1. Visualizing data (Iris dataset – Part of NB Iris program)

In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set(color_codes = True)


#ignore warning messages
import warnings
warnings.filterwarnings('ignore')
```

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

In [3]:
```python
# Printing top 5 rows
iris.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
#We can see the count of each column along with their mean value, standard deviation, minimum and maximum values.
iris.describe()
```

Out[4]:

	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 [5]:
```python
#We can see that only one column has categorical data and all the other columns are of the numeric type with non-Null entries.
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 [6]:
```python
#We will use the shape parameter to get the shape of the dataset.
iris.shape
#as we can see,there are 6 column and 150 rows.
```

Out[6]:
```
(150, 6)
```

In [7]:
```python
#We will check if our data contains any missing values or not.
iris.isnull().sum()
#as u can see there is 0 null values
```

Out[7]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [8]:
```python
# Let’s see if the dataset is balanced or not i.e. all the species contain equal amounts of rows or not.
iris.value_counts("Species")
```

Out[8]:
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

In [ ]:

In [ ]:

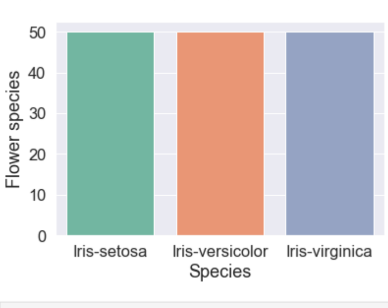
data visualization

In [9]:
```python
# importing packages
import seaborn as sns
#Seaborn is the python library that is used to create the plot and graphical representation of the data,
#internally it maps our data and create an informative plot of the data, which is easy to understand the data.

import matplotlib.pyplot as plt
#Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

sns.countplot(x='Species', data=iris, )
plt.show()


#Seaborn is much more functional and organized than Matplotlib.
```

the above plot is showing the counts of each species, present in dataset.

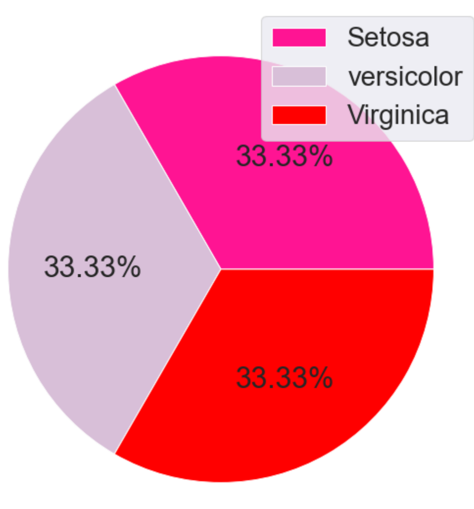
In [10]:
```python
#to check the no. of each species in dataset by bar graph
sns.set(font_scale = 1.5)
sns.countplot(x = 'Species', data=iris, palette="Set2")
plt.ylabel('Flower species')
```

Out[10]:
```
Text(0, 0.5, 'Flower species')
```



In [11]:
```python
#pie chart


plt.figure(figsize=(10,10))
pienew = iris['Species'].value_counts()
#explode = (0.05, 0)
colors = ['deeppink', 'thistle','red']
labels = ['Setosa','versicolor','Virginica']
sns.set(font_scale = 2.5)
plt.pie(pienew, autopct = "%.2f%%",  colors = colors)
plt.legend(labels, loc = 'best')
```

Out[11]:
```
<matplotlib.legend.Legend at 0x1d4b4703370>
```



In [12]:
```python
iris[iris['SepalLengthCm'] ==0 ]   # no missing values
```

Out[12]:

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species

In [13]:
```python
iris[iris['SepalWidthCm'] ==0 ]   # no missing values
```

Out[13]:

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species

In [14]:
```python
iris[iris['PetalLengthCm'] ==0 ]   # no missing values
```

Out[14]:

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species

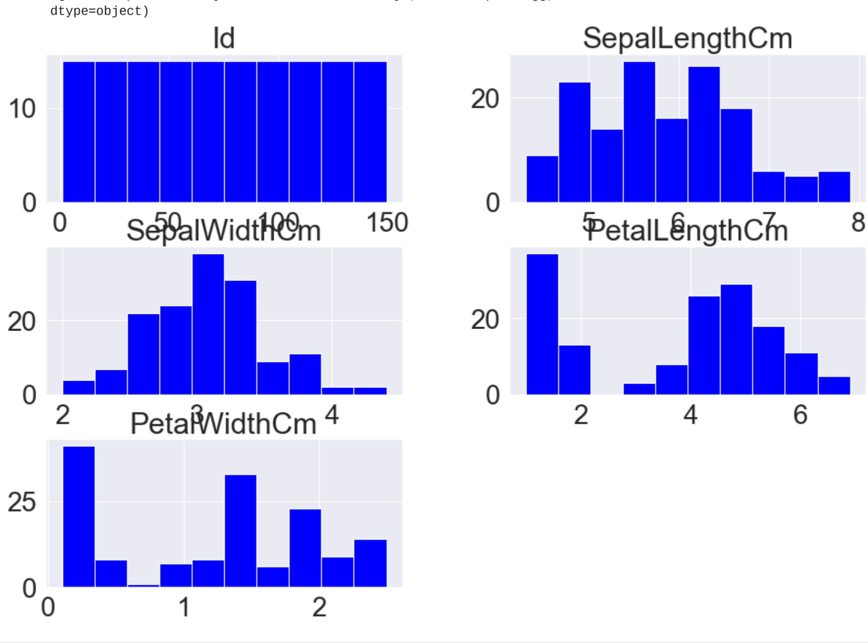
In [15]:
```python
iris[iris['PetalWidthCm'] ==0 ]   # no missing values
```

Out[15]:

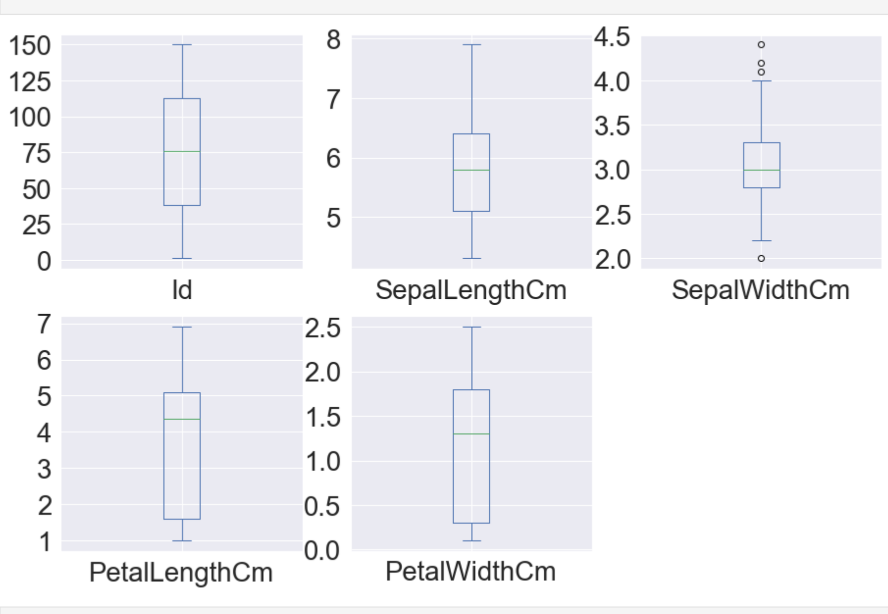
Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species

In [16]:
```python
#Visualization for understanding and analysing the distribution of data for different variables

iris.hist(figsize = (15,10),grid=True,color = 'blue')
```

Out[16]:
```
array([[<AxesSubplot:title={'center':'Id'}>,
        <AxesSubplot:title={'center':'SepalLengthCm'}>],
       [<AxesSubplot:title={'center':'SepalWidthCm'}>,
        <AxesSubplot:title={'center':'PetalLengthCm'}>],
       [<AxesSubplot:title={'center':'PetalWidthCm'}>, <AxesSubplot:>]],
      dtype=object)
```

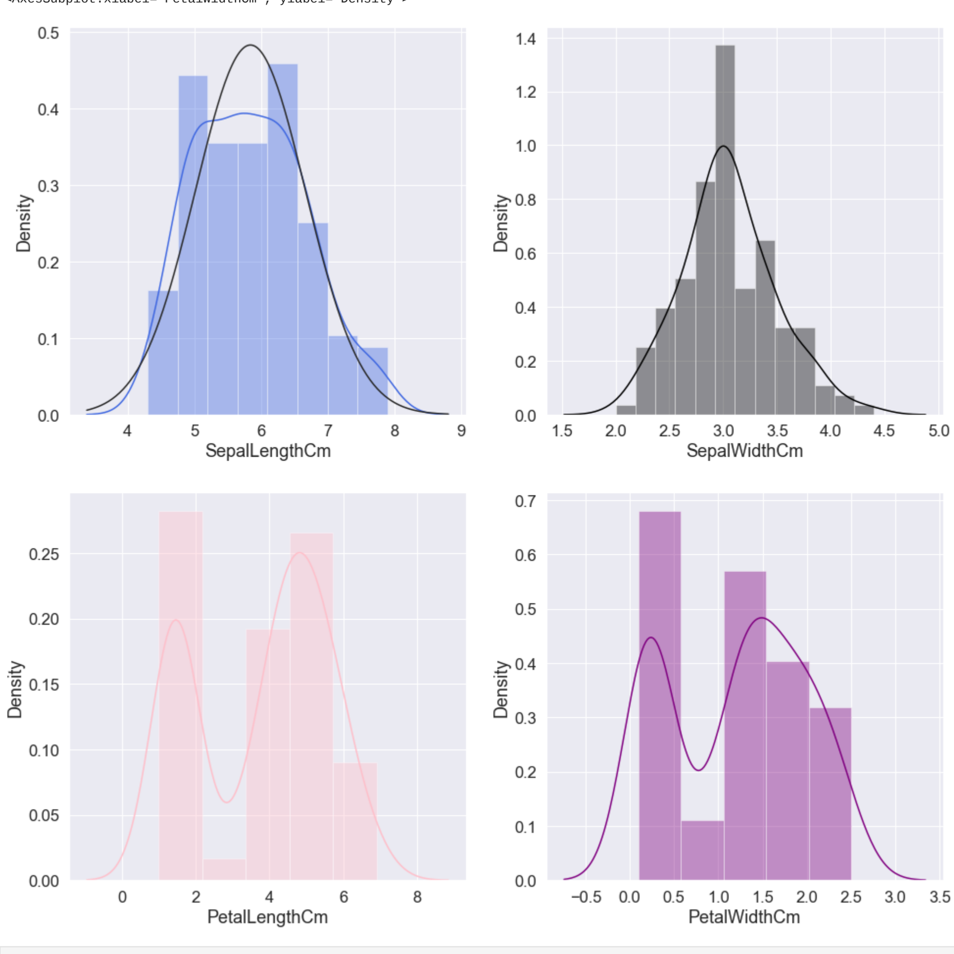


In [17]:
```python
#Box plots are used to show distributions of numeric data values,
#especially when you want to compare them between multiple groups.

iris.plot(kind= 'box' , subplots=True, layout=(3,3), sharex=False, sharey=False, figsize=(15,15))
sns.set(font_scale = 1.5)
```



In [18]:
```python
from scipy.stats import norm
fig, ax = plt.subplots(2,2, figsize=(16,16))
sns.set(font_scale = 1)
sns.distplot(iris.SepalLengthCm, ax = ax[0,0], fit = norm, color = 'royalblue')
sns.distplot(iris.SepalWidthCm, ax = ax[0,1], color = 'black')
sns.distplot(iris.PetalLengthCm, ax = ax[1,0], color = 'pink')
sns.distplot(iris.PetalWidthCm, ax = ax[1,1], color = 'purple')
```

Out[18]:
```
<AxesSubplot:xlabel='PetalWidthCm', ylabel='Density'>
```



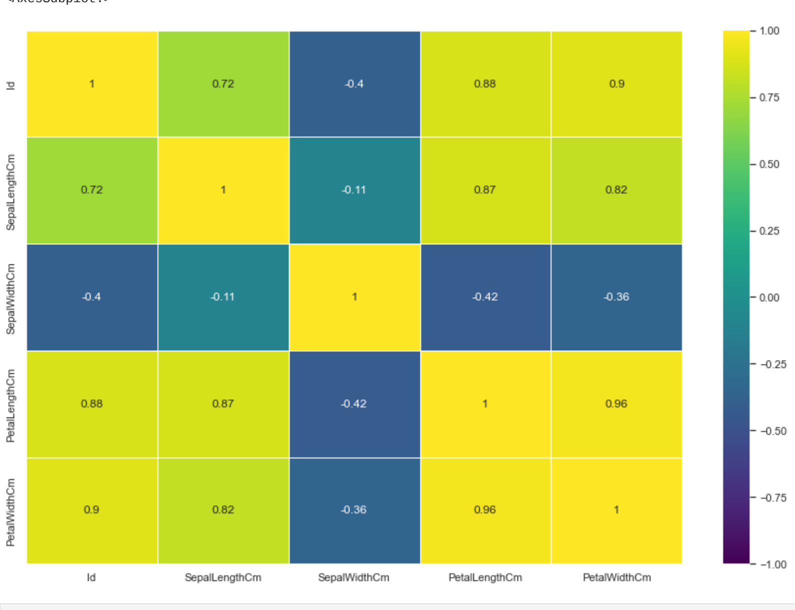
In [19]:
```python
#bi variate analysis of data
#establish the correlation between the length and the width of the sepals and petals of three species of iris flower.
corr = iris.corr()
corr
```

Out[19]:

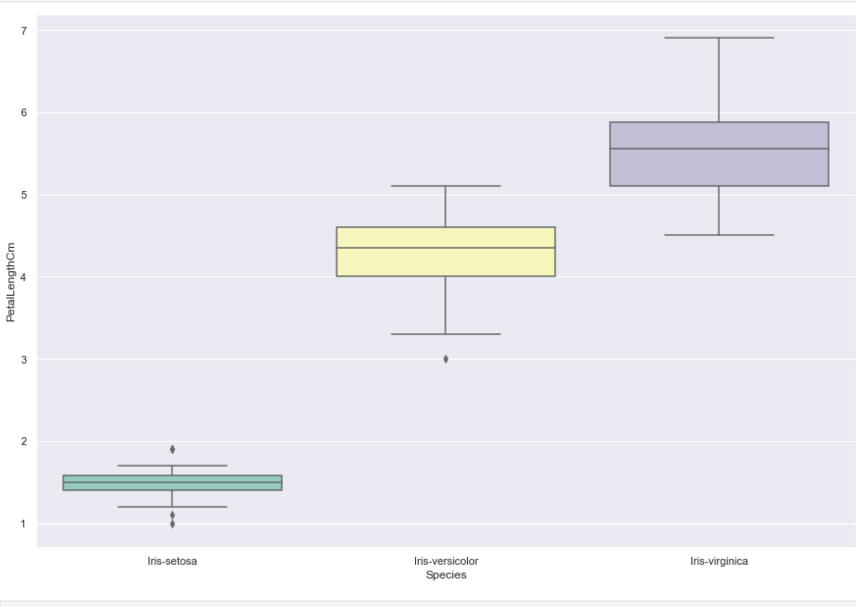
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Id	1.000000	0.716676	-0.397729	0.882747	0.899759
SepalLengthCm	0.716676	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.397729	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.882747	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm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

In [20]:
```python
# heat map
# heatmap is a graphical representation of data where values are depicted by color.

plt.figure(figsize=(15,10))
sns.set(font_scale = 1)
sns.heatmap(corr, annot = True, cmap = 'viridis', vmin = -1, vmax = 1, linecolor='white', linewidths= 1)
```

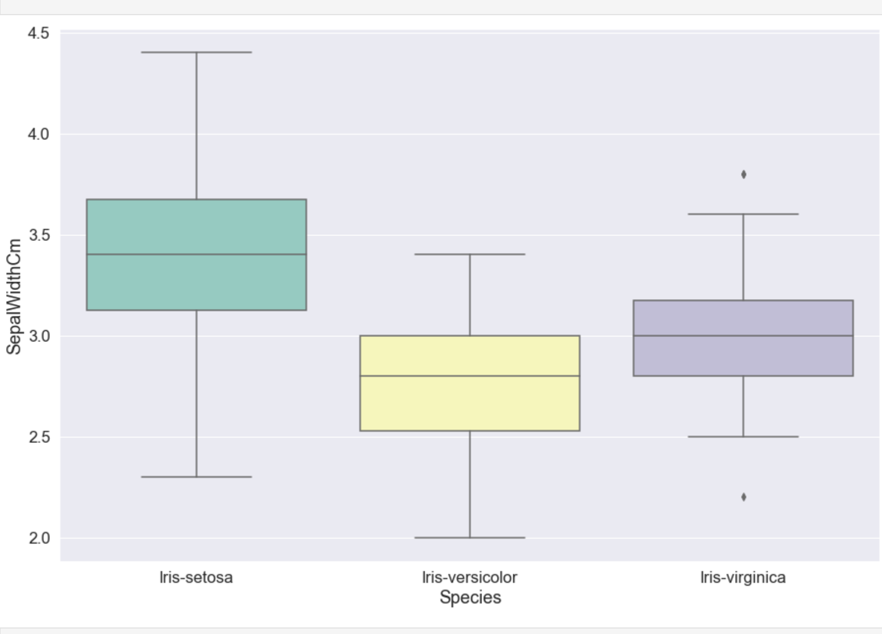
Out[20]:
```
<AxesSubplot:>
```



In [21]:
```python
#box plot of petal length
plt.figure(figsize=(15,10))
sns.boxplot(iris['Species'],iris['PetalLengthCm'], palette="Set3")
sns.set(font_scale = 1.5)
```

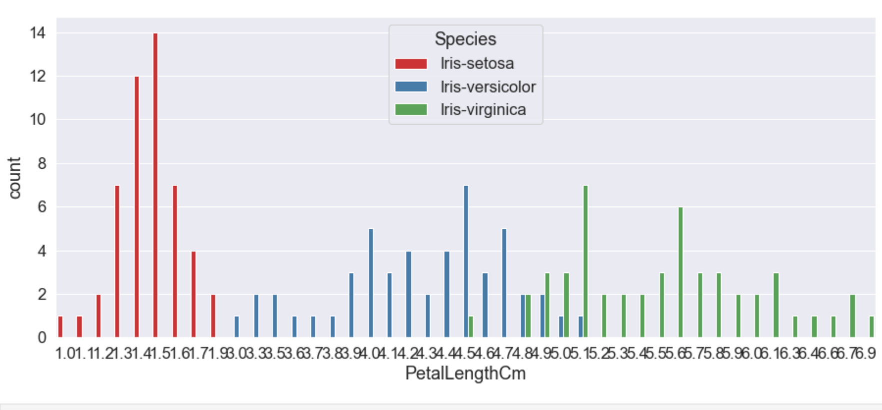


In [22]:
```python
plt.figure(figsize=(15,10))
sns.boxplot(iris['Species'],iris['SepalWidthCm'], palette="Set3")
sns.set(font_scale = 1.5)
```



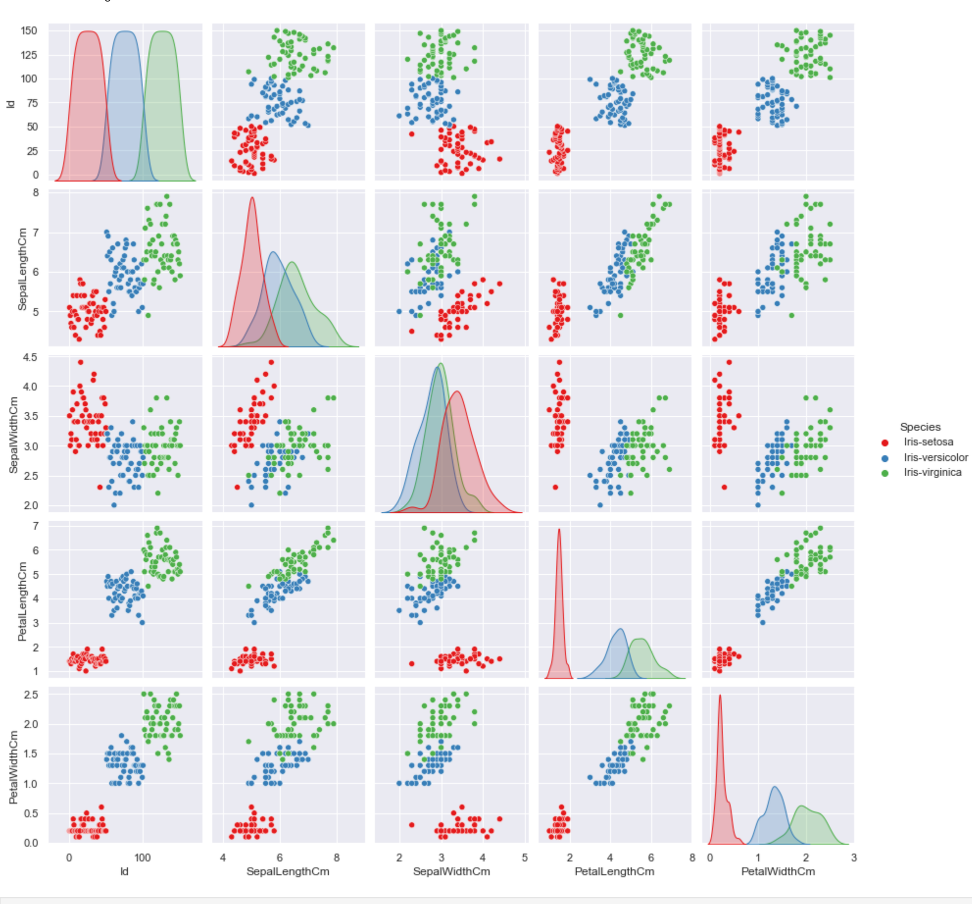
In [23]:
```python
plt.figure(figsize=(15,6))
sns.set(font_scale = 1.5)
sns.countplot(x = 'PetalLengthCm', hue = 'Species', data = iris, palette = 'Set1')
```

Out[23]:
```
<AxesSubplot:xlabel='PetalLengthCm', ylabel='count'>
```



In [24]:
```python
sns.set(font_scale = 1)
sns.pairplot(data = iris, hue = 'Species', diag_kind = 'kde', palette = 'Set1')
```

Out[24]:
```
<seaborn.axisgrid.PairGrid at 0x1d4b4def130>
```



In [25]:
```python
## a relationship between predictor variables and a categorical response variable.
from sklearn.linear_model import LogisticRegression
#Gaussian Naive Bayes is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data.
from sklearn.naive_bayes import GaussianNB
#for evaluating the quality of a model’s predictions
from sklearn import metrics
#A Classification report is used to measure the quality of predictions from a classification algorithm.
#How many predictions are True and how many are False.
from sklearn.metrics import classification_report

# It is mainly used for numerical and predictive analysis
from sklearn.metrics import roc_curve, auc              # building Logistic and Naive Bayes classifiers
```

In [26]:
```python
#to split the dataset for training and testing
from sklearn.model_selection import train_test_split
X = iris.drop('Species', axis  = 1)
y = iris['Species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 1)
```

In [27]:
```python
print(X_train.shape)
print(X_test.shape)
print(y_train.size)
print(y_test.size)
```

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

In [28]:
```python
model = LogisticRegression()
```

In [29]:
```python
model.fit(X_train, y_train)
```

Out[29]:
```
LogisticRegression()
```

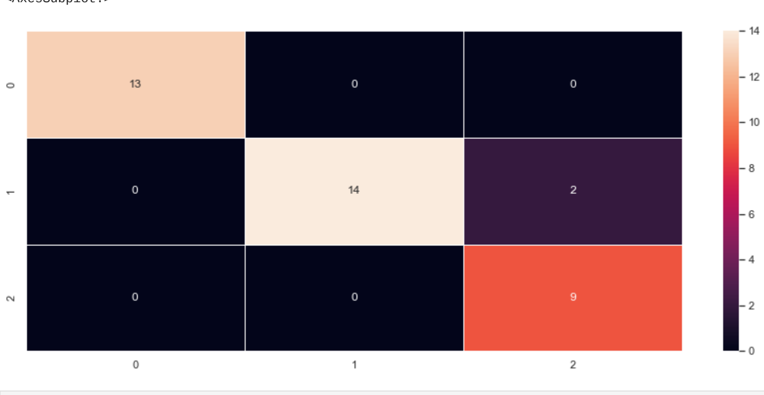
In [30]:
```python
y_pred = model.predict(X_test)
```

In [31]:
```python
#an N x N matrix used for evaluating the performance of a classification model
confusion = metrics.confusion_matrix(y_test,y_pred)
confusion
```

Out[31]:
```
array([[13,  0,  0],
       [ 0, 14,  2],
       [ 0,  0,  9]], dtype=int64)
```

In [32]:
```python
#heat map of the above confusion matrix
plt.figure(figsize=(15,6))
sns.heatmap(confusion, annot=True, linecolor='white', linewidths=1)
```

Out[32]:
```
<AxesSubplot:>
```



In [33]:
```python
print('Accuracy of Logistic Regression is: ', model.score(X_test,y_test) * 100,'%')
```

```
Accuracy of Logistic Regression is:  94.73684210526315 %
```

In [34]:
```python
print(classification_report(y_test,y_pred))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        13
Iris-versicolor       1.00      0.88      0.93        16
 Iris-virginica       0.82      1.00      0.90         9

       accuracy                           0.95        38
      macro avg       0.94      0.96      0.94        38
   weighted avg       0.96      0.95      0.95        38
```

In [35]:
```python
#f1 score
metrics.f1_score(y_test, y_pred, average = 'weighted')
```

Out[35]:
```
0.9482456140350877
```

In [36]:
```python
# Naive Bayes CLassifier
nbModel = GaussianNB()
```

In [37]:
```python
nbModel = GaussianNB()
```

In [38]:
```python
#fitting the model
nbModel.fit(X_train, y_train)
```

Out[38]:
```
GaussianNB()
```

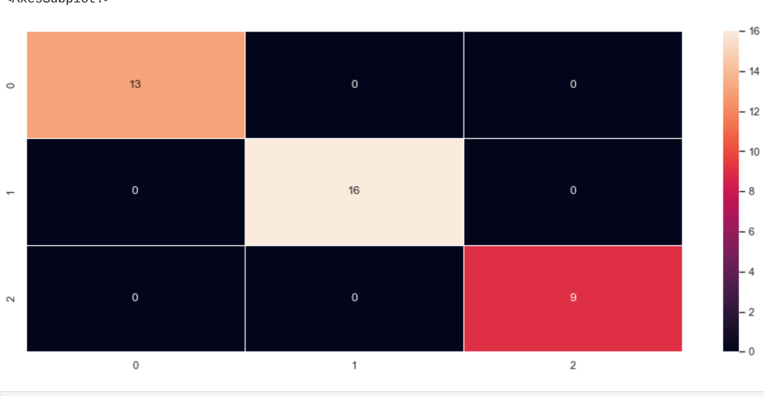
In [39]:
```python
#predicting the X-test
nb_y_pred = nbModel.predict(X_test)
```

In [40]:
```python
#matrix of y-test and ,y-pred
nbConfusion = metrics.confusion_matrix(y_test, nb_y_pred)
nbConfusion
```

Out[40]:
```
array([[13,  0,  0],
       [ 0, 16,  0],
       [ 0,  0,  9]], dtype=int64)
```

In [41]:
```python
#heat map for above confusion matrix
plt.figure(figsize=(15,6))
sns.heatmap(nbConfusion, annot=True,  linecolor='white', linewidths=1)
```

Out[41]:
```
<AxesSubplot:>
```



In [45]:
```python
#accuracy
print('Accuracy of Naive Bayes Classifier is: ', nbModel.score(X_test,y_test)*100 ,'%')
```

```
Accuracy of Naive Bayes Classifier is:  100.0 %
```

In [ ]:

In [ ]: