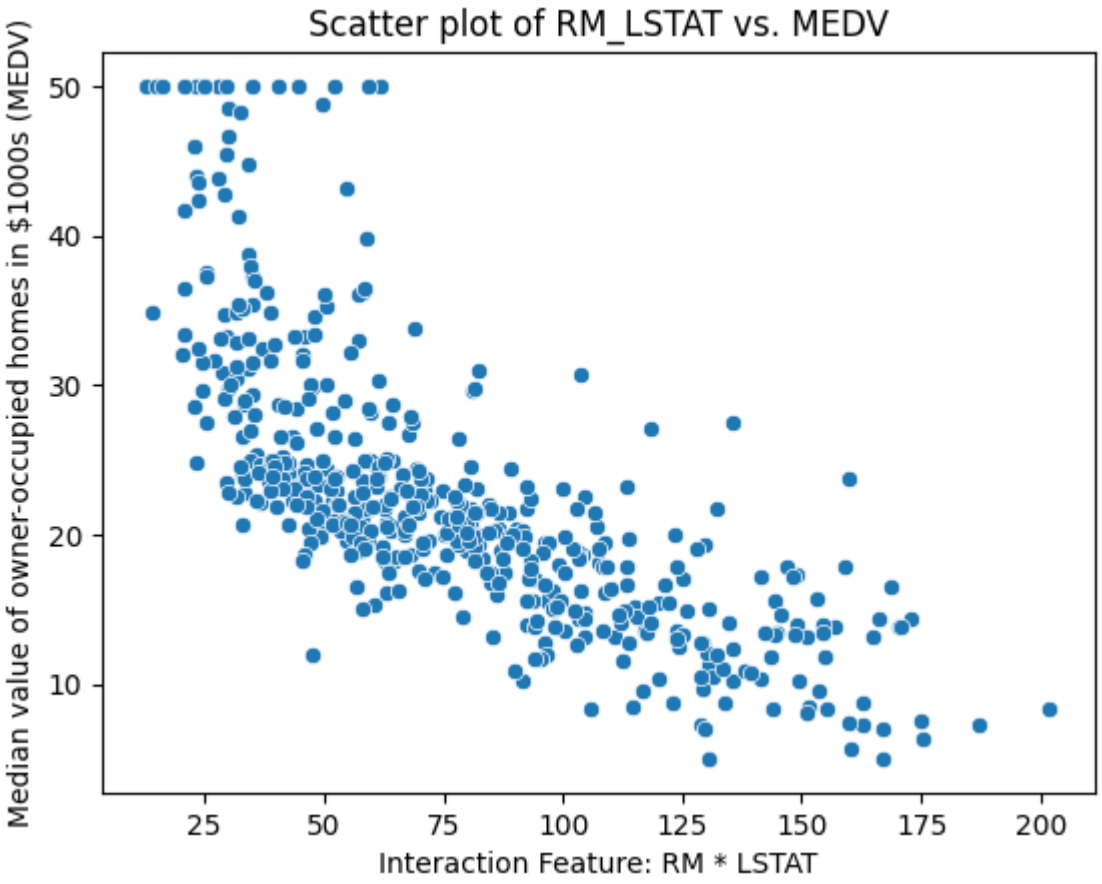
10. **Creating pair plots with regression lines for selected feature pairs:**

In [23]:
```
sns.pairplot(iris, kind="reg", hue='species')
plt.show()
```



11. **Creating interaction features (e.g., product of two features) and visualizing their effects:**

In [24]:
```
# Creating a new interaction feature
boston_df['RM_LSTAT'] = boston_df['RM'] * boston_df['LSTAT']

# Plotting the interaction feature against the target (MEDV)
sns.scatterplot(x='RM_LSTAT', y='MEDV', data=boston_df)
plt.xlabel("Interaction Feature: RM * LSTAT")
plt.ylabel("Median value of owner-occupied homes in $1000s (MEDV)")
plt.title("Scatter plot of RM_LSTAT vs. MEDV")
plt.show()
```



12. **Feature engineering: handling missing values, encoding categorical variables, and creating new features in Titanic dataset:**

In [25]:
```
titanic['age'].fillna(titanic['age'].median(), inplace=True)
titanic['embarked'].fillna(titanic['embarked'].mode()[0], inplace=True)

titanic_encoded = pd.get_dummies(titanic, columns=['embarked', 'sex'])

titanic_encoded['family_size'] = titanic_encoded['sibsp'] + titanic_encoded['parch'
```

13. **Preparing a dataset for model building by splitting it into training and testing sets:**

In [26]:
```
from sklearn.model_selection import train_test_split

X = titanic_encoded.drop(['survived'], axis=1)
y = titanic_encoded['survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```
