Task 10 : Binary Classification of Iris species with SVM

In [1]:
```python
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

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

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

Out[3]:

	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 [4]:
```python
df.shape
```

Out[4]: (150, 6)

In [5]:
```python
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 [6]:
```python
print(df.isna().sum())
```

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [7]:
```python
df.describe()
```

Out[7]:

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

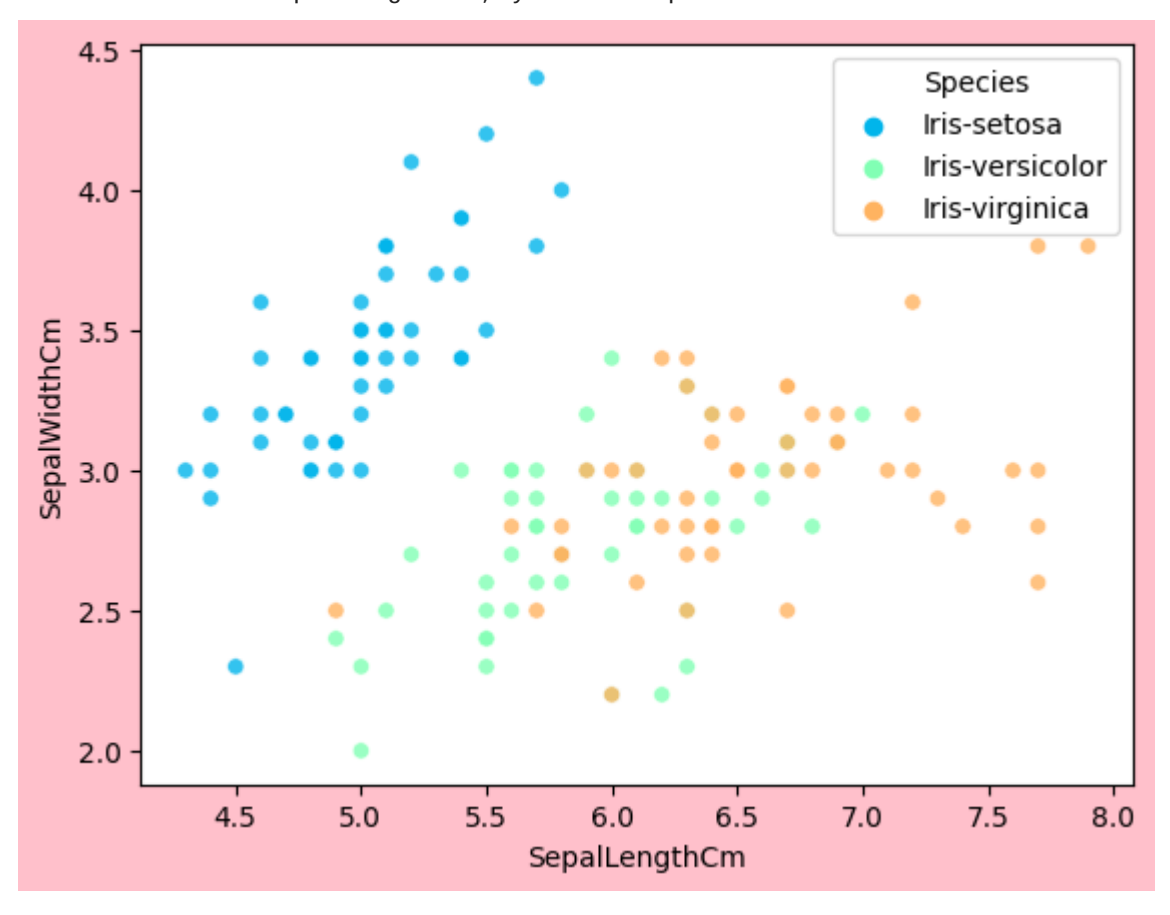
In [8]:
```python
print(df.isna().sum())
```

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [9]:
```python
plt.figure(facecolor='Pink')

sns.scatterplot(data=df,x='SepalLengthCm',y='SepalWidthCm',hue='Species',alpha=0.8,palette='rainbow')
```

Out[9]: <Axes: xlabel='SepalLengthCm', ylabel='SepalWidthCm'>



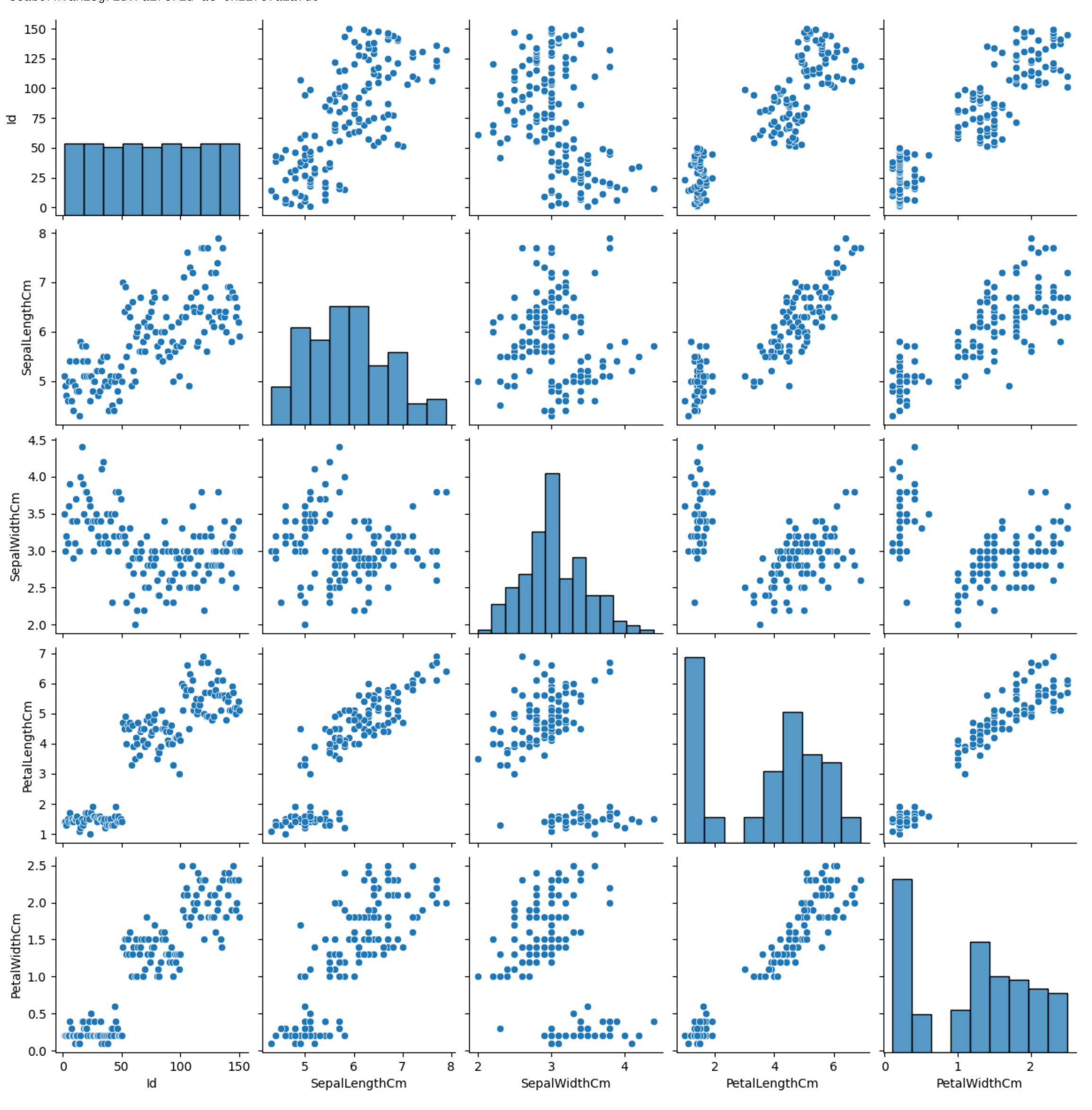
In [10]:
```python
df.corr()
```

Out[10]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Id	1.000000	0.716676	-0.397729	0.882747	0.899759
SepalLengthCm	0.716676	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.397729	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.882747	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.899759	0.817954	-0.356544	0.962757	1.000000

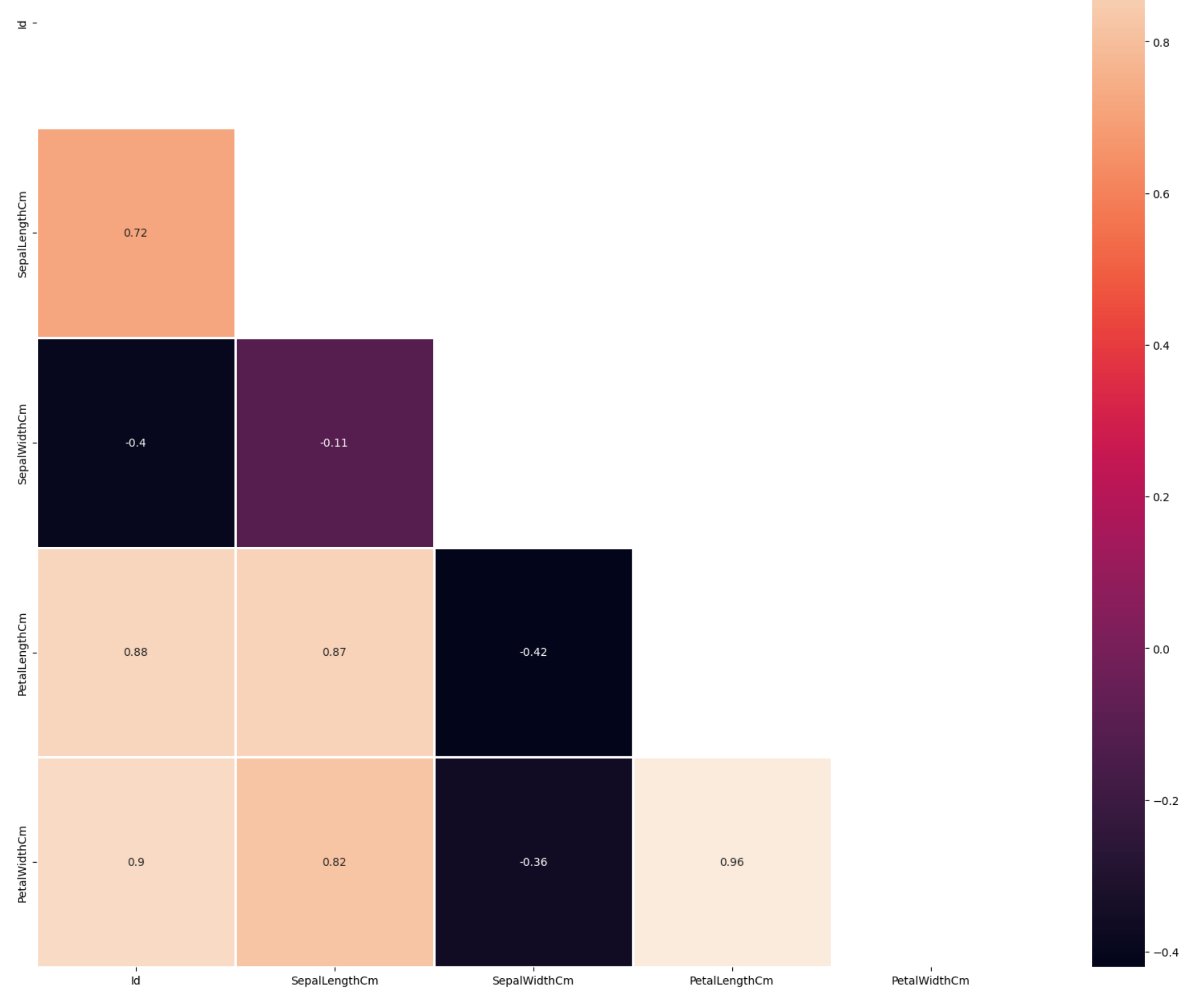
In [11]:
```python
sns.pairplot(df)
```

Out[11]: <seaborn.axisgrid.PairGrid at 0x21fe7a1afd0>



In [12]:
```python
df_corr=df.corr()
```

In [13]:
```python
plt.figure(figsize=(20,17))
matrix=np.triu(df_corr)
sns.heatmap(df_corr, annot=True, linewidth=.8, mask=matrix, cmap='rocket');
plt.show()
```



In [14]:
```python
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Species'] = le.fit_transform(df['Species'])
df.head()
```

Out[14]:

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

In [15]:
```python
x = df.drop(columns='Species',axis=1)
y = df['Species']
```

In [16]:
```python
from sklearn.model_selection import train_test_split
x_train , x_test , y_train , y_test = train_test_split(x,y , test_size=0.25 , random_state=0)
```

In [17]:
```python
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
sc.fit(x_train)
x_train=sc.transform(x_train)
x_test=sc.transform(x_test)
```

In [18]:
```python
print(x_train.shape,x_test.shape, y_train.shape,y_test.shape)
```

```
(112, 5) (38, 5) (112,) (38,)
```

In [19]:
```python
from sklearn.metrics import r2_score,accuracy_score
```

In [20]:
```python
y_test
```

Out[20]:
```
114    2
62     1
33     0
107    2
7      0
100    2
40     0
86     1
76     1
71     1
134    2
51     1
73     1
54     1
63     1
37     0
78     1
90     1
45     0
16     0
121    2
66     1
24     0
8      0
126    2
22     0
44     0
97     1
93     1
26     0
137    2
84     1
27     0
127    2
132    2
59     1
18     0
83     1
Name: Species, dtype: int32
```

In [21]:
```python
from sklearn.svm import SVC
```

In [22]:
```python
from sklearn.metrics import accuracy_score,r2_score
```

In [23]:
```python
model=SVC()
```

In [24]:
```python
model.fit(x_train,y_train)
```

Out[24]:
```
▾ SVC
SVC()
```

In [25]:
```python
y_pred=model.predict(x_test)
```

In [26]:
```python
y_pred
```

Out[26]:
```
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])
```

In [27]:
```python
y_test
```

Out[27]:
```
114    2
62     1
33     0
107    2
7      0
100    2
40     0
86     1
76     1
71     1
134    2
51     1
73     1
54     1
63     1
37     0
78     1
90     1
45     0
16     0
121    2
66     1
24     0
8      0
126    2
22     0
44     0
97     1
93     1
26     0
137    2
84     1
27     0
127    2
132    2
59     1
18     0
83     1
Name: Species, dtype: int32
```

In [28]:
```python
model.score(x_test,y_test)
```

Out[28]: 1.0

In [30]:
```python
print(accuracy_score(y_pred,y_test))
```

```
1.0
```

In [31]:
```python
acc=accuracy_score(y_test,y_pred)
acc
```

Out[31]: 1.0

In [32]:
```python
from sklearn.neighbors import KNeighborsClassifier
```

In [33]:
```python
neigh=KNeighborsClassifier(n_neighbors=3)
```

In [34]:
```python
neigh.fit(x_train,y_train)
```

Out[34]:
```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

In [52]:
```python
y_pred=neigh.predict(x_test)
```

In [53]:
```python
neigh.score(x_test,y_test)
```

Out[53]: 1.0

In [54]:
```python
acc=accuracy_score(y_test,y_pred)
acc
```

Out[54]: 1.0

In [55]:
```python
print(accuracy_score(y_test,y_pred))
```

```
1.0
```

In [56]:
```python
r2_score(y_test,y_pred)
```

Out[56]: 1.0

In [57]:
```python
import xgboost as xgb
```

In [58]:
```python
mod = xgb.XGBRegressor()
```

In [59]:
```python
mod.fit(x_train,y_train)
```

Out[59]:
```
▾ XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
```

In [43]:
```python
r2_score(y_test,mod.predict(x_test))
```

Out[43]: 0.9999999864080886

In [44]:
```python
def knn_func(train_data,label_data,test_data,k) :
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(train_data,label_data)
    pred_label = knn.predict(test_data)
    return pred_label
```

In [45]:
```python
n=len(df)
n
```

Out[45]: 150

In [46]:
```python
import math
```

In [47]:
```python
math.sqrt(n)
```

Out[47]: 12.24744871391589

In [48]:
```python
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
cm
```

Out[48]:
```
array([[13,  0,  0],
       [ 0, 16,  0],
       [ 0,  0,  9]], dtype=int64)
```

Documentation

1. Title and Introduction:

Title: "Iris Species Classification using Support Vector Machines"

Introduction: The goalof the task is to build a model that can accurately classify iris flowers into their respective species based on features like sepal length, sepal width, petal length, and petal width. Emphasize the importance of this task in the context of botanical research and plant species identification.

2. Purpose of the Task:

Clearly state that the purpose is to develop a machine learning model capable of classifying iris flowers into setosa, versicolor, and virginica species based on their morphological features.

2. Project Objectives:

1. Develop a robust machine learning model for iris species classification.
2. Evaluate the model's performance using appropriate metrics.
3. Provide a user-friendly interface for predicting iris species.

3. Dataset:

Source: The Iris dataset is a classic dataset often used for classification tasks, and it's available through various machine learning libraries like scikit-learn. Features: The dataset contains four features: sepal length, sepal width, petal length, and petal width. Size: Each species has 50 samples, making it a balanced dataset. Preprocessing: Check for any missing values or outliers in the data.

4. SVM Kernel Chosen:

The chosen model for this project is a Support Vector Machine (SVM), a popular algorithm for classification tasks.

5. Model Training:

Describe the training process for the SVM model:

1. Split the dataset into training and testing sets.
2. Train the SVM model using the training data.
3. Fine-tune hyperparameters for optimal performance.

6. Evaluation Results:

Present the evaluation results for the SVM model:

Metrics: Evaluate the model's performance using metrics such as accuracy, precision, recall, and F1 score. Confusion Matrix: Use confusion matrix to analyze classification results.

7.Discussion:

Interpret the results:

Discuss the model's overall performance and its ability to correctly classify each iris species.

Address any challenges or limitations encountered during the training and evaluation process.

8. Conclusion:

1. Summarize the key findings and highlight the success of the SVM model in classifying iris species based on the chosen features.
2. Summarize the model's performance and any insights gained from the analysis.

In [ ]: