```
#Import scikit-learn dataset library
from google.colab import files
import io
import pandas as pd
import numpy as np
#import svm
from sklearn import svm
#import test_train_splt
from sklearn.model_selection import train_test_split
#import metrics for accuracy
from sklearn import metrics
#importing ggplot
from plotnine import ggplot, aes , geom_point


#Load dataset
uploaded = files.upload()
iris = pd.read_csv(io.BytesIO(uploaded['iris-1.csv']))
```

Choose Files iris-1.csv

- **iris-1.csv**(text/csv) - 5164 bytes, last modified: 9/6/2022 - 100% done

```
Saving iris-1.csv to iris-1.csv
```

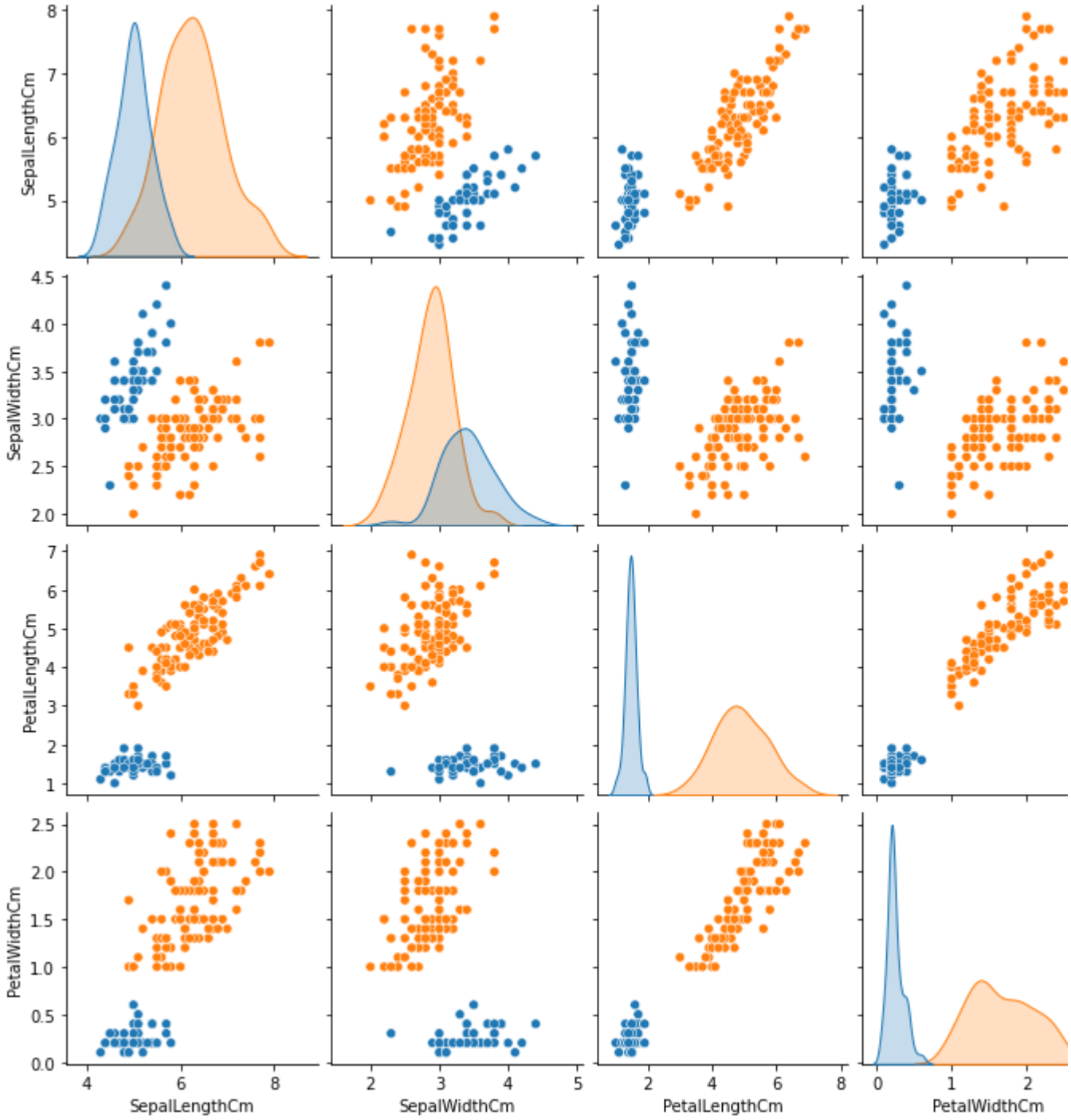
**Exploring the Data Set **

```
#Exploring the data set
print(iris.shape)
print(iris.head())
print(iris.isnull().sum())
```

```
(150, 6)
   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
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

```
import seaborn
import matplotlib.pyplot as plt
seaborn.pairplot(iris.drop(['Id'], axis =1), hue = "Species")
plt
```

```
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.7/dist-packages/matplotlib/pypl
```



SVM

```
X = iris.drop(['Species','Id'], axis=1)
y = iris["Species"]


# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2) # 80% training and
```

```
#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)


# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:",metrics.precision_score(y_test, y_pred, pos_label='Iris-setosa'))

# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_pred, pos_label='Iris-setosa'))
```

```
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
```