In [1]:

```
# import all required libaries
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

from sklearn.linear_model import LogisticRegression as lr
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

In [2]:

```
irisdf = pd.read_csv("iris.csv")
#first 5 dataset
irisdf.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]:

```

#last 5 dataset
irisdf.tail()
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [4]:

```

#description of dataset
irisdf.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]:

```
# Dimension of data set
irisdf.shape
```

Out[5]:

```
(150, 6)
```

In [6]:

```
irisdf.size
```

Out[6]:

```
900
```

In [7]:

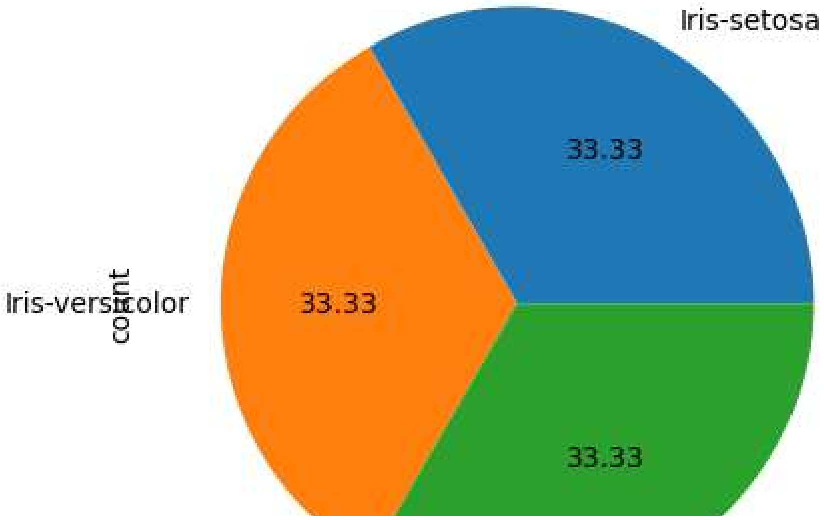
```
#check for values of each species whether its balanced or imbalanced
irisdf['Species'].value_counts()
```

Out[7]:

```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

```
In [8]:  # Visualization in form of pie chart
         irisdf['Species'].value_counts().plot(kind='pie',autopct='%.2f')
```

```
Out[8]:  <Axes: ylabel='count'>
```



```
In [9]:  #Check if any null values is present in give dataset
         irisdf.isnull().sum()
```

```
Out[9]:  Id               0
         SepalLengthCm    0
         SepalWidthCm     0
         PetalLengthCm    0
         PetalWidthCm     0
         Species          0
         dtype: int64
```

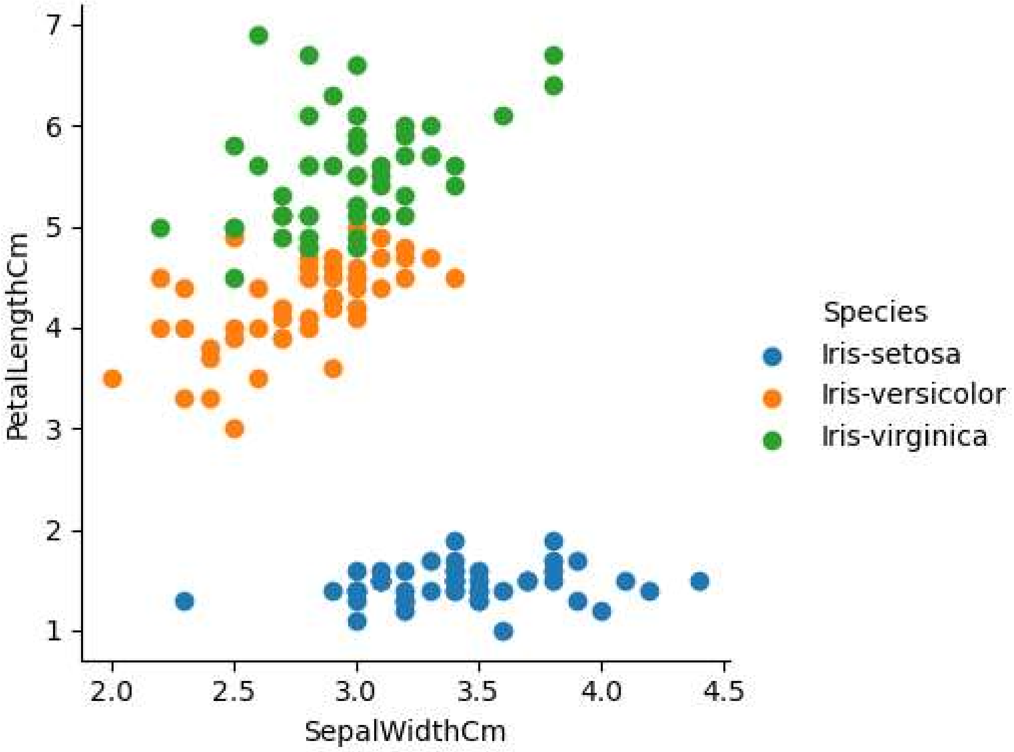
```
In [10]: #Check for duplicate values
         duplicate_count = irisdf.duplicated().sum()
         print(duplicate_count)
```

```
         0
```

Exploratory Data Analysis

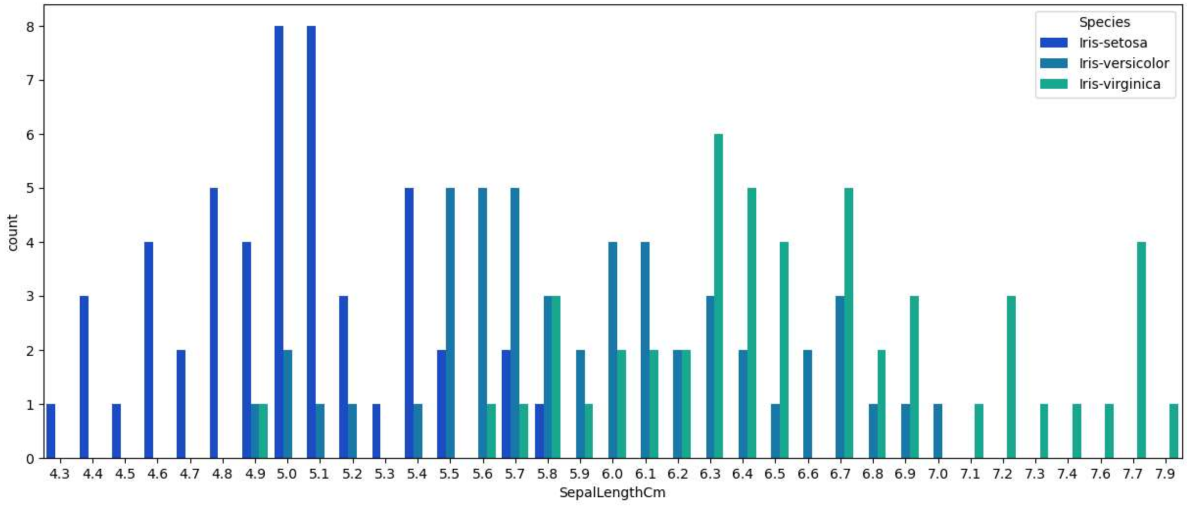
In [11]:
```
#Scatter Plot to  visualize the datset
sns.FacetGrid(irisdf,hue='Species',height=4).map(plt.scatter,"SepalWidthCm" ,
```

Out[11]:
```
<seaborn.axisgrid.FacetGrid at 0x21607287110>
```

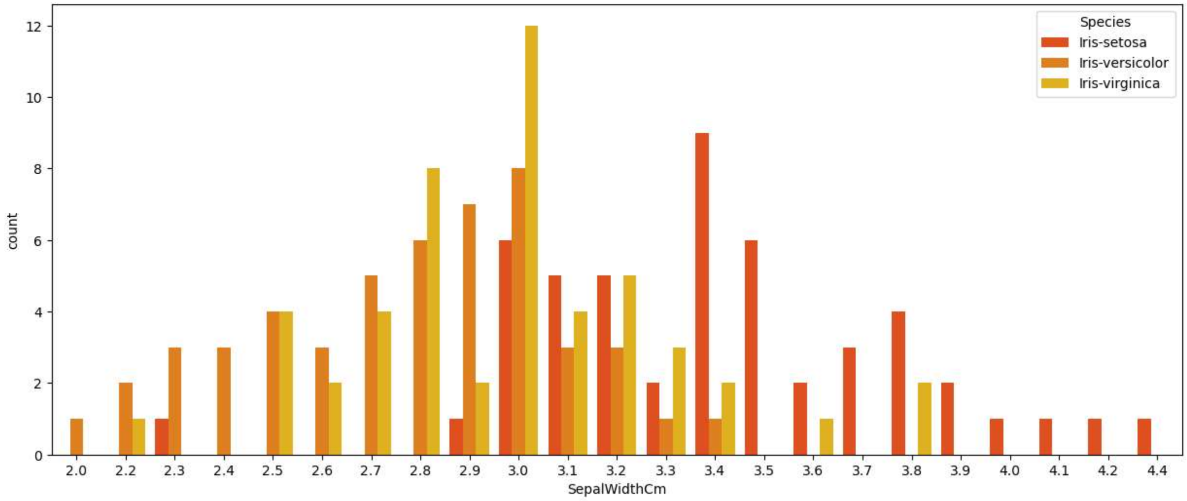


In [12]:
```

plt.figure(figsize =(15,6))
sns.countplot(x='SepalLengthCm', data=irisdf, hue= irisdf['Species'], palette=
plt.show()
```

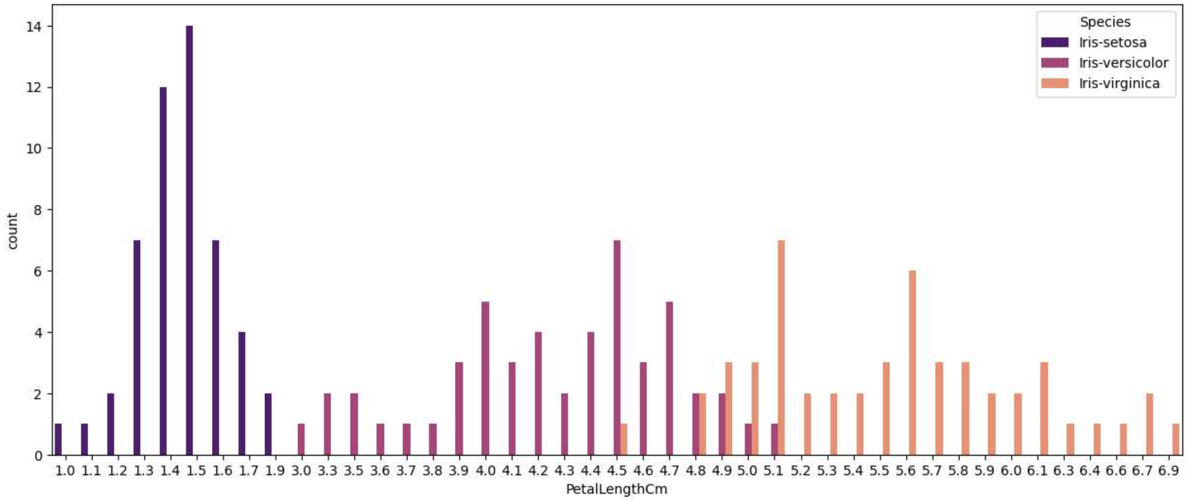


In [13]:
```
plt.figure(figsize =(15,6))
sns.countplot(x='SepalWidthCm', data=irisdf, hue= irisdf['Species'], palette=
plt.show()
```



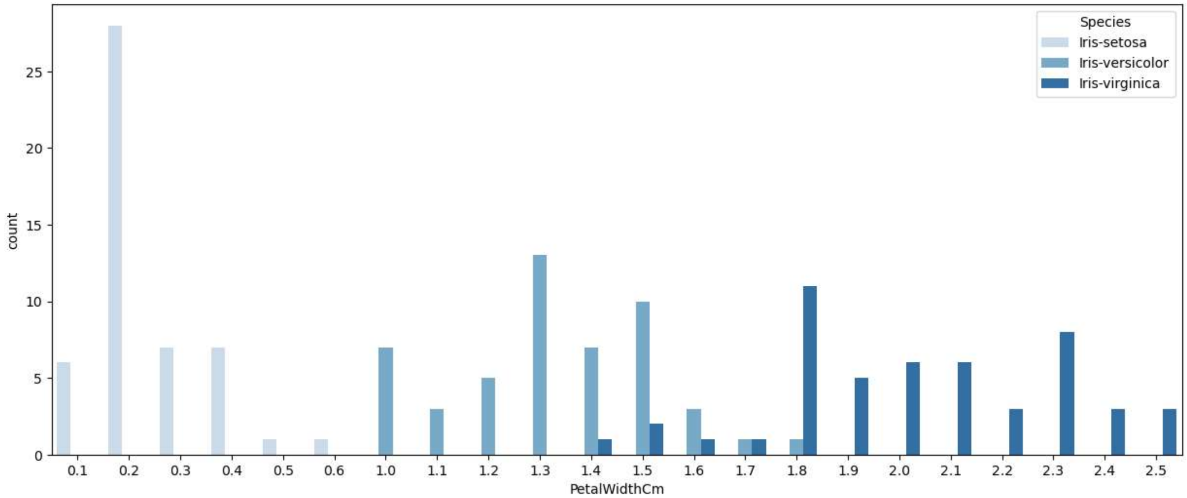
In [14]:
```

plt.figure(figsize =(15,6))
sns.countplot(x='PetalLengthCm', data=irisdf, hue= irisdf['Species'], palette=
plt.show()
```



In [15]:

```
plt.figure(figsize =(15,6))
sns.countplot(x='PetalWidthCm', data=irisdf, hue= irisdf['Species'], palette=
plt.show()
```



In [28]:

```
irisdf.corr()
```

Out[28]:

```
<bound method DataFrame.corr of       Id  SepalLengthCm  SepalWidthCm  PetalL
engthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]>
```

In [30]:

```
# Coor in form of heat map
corr = irisdf.corr()
sns.heatmap(corr,annot=True,cmap='Blues')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[30], line 3
      1 # Coor in form of heat map
      2 corr = irisdf.corr
----> 3 sns.heatmap(corr,annot=True,cmap='Blues')

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\matrix.py:446, in hea
tmap(data, vmin, vmax, cmap, center, robust, annot, fmt, annot_kws, linewidth
s, linecolor, cbar, cbar_kws, cbar_ax, square, xticklabels, yticklabels, mas
k, ax, **kwargs)
    365 """Plot rectangular data as a color-encoded matrix.
    366 
    367 This is an Axes-level function and will draw the heatmap into the
   (...)
    443 
    444 """
    445 # Initialize the plotter object
--> 446 plotter = _HeatMapper(data, vmin, vmax, cmap, center, robust, annot,
fmt,
    447                       annot_kws, cbar, cbar_kws, xticklabels,
    448                       yticklabels, mask)
    450 # Add the pcolormesh kwargs here
    451 kwargs["linewidths"] = linewidths

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\matrix.py:110, in _He
atMapper.__init__(self, data, vmin, vmax, cmap, center, robust, annot, fmt, a
nnot_kws, cbar, cbar_kws, xticklabels, yticklabels, mask)
    108 else:
    109     plot_data = np.asarray(data)
--> 110     data = pd.DataFrame(plot_data)
    112 # Validate the mask and convert to DataFrame
    113 mask = _matrix_mask(data, mask)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:758, in
DataFrame.__init__(self, data, index, columns, dtype, copy)
    747         mgr = dict_to_mgr(
    748             # error: Item "ndarray" of "Union[ndarray, Series, Inde
x]" has no
    749             # attribute "name"
   (...)
    755             copy=_copy,
    756         )
    757     else:
--> 758         mgr = ndarray_to_mgr(
    759             data,
    760             index,
    761             columns,
    762             dtype=dtype,
    763             copy=copy,
    764             typ=manager,
    765         )
    767 # For data is list-like, or Iterable (will consume into list)
    768 elif is_list_like(data):

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\internals\constru
ction.py:315, in ndarray_to_mgr(values, index, columns, dtype, copy, typ)
```

```
    309     _copy = (
    310         copy_on_sanitize
    311         if (dtype is None or astype_is_view(values.dtype, dtype))
    312         else False
    313     )
    314     values = np.array(values, copy=_copy)
--> 315     values = _ensure_2d(values)
    317 else:
    318     # by definition an array here
    319     # the dtypes will be coerced to a single dtype
    320     values = _prep_ndarraylike(values, copy=copy_on_sanitize)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\internals\constru
ction.py:570, in _ensure_2d(values)
    568     values = values.reshape((values.shape[0], 1))
    569 elif values.ndim != 2:
--> 570     raise ValueError(f"Must pass 2-d input. shape={values.shape}")
    571 return values

ValueError: Must pass 2-d input. shape=()
```

```
In [ ]:
```