#BEGINNER LEVEL TASK-01

Iris Flowers Classification ML Project :

In [17]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="white", color_codes=True)
```

In [51]:

```
iris = pd.read_csv("C:/Users/Dell/Desktop/Iris.csv")  #"C:\Users\Dell\Desktop\Iris.csv"
```

In [52]:

```
iris.head()
```

Out[52]:

	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 [53]:

```
iris.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 [54]:

```
iris.shape
```

Out[54]:

```
(150, 6)
```

In [55]:

```
iris.describe()
```

Out[55]:

	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 [58]:

```
#returns count of each data value in a column
iris['SepalLengthCm'].value_counts()
```

Out[58]:

```
5.0    10
5.1     9
6.3     9
5.7     8
6.7     8
5.8     7
5.5     7
6.4     7
4.9     6
5.4     6
6.1     6
6.0     6
5.6     6
4.8     5
6.5     5
6.2     4
7.7     4
6.9     4
4.6     4
5.2     4
5.9     3
4.4     3
7.2     3
6.8     3
6.6     2
4.7     2
7.6     1
7.4     1
7.3     1
7.0     1
7.1     1
5.3     1
4.3     1
4.5     1
7.9     1
Name: SepalLengthCm, dtype: int64
```

In [59]:

```
iris['SepalWidthCm'].value_counts()
```

Out[59]:

```
3.0    26
2.8    14
3.2    13
3.1    12
3.4    12
2.9    10
2.7     9
2.5     8
3.5     6
3.3     6
3.8     6
2.6     5
2.3     4
3.7     3
2.4     3
2.2     3
3.6     3
3.9     2
4.4     1
4.0     1
4.1     1
4.2     1
2.0     1
Name: SepalWidthCm, dtype: int64
```

In [60]:

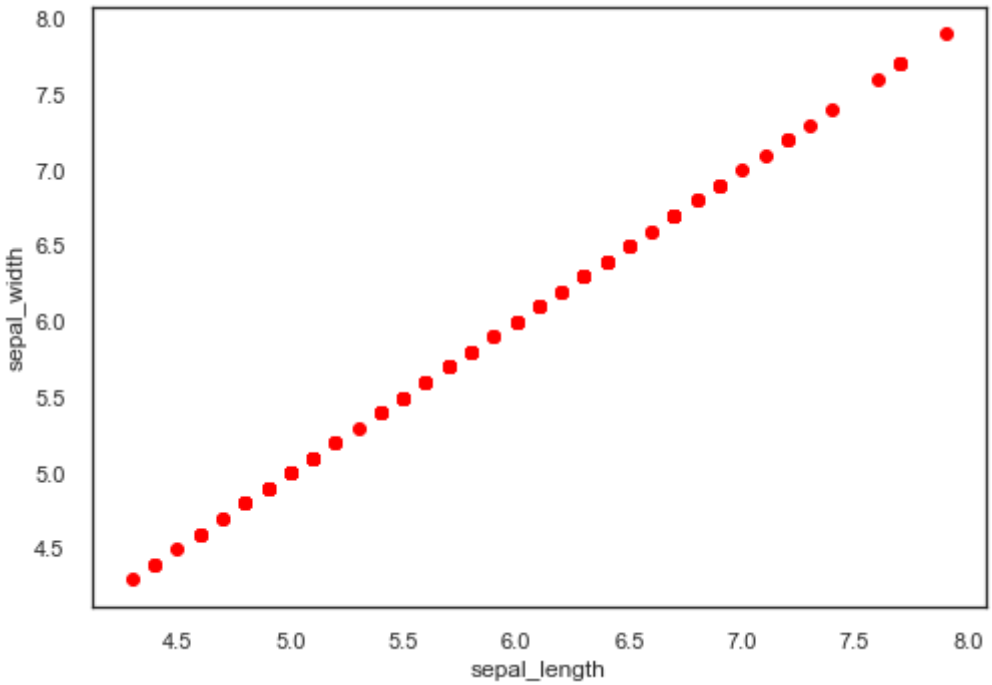
```
#gives the count of missing / null values for each column
iris.isnull().sum()
```

Out[60]:

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

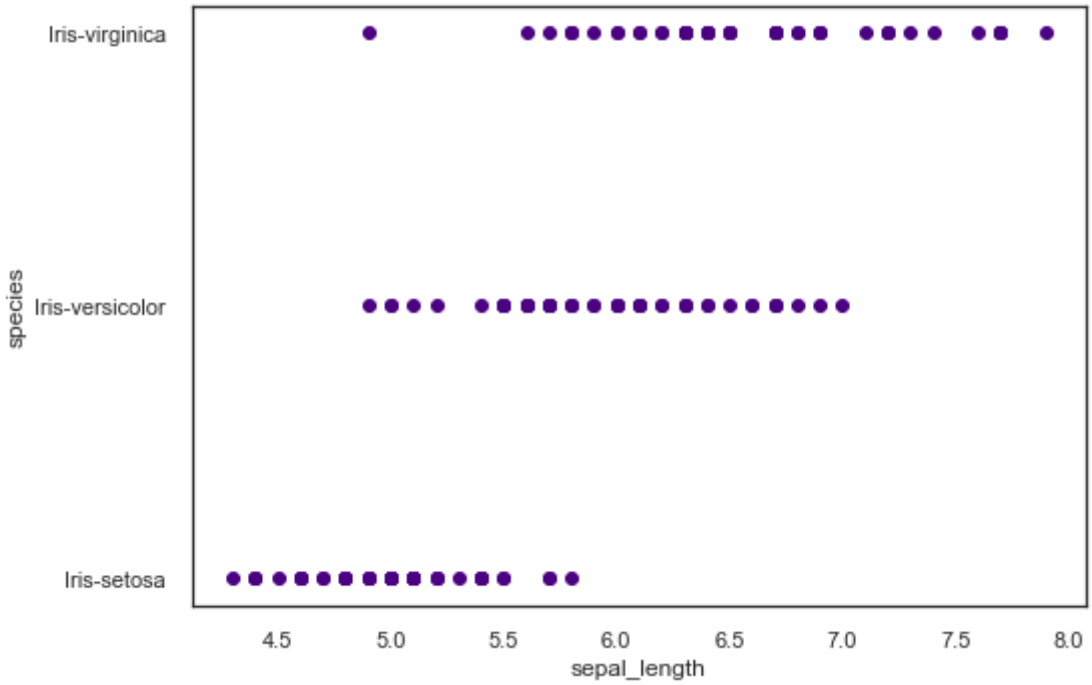
In [63]:

```
#plotting various scatter plots to see how values are arranged
plt.scatter(iris['SepalLengthCm'] , iris['SepalLengthCm'] , color = 'red')
plt.xlabel('sepal_length')
plt.ylabel('sepal_width')
plt.show()
```



In [45]:

```
plt.scatter(iris['SepalLengthCm'] , iris['species'] , color = 'indigo')
plt.xlabel('SepalLengthCm')
plt.ylabel('species')
plt.show()
```



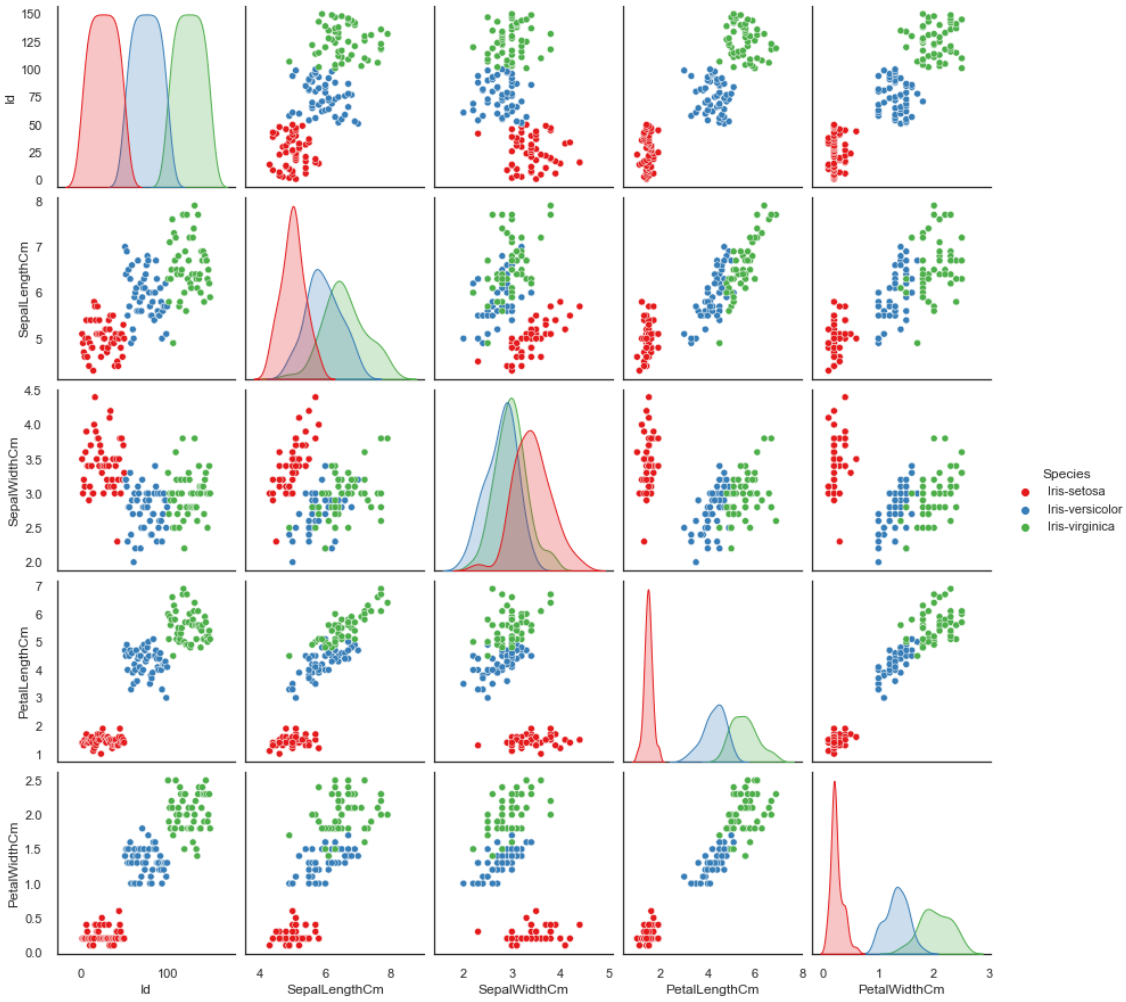
SEPARATE FEATURES AND LABELED DATA

In [46]:

```
x = iris.iloc[: , :-1]
y = iris.iloc[: , -1]
```

In [65]:

```
sns.pairplot(iris, hue='Species',palette='Set1');
```



We used all the features of iris in above models. Now we will use Petals and Sepals Seperately

Creating Petals And Sepals Training Data

In [66]:

```
petal=iris[['PetalLengthCm','PetalWidthCm','Species']]
sepal=iris[['SepalLengthCm','SepalWidthCm','Species']]
```

In [71]:

```
from sklearn import metrics #for checking the model accuracy
```

In [73]:

```
# importing alll the necessary packages to use the various classification algorithms
X = iris.drop('Species', axis=1)
y = iris['Species']

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 [76]:

```
# Support Vector Machine (SVM)
#for Support Vector Machine (SVM) Algorithm
# Training
from sklearn import svm
classifier1 = svm.SVC()
classifier1.fit(X_train,y_train)
y_pred1 = classifier1.predict(X_test)

# Evaluating the Algorithm
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(confusion_matrix(y_test, y_pred1))
print(classification_report(y_test, y_pred1))
print(accuracy_score(y_test,y_pred1))
```

```
[[13  0  0]
 [ 0 16  0]
 [ 0  0  9]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        13
Iris-versicolor       1.00      1.00      1.00        16
 Iris-virginica       1.00      1.00      1.00         9

       accuracy                           1.00        38
      macro avg       1.00      1.00      1.00        38
   weighted avg       1.00      1.00      1.00        38

1.0
```

In [77]:

```
# Logistic Regression (LR)
# for Logistic Regression algorithm
# Training
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(X_train,y_train)
y_pred= classifier.predict(X_test)

# Evaluating the Algorithm
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(accuracy_score(y_test,y_pred))
```

```
[[13  0  0]
 [ 0 16  0]
 [ 0  0  9]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        13
Iris-versicolor       1.00      1.00      1.00        16
 Iris-virginica       1.00      1.00      1.00         9

       accuracy                           1.00        38
      macro avg       1.00      1.00      1.00        38
   weighted avg       1.00      1.00      1.00        38

1.0

C:\Users\Dell\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://s
cikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression (https://scikit-learn.org/stable/modules/linear_model.html#logis
tic-regression)
  n_iter_i = _check_optimize_result(
```

In [ ]: