Tugas Pertemuan 11_Andrean Yonathan_Institut Teknologi Sepuluh Nopember

1. Data Preprocessing

Latihan (1)

Melakukan import library yang dibutuhkan

In [1]:
```
# import library pandas
import pandas as pd

# Import library numpy
import numpy as np

# Import library matplotlib dan seaborn untuk visualisasi
import matplotlib.pyplot as plt
import seaborn as sns

# me-non aktifkan peringatan pada python
import warnings
warnings.filterwarnings('ignore')
```

Load Dataset

In [2]:
```
#Panggil file (load file bernama Iris_AfterClean.csv) dan simpan dalam dataframe
dataset = 'Iris_AfterClean.csv'
iris = pd.read_csv(dataset)
```

Latihan (2)

Review Dataset

In [3]:
```
# tampilkan 5 baris awal dataset dengan function head()
iris.head()
```

Out[3]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	4.6	3.1	1.5	0.2	Iris-setosa
1	5.0	3.6	1.4	0.2	Iris-setosa
2	5.4	3.9	1.7	0.4	Iris-setosa
3	4.9	3.1	1.5	0.1	Iris-setosa
4	5.4	3.7	1.5	0.2	Iris-setosa

In [4]:
```
# tampilkan unique value dari species
iris['Species'].unique()
```

Out[4]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

dari output diatas, dataset ini memiliki tiga varietas tanaman Iris.

In [5]:
```
# melihat statistik data untuk data numeric dan non numeric
iris.describe(include='all')
```

Out[5]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	140.000000	140.000000	140.000000	140.000000	140
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	Iris-virginica
freq	NaN	NaN	NaN	NaN	50
mean	5.902857	3.028571	3.910714	1.262857	NaN
std	0.819365	0.398791	1.720369	0.746825	NaN
min	4.300000	2.200000	1.000000	0.100000	NaN
25%	5.200000	2.800000	1.675000	0.400000	NaN
50%	5.850000	3.000000	4.500000	1.400000	NaN
75%	6.425000	3.300000	5.100000	1.800000	NaN
max	7.900000	4.000000	6.900000	2.500000	NaN

In [6]:
```
# Melihat Informasi lebih detail mengenai struktur DataFrame dapat dilihat menggunakan fungsi info()
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 140 entries, 0 to 139
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   SepalLengthCm  140 non-null    float64
 1   SepalWidthCm   140 non-null    float64
 2   PetalLengthCm  140 non-null    float64
 3   PetalWidthCm   140 non-null    float64
 4   Species        140 non-null    object
dtypes: float64(4), object(1)
memory usage: 5.6+ KB
```

Seperti yang kita lihat di atas distribusi tiap data di setiap kelas adalah sama sehingga Iris adalah dataset seimbang

Latihan (3)

Checking if there are any missing values

In [7]:
```
# cek jumlah nilai yang hilang / missing values dari setiap kolom dengan function isnull() dan sum()
iris.isnull().sum()
```

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

In [8]:
```
# cek missing values dengan visualisasi menggunakan library: Missingno adalah pustaka khusus untuk menampilkan
# jenis barchart
import missingno as msno
msno.bar(iris,figsize=(8,6),color='skyblue')
plt.show()
```

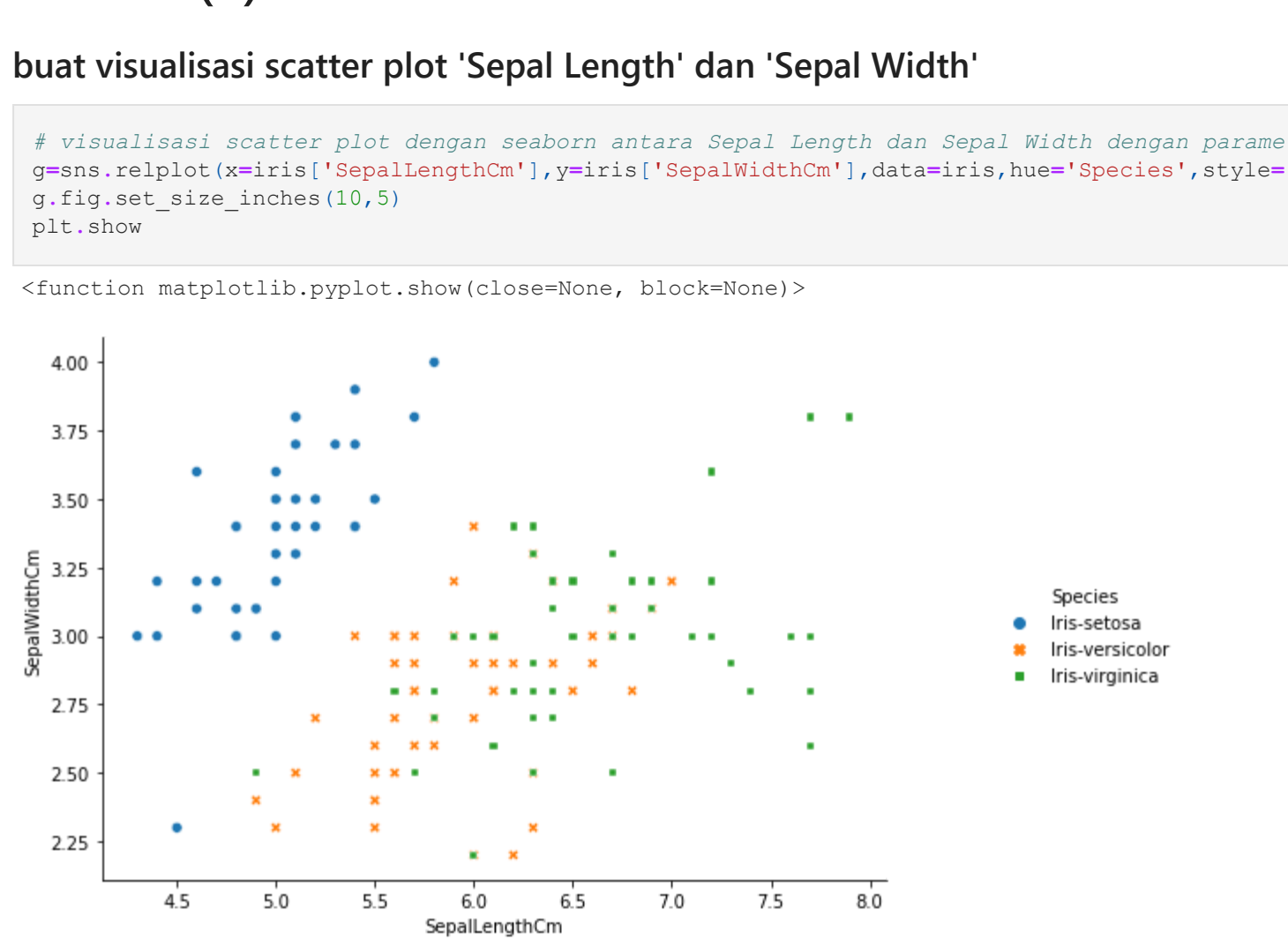
2. Data Visualization

Latihan (4)

buat visualisasi scatter plot 'Sepal Length' dan 'Sepal Width'

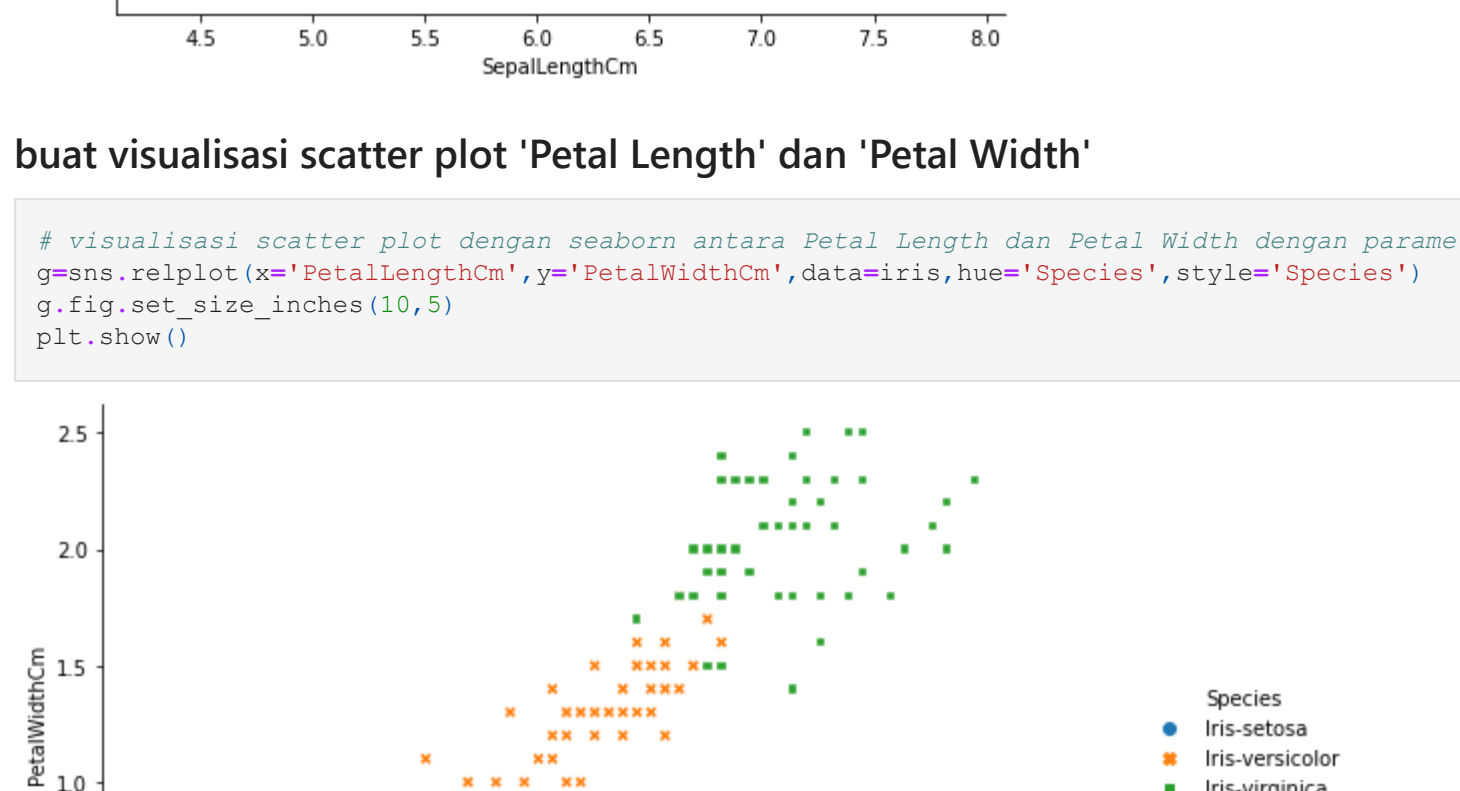
In [9]:
```
# visualisasi scatter plot dengan seaborn antara Sepal Length dan Sepal Width dengan parameter hue dan style='s
g=sns.relplot(x=iris['SepalLengthCm'],y=iris['SepalWidthCm'],data=iris,hue='Species',style='Species')
g.fig.set_size_inches(10,5)
plt.show
```

Out[9]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



buat visualisasi scatter plot 'Petal Length' dan 'Petal Width'

In [10]:
```
# visualisasi scatter plot dengan seaborn antara Petal Length dan Petal Width dengan parameter hue dan style='s
g=sns.relplot(x='PetalLengthCm',y='PetalWidthCm',data=iris,hue='Species',style='Species')
g.fig.set_size_inches(10,5)
plt.show()
```

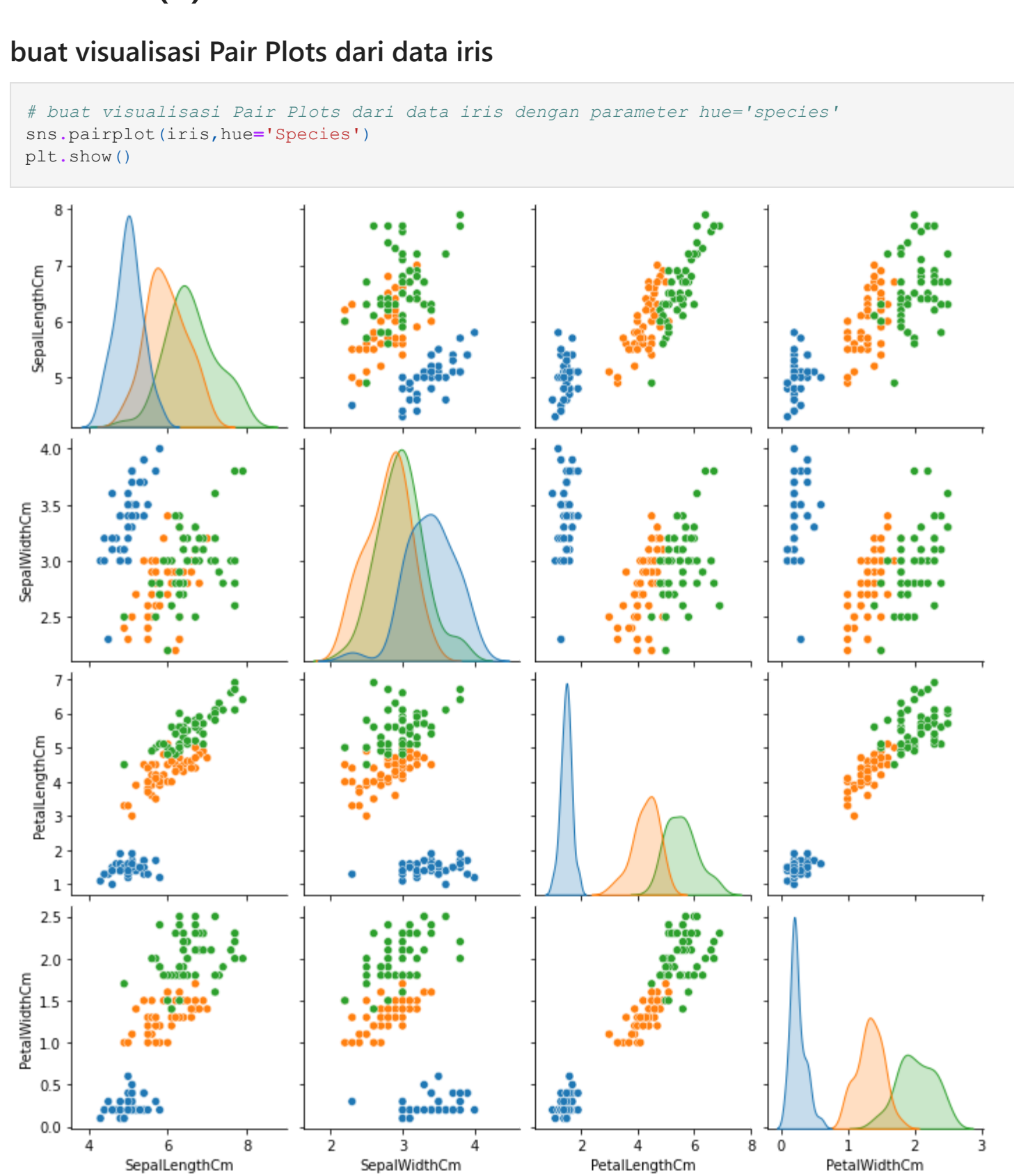


Seperti yang dapat kita lihat bahwa Fitur Petal memberikan pembagian cluster yang lebih baik dibandingkan dengan fitur Sepal. Ini merupakan indikasi bahwa Petal dapat membantu dalam Prediksi yang lebih baik dan akurat dari pada Sepal.

Latihan (5)

buat visualisasi Pair Plots dari data iris

In [11]:
```
# buat visualisasi Pair Plots dari data iris dengan parameter hue='species'
sns.pairplot(iris,hue='Species')
plt.show()
```

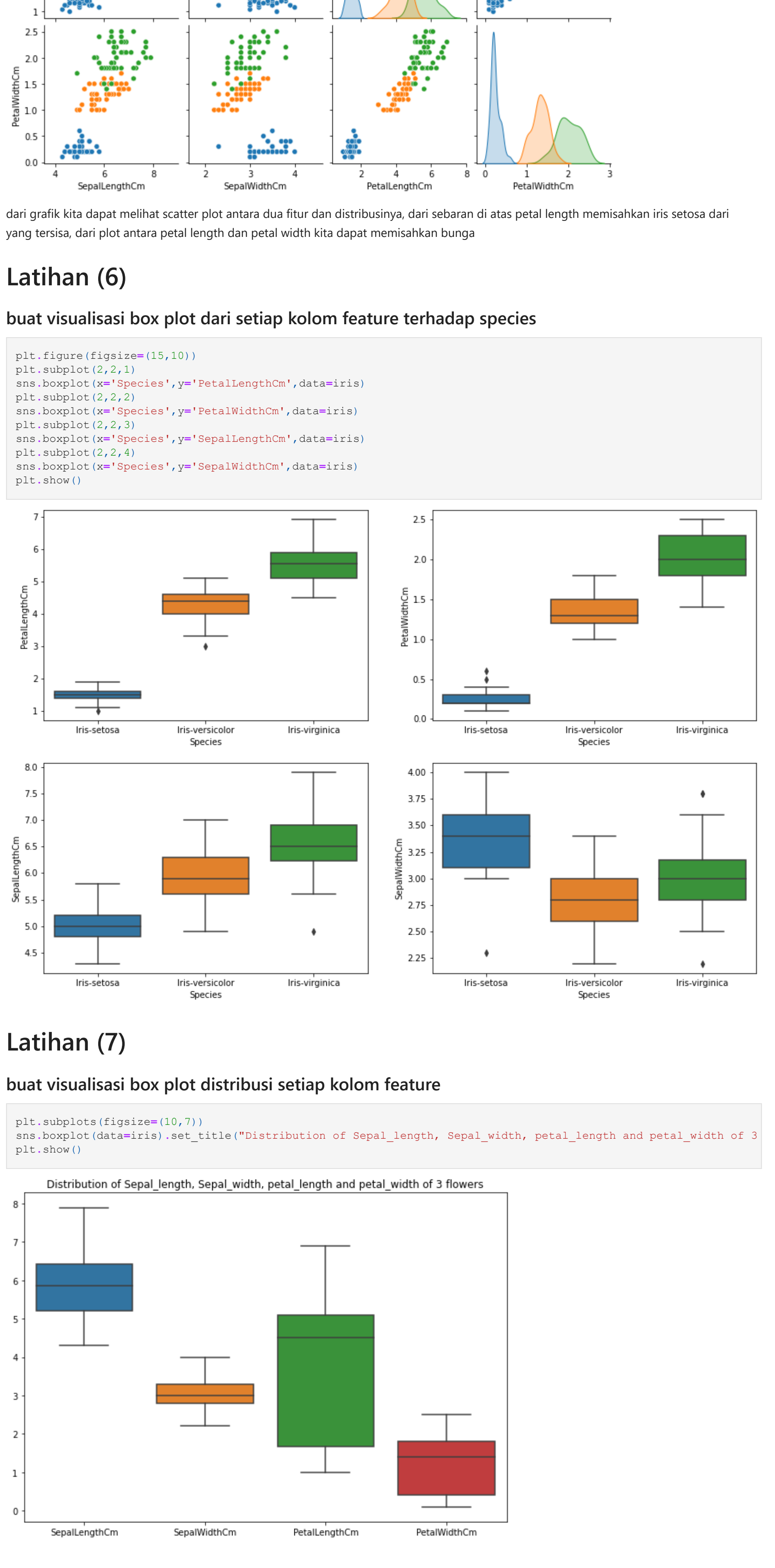


dari grafik kita dapat melihat scatter plot antara dua fitur dan distribusinya, dari sebaran di atas petal length memisahkan iris setosa dari yang tersisa, dari plot antara petal length dan petal width kita dapat memisahkan bunga

Latihan (6)

buat visualisasi box plot dari setiap kolom feature terhadap species

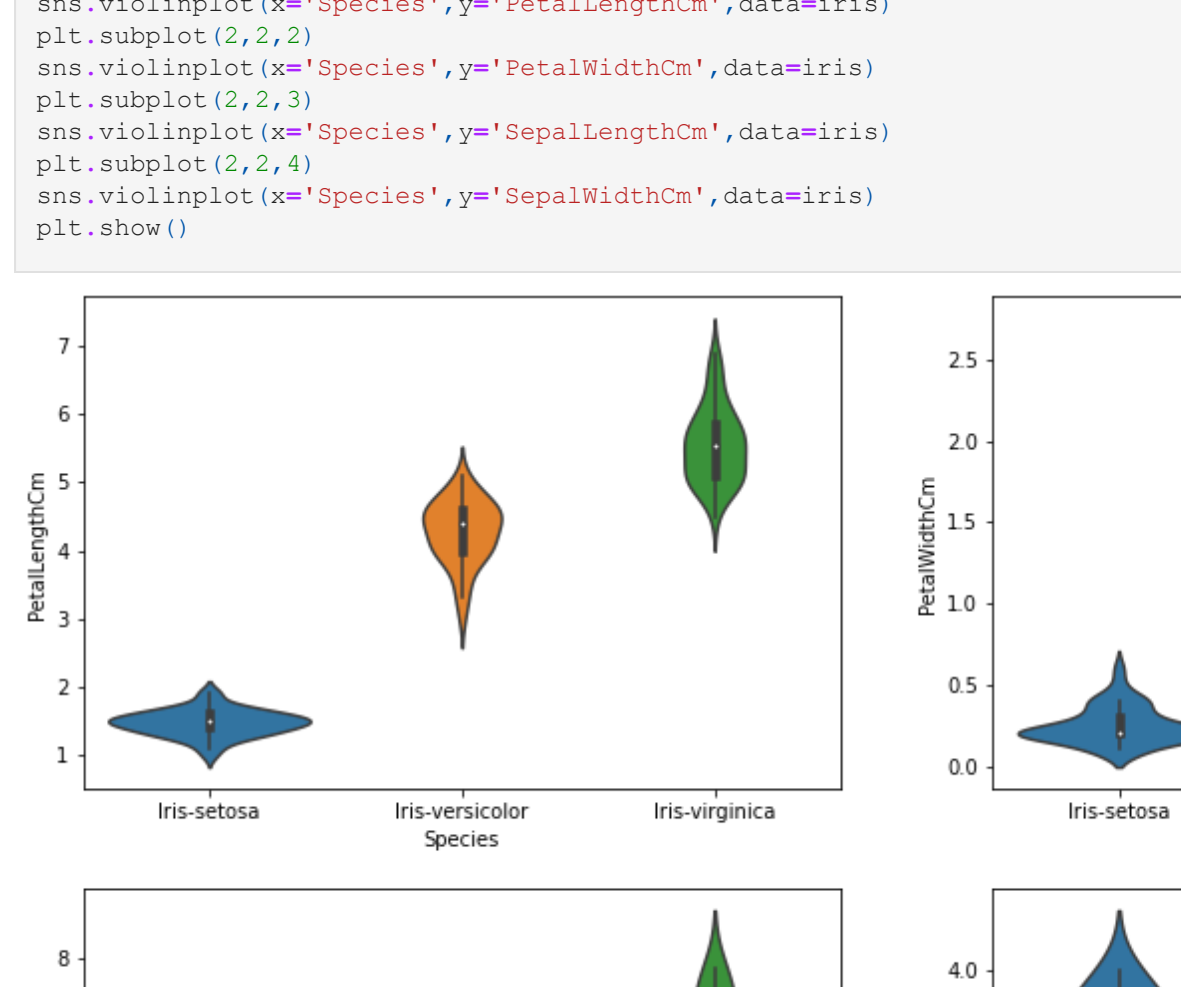
In [12]:
```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.boxplot(x='Species',y='PetalLengthCm',data=iris)
plt.subplot(2,2,2)
sns.boxplot(x='Species',y='PetalWidthCm',data=iris)
plt.subplot(2,2,3)
sns.boxplot(x='Species',y='SepalLengthCm',data=iris)
plt.subplot(2,2,4)
sns.boxplot(x='Species',y='SepalWidthCm',data=iris)
plt.show()
```



Latihan (7)

buat visualisasi box plot distribusi setiap kolom feature

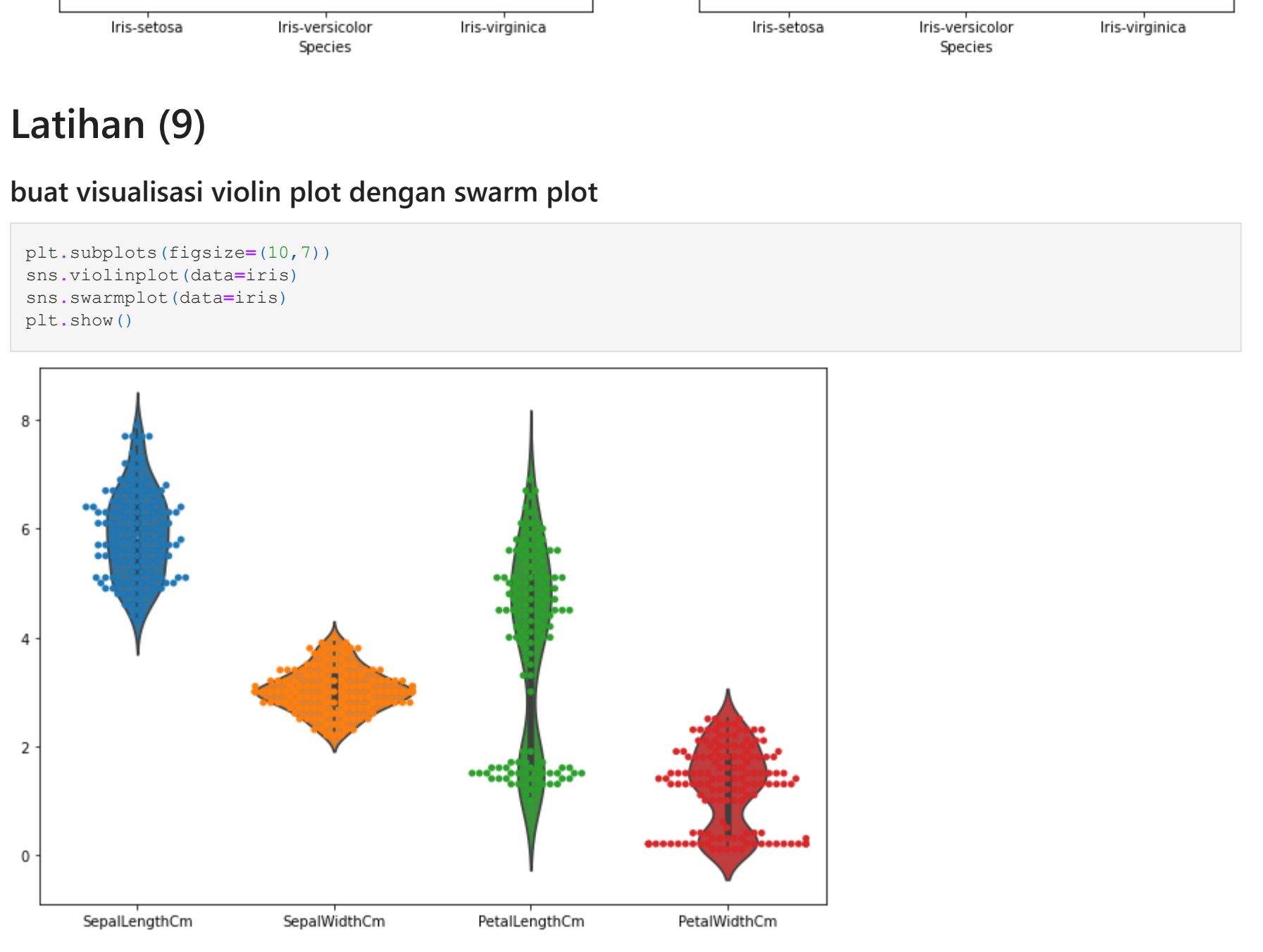
In [13]:
```
plt.subplots(figsize=(10,7))
sns.boxplot(data=iris).set_title("Distribution of Sepal_length, Sepal_width, petal_length and petal_width of 3
plt.show()
```



Latihan (8)

buat visualisasi violin plot setiap kolom feature

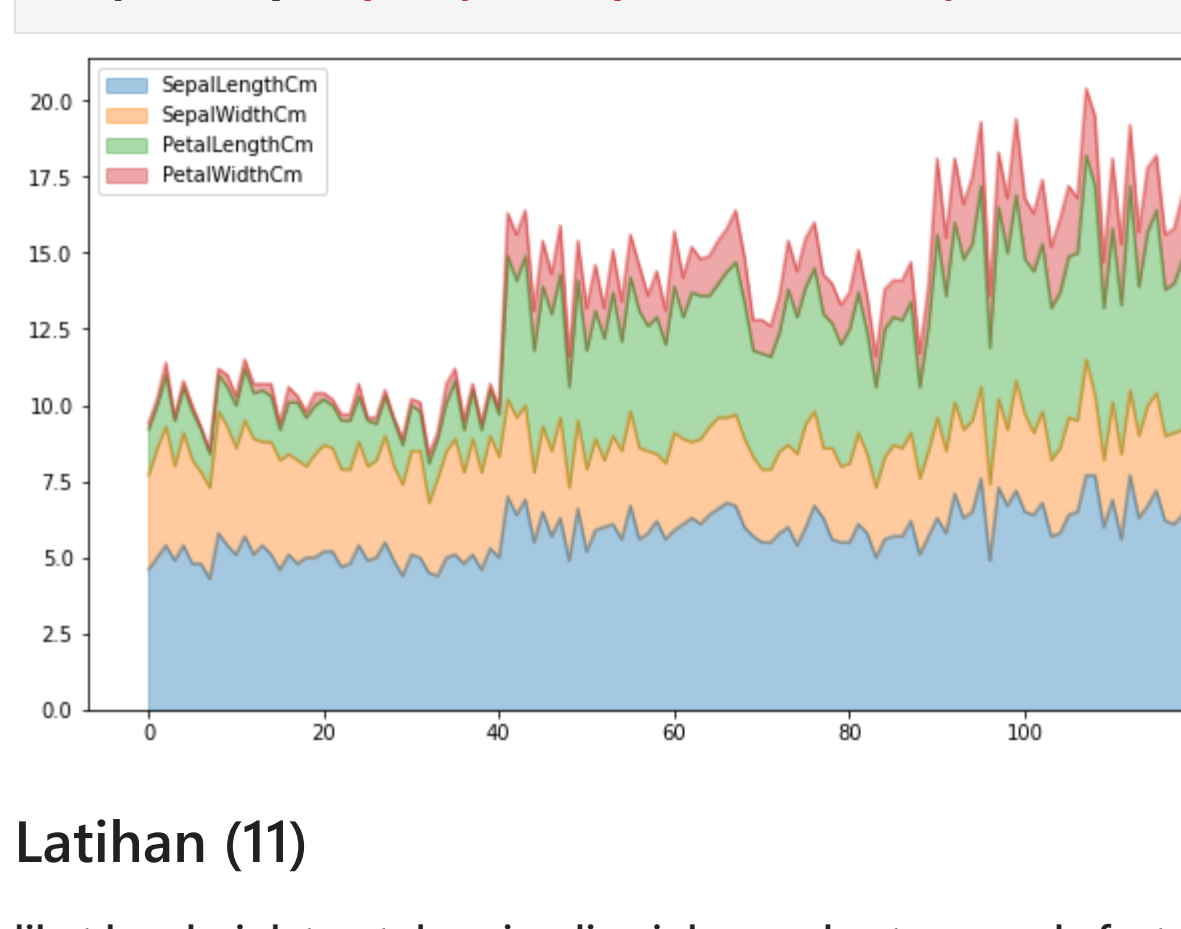
In [14]:
```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.violinplot(x='Species',y='PetalLengthCm',data=iris)
plt.subplot(2,2,2)
sns.violinplot(x='Species',y='PetalWidthCm',data=iris)
plt.subplot(2,2,3)
sns.violinplot(x='Species',y='SepalLengthCm',data=iris)
plt.subplot(2,2,4)
sns.violinplot(x='Species',y='SepalWidthCm',data=iris)
plt.show()
```



Latihan (9)

buat visualisasi violin plot dengan swarm plot

In [15]:
```
plt.subplots(figsize=(10,7))
sns.violinplot(data=iris)
sns.swarmplot(data=iris)
plt.show()
```



Latihan (10)

buat visualisasi area plot pada setiap feature kolom

In [16]:
```
iris.plot.area(y=['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm'],alpha=0.4,figsize=(12, 6));
```

Latihan (11)

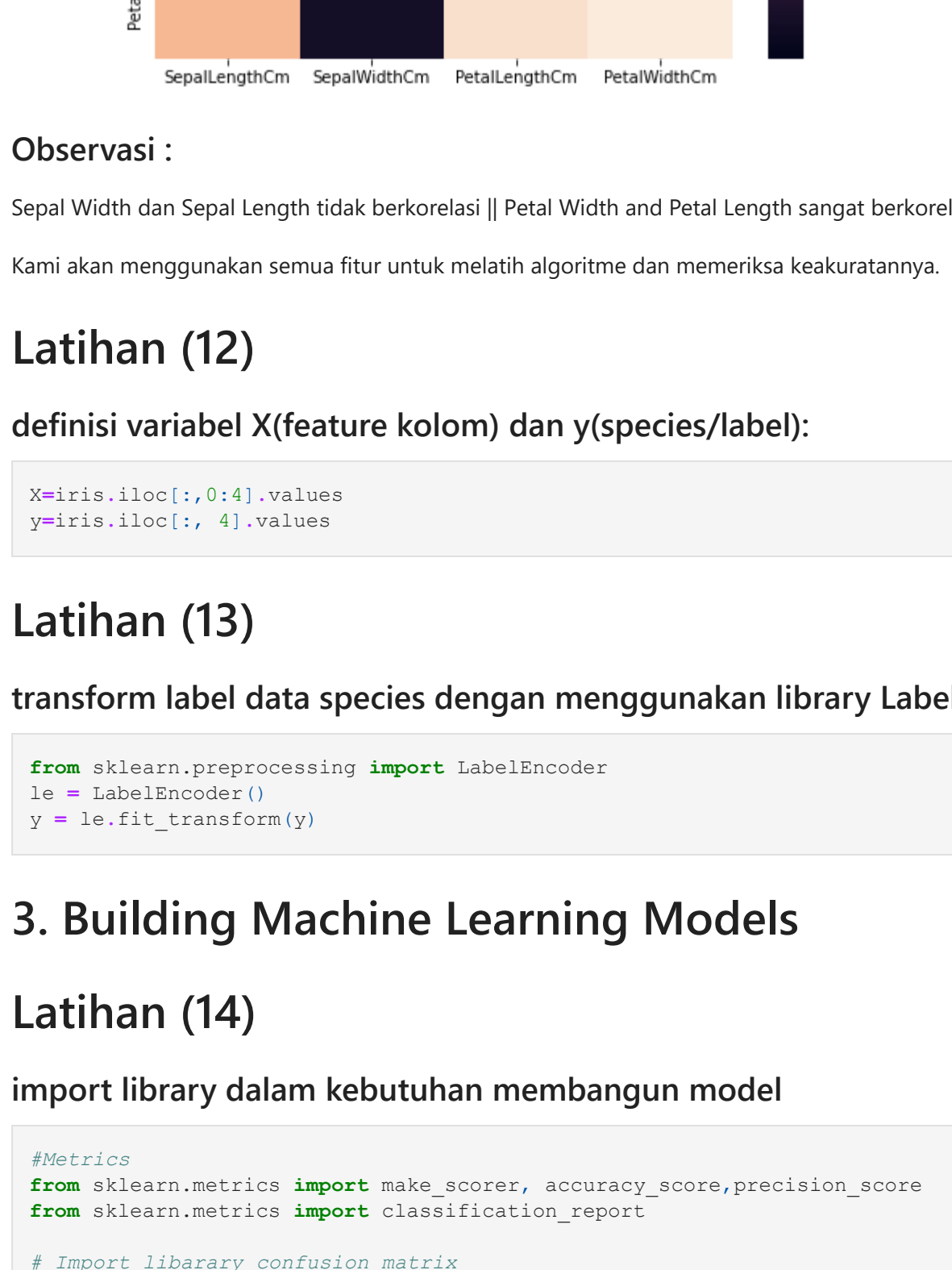
lihat korelasi dataset dan visualisasi dengan heatmap pada feature kolom

In [17]:
```
# lihat korelasi dengan function corr()
iris.corr()
```

Out[17]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.095146	0.871080	0.810336
SepalWidthCm	-0.095146	1.000000	-0.388439	-0.317683
PetalLengthCm	0.871080	-0.388439	1.000000	0.958657
PetalWidthCm	0.810336	-0.317683	0.958657	1.000000

In [18]:
```
# lihat korelasi dengan visualisasi heatmap
plt.figure(figsize = (8,6))
sns.heatmap(iris.corr(),annot=True,fmt="f").set_title("Corelation of attributes (petal length,width and sepal l
plt.show()
```



Observasi :

Sepal Width dan Sepal Length tidak berkorelasi || Petal Width and Petal Length sangat berkorelasi

Kami akan menggunakan semua fitur untuk melatih algoritme dan memeriksa keakuratannya.

Latihan (12)

definisi variabel X(feature kolom) dan y(species/label):

In [19]:
```
X=iris.iloc[:,0:4].values
y=iris.iloc[:,4].values
```

Latihan (13)

transform label data species dengan menggunakan library LabelEncoder

In [20]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
```

3. Building Machine Learning Models

Latihan (14)

import library dalam kebutuhan membangun model

In [21]:
```
#Metrics
from sklearn.metrics import make_scorer, accuracy_score,precision_score
from sklearn.metrics import classification_report

# Import library confusion matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score ,precision_score,recall_score,f1_score

#Model Select
from sklearn.model_selection import KFold,train_test_split,cross_val_score
from sklearn.model_selection import train_test_split

# Import library Logistic Regression
from sklearn.linear_model import LogisticRegression

from sklearn import linear_model
from sklearn.linear_model import SGDClassifier

# Import library KNN
from sklearn.neighbors import KNeighborsClassifier

# Import library Support Vector Machines dan linier Support Vector Machines
from sklearn.svm import SVC, LinearSVC
from sklearn import svm

# Import library Gaussian Naive Bayes
from sklearn import naive_bayes
```

Latihan (15)

split data train dan test dengan function train_test_split() dengan train_size=0.7, test_size=0.3 dan random_state=0

In [22]:
```
#Train and Test split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=0)
```

Sekarang kita akan melatih beberapa model Machine Learning dan membandingkan hasilnya. Perhatikan bahwa karena set data tidak memberikan label untuk set pengujiannya, kita perlu menggunakan prediksi pada set pelatihan untuk membandingkan algoritme satu sama lain.

Bangun model LogisticRegression dan akurasi nya

In [23]:
```
logreg = LogisticRegression(solver= 'lbfgs',max_iter=400)
logreg.fit(X_train, y_train)
Y_pred = logreg.predict(X_test)
accuracy_lr=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_log = round(logreg.score(X_train, y_train) * 100, 2)

cm = confusion_matrix(y_test, Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall =  recall_score(y_test, Y_pred,average='micro')
f1 = f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for Logistic Regression\n',cm)
print('accuracy_Logistic Regression : %.3f' %accuracy)
print('precision_Logistic Regression : %.3f' %precision)
print('recall_Logistic Regression: %.3f' %recall)
print('f1-score_Logistic Regression : %.3f' %f1)
```

```
Confusion matrix for Logistic Regression
 [[12  0  0]
 [ 0 14  1]
 [ 0  1 14]]
accuracy_Logistic Regression : 0.952
precision_Logistic Regression : 0.952
recall_Logistic Regression: 0.952
f1-score_Logistic Regression : 0.952
```

Latihan (16)

Bangun model KNN dan akurasi nya

In [24]:
```
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, y_train)
Y_pred = knn.predict(X_test)
accuracy_knn=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_knn = round(knn.score(X_train, y_train) * 100, 2)

cm = confusion_matrix(y_test, Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall =  recall_score(y_test, Y_pred,average='micro')
f1 = f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for Logistic Regression\n',cm)
print('accuracy_Logistic Regression : %.3f' %accuracy)
print('precision_Logistic Regression : %.3f' %precision)
print('recall_Logistic Regression: %.3f' %recall)
print('f1-score_Logistic Regression : %.3f' %f1)
```

```
Confusion matrix for Logistic Regression
 [[12  0  0]
 [ 0 14  1]
 [ 0  2 13]]
accuracy_Logistic Regression : 0.929
precision_Logistic Regression : 0.929
recall_Logistic Regression: 0.929
f1-score_Logistic Regression : 0.929
```

Mari kita periksa akurasi untuk berbagai nilai n untuk Model KNN

In [25]:
```
plt.subplots(figsize=(20,5))
a_index=list(range(1,50))
a=pd.Series()
x=range(1,50)
#x=[1,2,3,4,5,6,7,8,9,10]
for i in list(range(1,50)):
    model=KNeighborsClassifier(n_neighbors=i)
    model.fit(X_train, y_train)
    prediction=model.predict(X_test)
    a=a.append(pd.Series(accuracy_score(y_test,prediction)))
plt.plot(a_index, a,marker="*")
plt.xticks(x)
plt.show()
```

Di atas adalah grafik yang menunjukkan akurasi untuk model KNN menggunakan nilai n yang berbeda.

Latihan (17)

Bangun model gaussian Naive Bayes dan akurasi nya

In [26]:
```
gaussian = naive_bayes.GaussianNB()
gaussian.fit(X_train, y_train)
Y_pred = gaussian.predict(X_test)
accuracy_nb=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_gaussian = round(gaussian.score(X_train, y_train) * 100, 2)

cm = confusion_matrix(y_test, Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall =  recall_score(y_test, Y_pred,average='micro')
f1 = f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for Logistic Regression\n',cm)
print('accuracy_Logistic Regression : %.3f' %accuracy)
print('precision_Logistic Regression : %.3f' %precision)
print('recall_Logistic Regression: %.3f' %recall)
print('f1-score_Logistic Regression : %.3f' %f1)
```

```
Confusion matrix for Logistic Regression
 [[12  0  0]
 [ 0 14  1]
 [ 0  2 13]]
accuracy_Logistic Regression : 0.929
precision_Logistic Regression : 0.929
recall_Logistic Regression: 0.929
f1-score_Logistic Regression : 0.929
```

Latihan (18)

Bangun model gaussian Naive Bayes dan akurasi nya

Latihan (19)

Bangun model Linear Support Vector Machines dan akurasi nya

In [27]:
```
linear_svc = LinearSVC(max_iter=4000)
linear_svc.fit(X_train, y_train)
Y_pred = linear_svc.predict(X_test)
accuracy_svc=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_linear_svc = round(linear_svc.score(X_train, y_train) * 100, 2)

cm = confusion_matrix(y_test, Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall =  recall_score(y_test, Y_pred,average='micro')
f1 = f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for Logistic Regression\n',cm)
print('accuracy_Logistic Regression : %.3f' %accuracy)
print('precision_Logistic Regression : %.3f' %precision)
print('recall_Logistic Regression: %.3f' %recall)
print('f1-score_Logistic Regression : %.3f' %f1)
```

```
Confusion matrix for Logistic Regression
 [[12  0  0]
 [ 0 13  2]
 [ 0  2 13]]
accuracy_Logistic Regression : 0.905
precision_Logistic Regression : 0.905
recall_Logistic Regression: 0.905
f1-score_Logistic Regression : 0.905
```

Latihan (20)

Model mana yang terbaik ?

In [28]:
```
results = pd.DataFrame({
    'Model': [ 'KNN',
              'Logistic Regression',
              'Naive Bayes',
              ' Support Vector Machine'],
    'Score': [ acc_knn,
              acc_log,
              acc_gaussian,
              acc_linear_svc],
    "Accuracy_score":[accuracy_knn,
                      accuracy_lr,
                      accuracy_nb,
                      accuracy_svc
                     ]})
result_df = results.sort_values(by='Accuracy_score', ascending=False)
result_df = result_df.reset_index(drop=True)
result_df.head(9)
```

Out[28]:

	Model	Score	Accuracy_score
0	KNN	96.94	92.86
1	Logistic Regression	96.94	95.24
2	Naive Bayes	96.94	92.86
3	Support Vector Machine	95.92	90.48

Seperti yang kita lihat Model terbaik diberikan oleh Logistic Regression (Akurasi 95%).

Visualisasikan Hasil Akurasi

In [29]:
```
plt.figure(figsize=(12,8))
ax=sns.barplot(x='Model',y="Accuracy_score",data=result_df)
labels = (result_df["Accuracy_score"])
# add result numbers on barchart
for i, v in enumerate(labels):
    ax.text(i, v+1, str(v), horizontalalignment = 'center', size = 15, color = 'black')
```

Hasil Observasi

Hal ini seperti yang dharapkan dapat terlihat pada hasinya di atas bahwa kondisi antara dapat Model dan Dapat tunjuk sangat sesuai dengan model terbaik dikarena oleh Logistic Regression dengan akurasi 45%

Thank you!!