In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
data = pd.read_csv ('Iris.csv')
data.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]:

```
data.describe()
```

Out[3]:

	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 [4]:

```
data.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]:

```
data.isnull().sum()
```

Out[5]:

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

In [6]:

```
lab = data.Species.unique().tolist()
lab
```

Out[6]:

```
['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
```

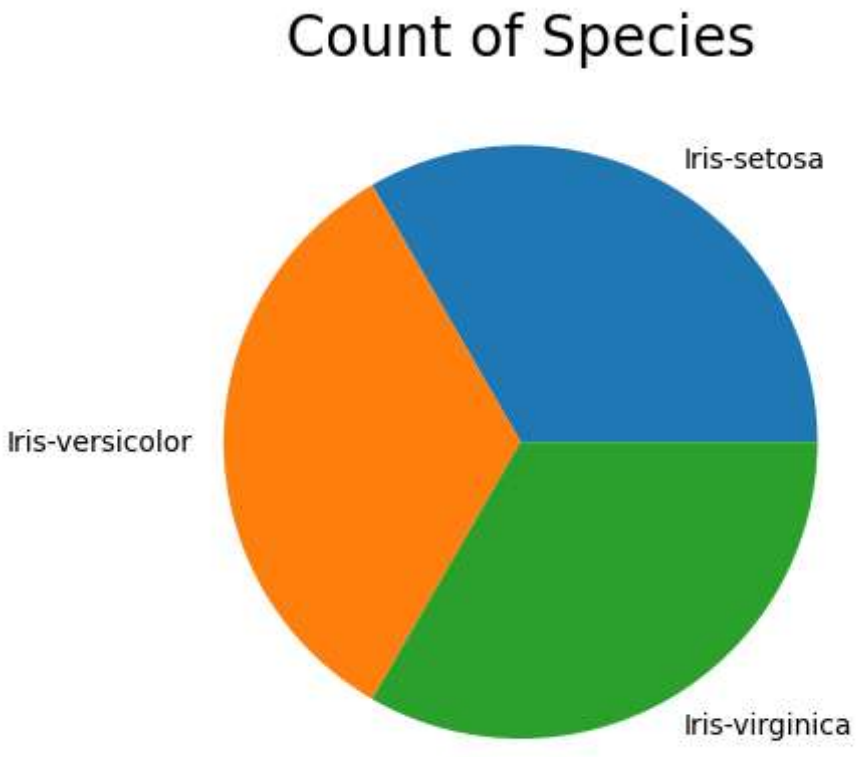
In [8]:

```
count =  data.Species.value_counts()
print(count)
```

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

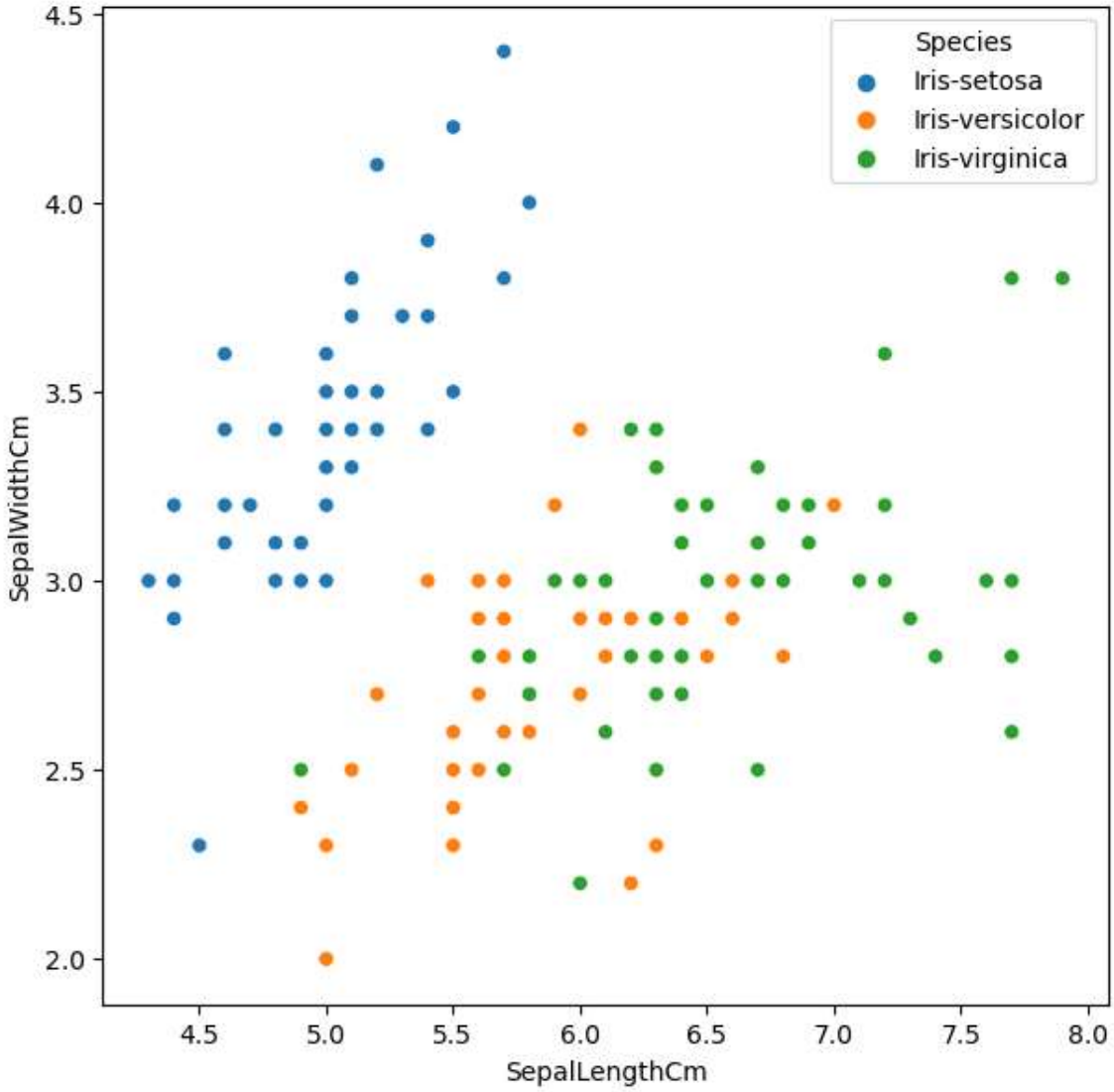
In [9]:

```
plt.pie(count,labels=lab)
plt.title("Count of Species",fontsize=20)
plt.show()
```



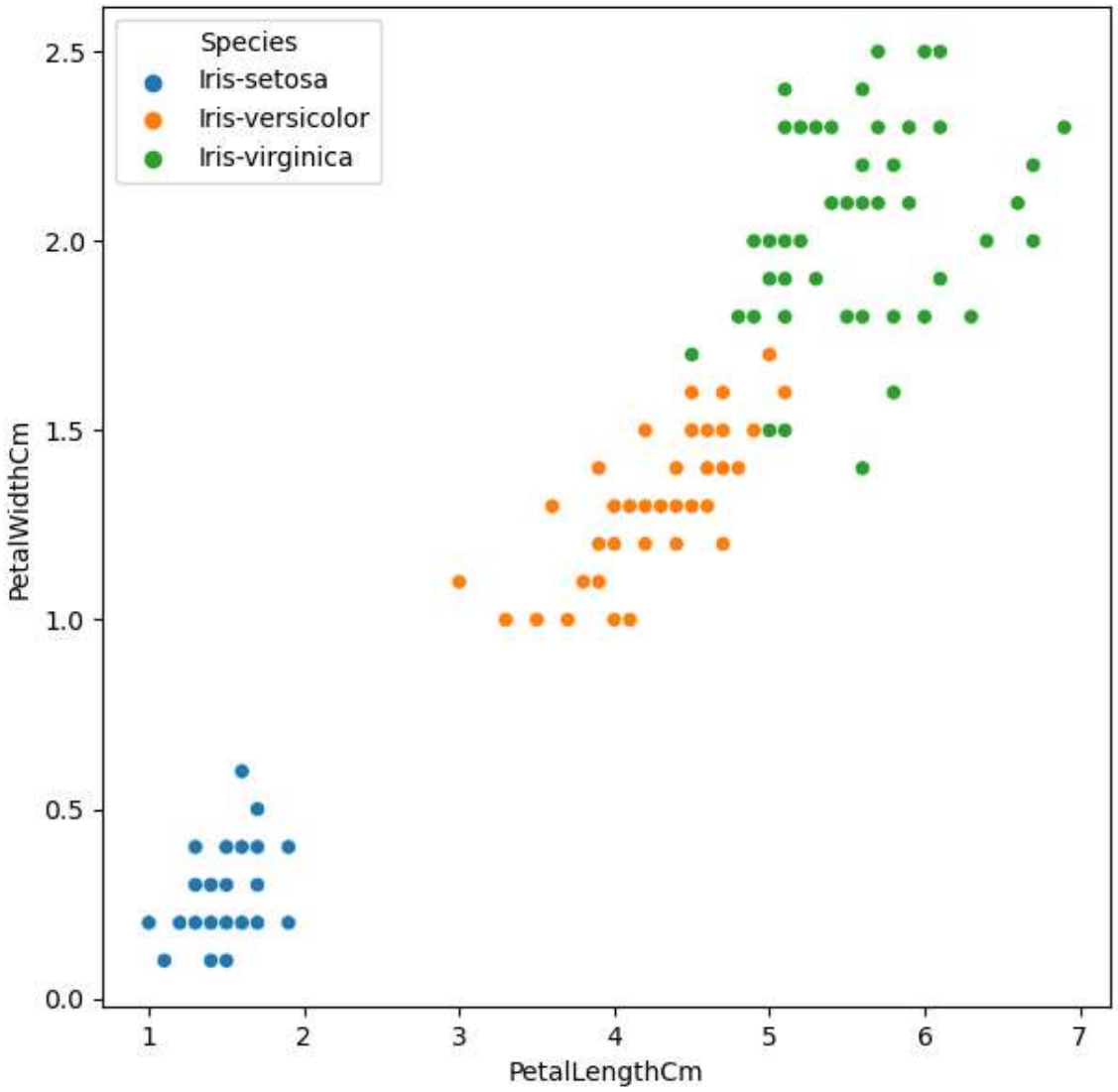
In [11]:

```
plt.subplots(figsize=(7,7))
sns.scatterplot(x="SepalLengthCm",y="SepalWidthCm",data=data,hue="Species")
plt.show()
```



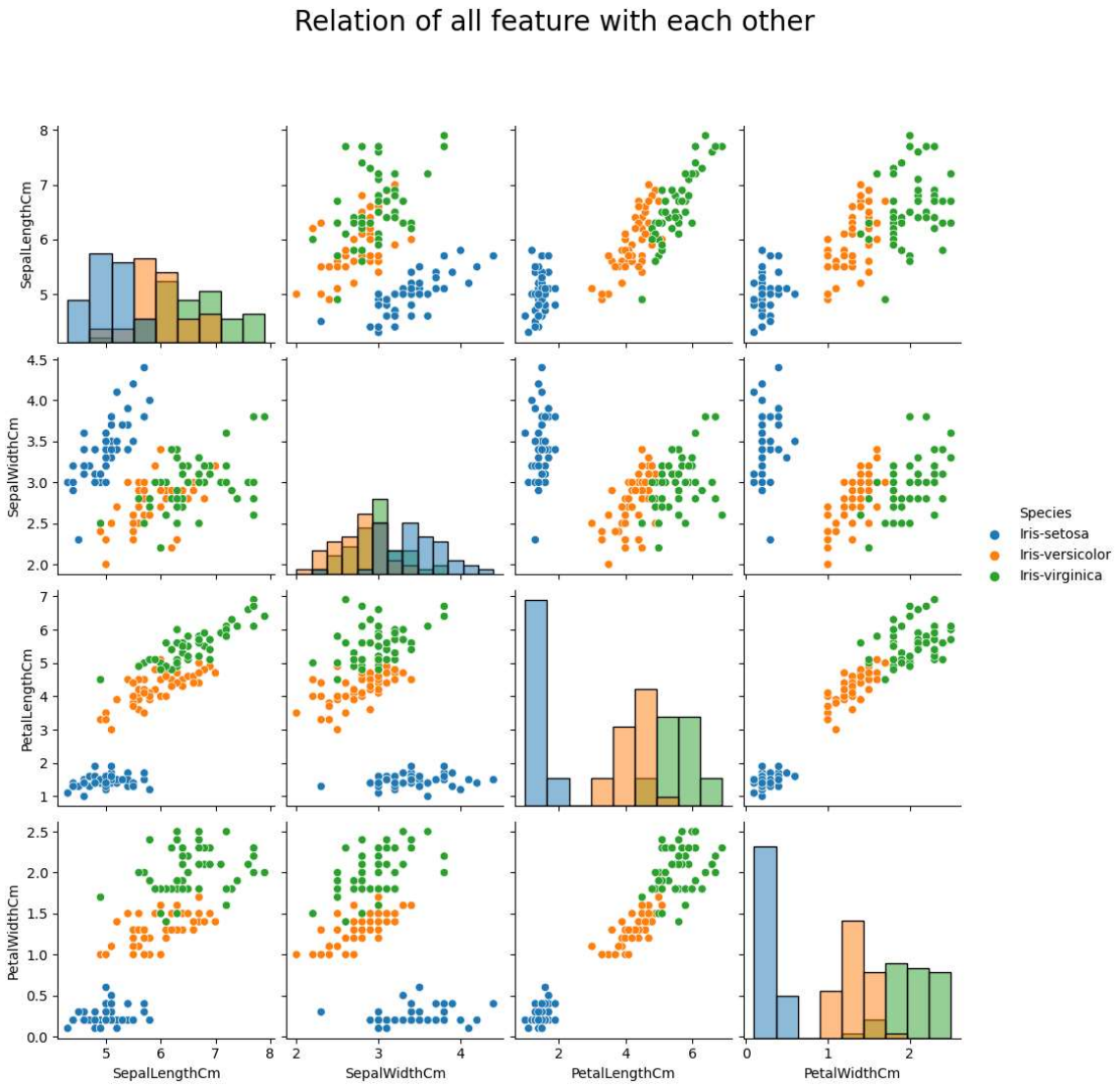
In [12]:

```
plt.subplots(figsize=(7,7))
sns.scatterplot(x="PetalLengthCm",y="PetalWidthCm",data=data,hue="Species")
plt.show()
```



In [13]:

```
data1 = data.drop("Id",axis=1)
plot=sns.pairplot(data1,hue="Species",diag_kind="hist")
plot.fig.suptitle("Relation of all feature with each other",y=1.1,fontsize=20)
plt.show()
```



In [14]:

```
from sklearn.model_selection import train_test_split
```

In [15]:

```
X = data.drop(["Species","Id"],axis=1)
X
```

Out[15]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [16]:

```
Y = data["Species"]
Y
```

Out[16]:

```
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
Name: Species, Length: 150, dtype: object
```

In [17]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
```

In [18]:

```
model = LogisticRegression(max_iter=1000)
```

In [19]:

```
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.3,random_state=1)
model.fit(x_train,y_train)
```

Out[19]:

```
LogisticRegression(max_iter=1000)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [20]:

```
predictions = model.predict(x_test)
print(classification_report(y_test,predictions))
print("Confusion Matrix\n",confusion_matrix(y_test,predictions))
```

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      0.94      0.97        18
 Iris-virginica       0.93      1.00      0.96        13

       accuracy                           0.98        45
      macro avg       0.98      0.98      0.98        45
   weighted avg       0.98      0.98      0.98        45

Confusion Matrix
 [[14  0  0]
 [ 0 17  1]
 [ 0  0 13]]
```

In [21]:

```
print(accuracy_score(y_test,predictions))
```

```
0.9777777777777777
```

In [ ]: