1. Get the data.

In [337]:
```
import pandas as pd
import os
import csv
import numpy as np
import matplotlib.pyplot as plot
%matplotlib inline
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import Imputer
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
from sklearn.model_selection import train_test_split

def load_data():
    path = os.path.join("/home/raj/Downloads/iris-data.csv")
    mycsv= pd.read_csv(path)
    return mycsv

iris = load_data()
# iris.tail()
```

3)a. take a look at the data structure by calling the method .head() the date is divided into the 5 columns sepal,petal length and width with calss of flowers.

In [338]:
```
iris.head()
```

Out[338]:

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

3) b. Get the quick description of data, there are missing values in the columns(petal_with_cm) with NAN values in 7,8,9,10,11 rows

In [339]:
```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal_length_cm    150 non-null float64
sepal_width_cm     150 non-null float64
petal_length_cm    150 non-null float64
petal_width_cm     145 non-null float64
class              150 non-null object
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
```

3) c. Summary of the numericals. using method describe().

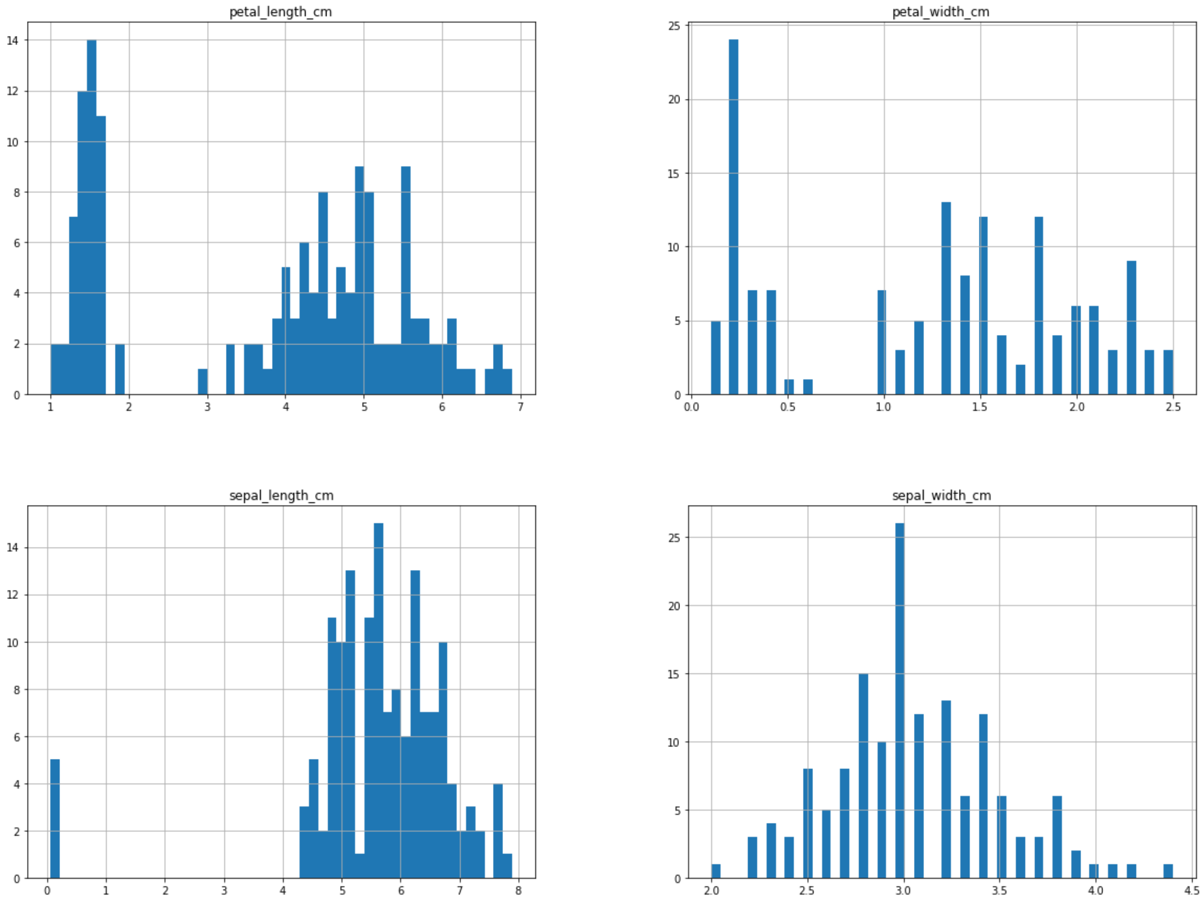
In [340]:
```
iris.describe()
```

Out[340]:

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm
count	150.000000	150.000000	150.000000	145.000000
mean	5.644627	3.054667	3.758667	1.236552
std	1.312781	0.433123	1.764420	0.755058
min	0.055000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.400000
50%	5.700000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

3) d. histogram plot

In [341]:
```
iris.hist(bins=50, figsize=(20,15))
plot.show()
```

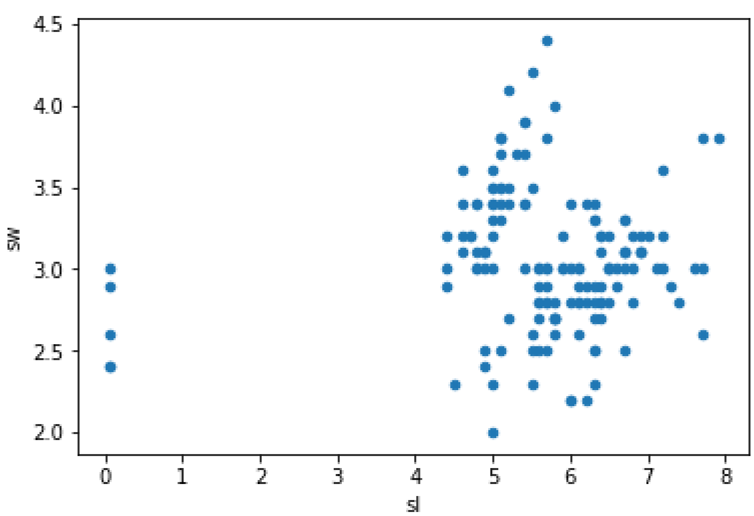


4) Discover and visualize

4)a. scatter matrix

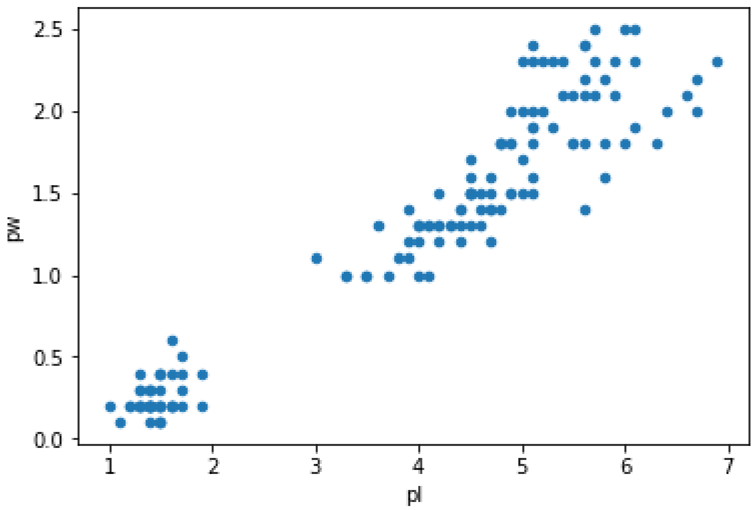
In [342]:
```
iris.columns = ['sl','sw','pl','pw','c']
iris.plot(kind="scatter", x="sl", y="sw")
```

Out[342]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f072c7d3390>
```



In [343]:
```
iris.plot(kind="scatter", x="pl", y="pw")
```

Out[343]: <matplotlib.axes._subplots.AxesSubplot at 0x7f072c5564e0>



4) b. correlation matrix

In [344]:
```
corr_matrix = iris.corr()
corr_matrix["pl"].sort_values(ascending=False)
```

Out[344]:
```
pl    1.000000
pw    0.958934
sl    0.489083
sw   -0.419796
Name: pl, dtype: float64
```

In [345]:
```
corr_matrix = iris.corr()
corr_matrix["pw"].sort_values(ascending=False)
```

Out[345]:
```
pw    1.000000
pl    0.958934
sl    0.469734
sw   -0.348464
Name: pw, dtype: float64
```

In [346]:
```
corr_matrix = iris.corr()
corr_matrix["sl"].sort_values(ascending=False)
```

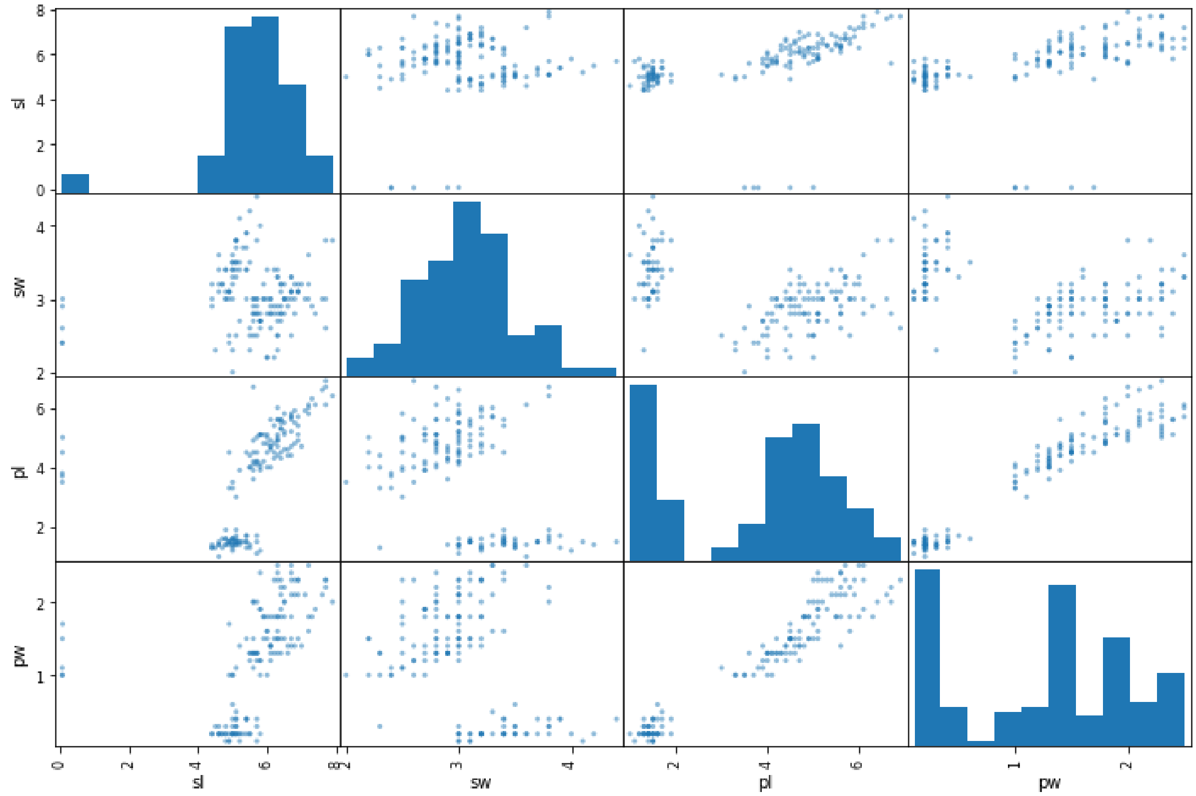
Out[346]:
```
sl    1.000000
pl    0.489083
pw    0.469734
sw    0.066091
Name: sl, dtype: float64
```

In [347]:
```
corr_matrix = iris.corr()
corr_matrix["sw"].sort_values(ascending=False)
```

Out[347]:
```
sw    1.000000
sl    0.066091
pw   -0.348464
pl   -0.419796
Name: sw, dtype: float64
```

In [348]:
```
attributes = ["sl", "sw", "pl","pw"]
scatter_matrix(iris[attributes], figsize=(12, 8))
```

Out[348]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f072c2a7400
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f072c1db400
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f072c38c908
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f072c16f240
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f072c13a208
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f072c094be0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f072c05f588
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f072c08d518
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f072bffa278
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f072c0051d0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f072bf13390
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f072beda4a8
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f072bead4e0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f072be784a8
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f072be50ef0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f072bd9f898
>]], dtype=object)
```



5) Data cleaning

5) a. Drop the data points with NA

In [349]:
```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sl    150 non-null float64
sw    150 non-null float64
pl    150 non-null float64
pw    145 non-null float64
c     150 non-null object
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
```

In [350]:
```
iris = iris.dropna(subset=["pw"])
```

In [351]:
```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 145 entries, 0 to 149
Data columns (total 5 columns):
sl    145 non-null float64
sw    145 non-null float64
pl    145 non-null float64
pw    145 non-null float64
c     145 non-null object
dtypes: float64(4), object(1)
memory usage: 6.8+ KB
```

The data that contains the null values are dropped and can be seen from the above info and used the option 1 mentioned in the text book.

5) b. Tidy Up the data

In [352]:
```
iris.loc[iris['c'] == 'versicolor', 'c'] = 'Iris-versicolor'
iris.loc[iris['c'] == 'Iris-setossa', 'c'] = 'Iris-setosa'
iris['c'].unique()
```

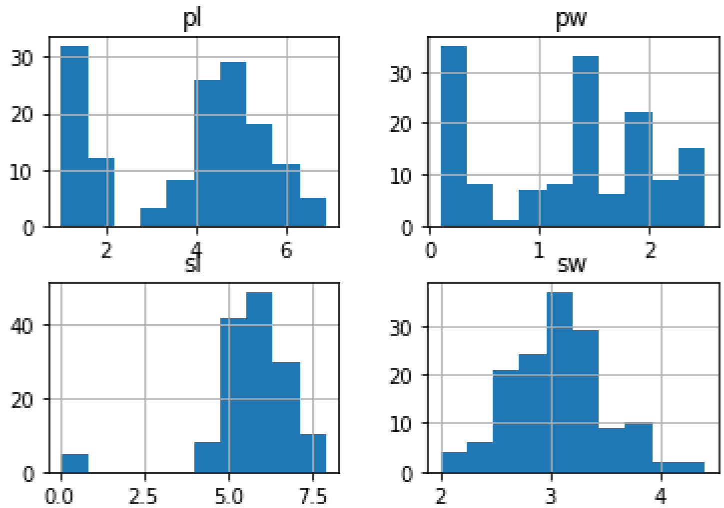
Out[352]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=objec
t)
```

5) c. Removing the outliers and drop the iris-setosa with SW < 2.5

In [353]:
```
i = iris[((iris.c =='Iris-setosa') & (iris.sw < 2.5))].index
iris = iris.drop(i)
```

In [354]:
```
iris.hist()
```

Out[354]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f072be6e8d0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f072c45cdd8
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f072c429898
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f072c3755c0
>]], dtype=object)
```



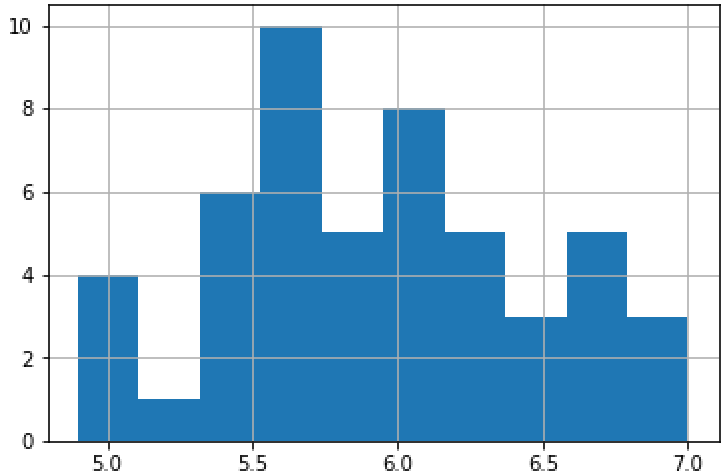
In [355]:
```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 144 entries, 0 to 149
Data columns (total 5 columns):
sl    144 non-null float64
sw    144 non-null float64
pl    144 non-null float64
pw    144 non-null float64
c     144 non-null object
dtypes: float64(4), object(1)
memory usage: 6.8+ KB
```

5) d. converting the data from meters into cm in Iris-versicolor

In [356]:
```
iris.loc[(iris['c'] == 'Iris-versicolor') & (iris['sl'] < 1.0), 'sl'] *=
iris.loc[iris['c'] == 'Iris-versicolor', 'sl'].hist()
```

Out[356]: <matplotlib.axes._subplots.AxesSubplot at 0x7f072c5f9e48>



5) e. handle catergorical values

In [357]:
```
encoder = LabelEncoder()
iris_cat = iris["c"]
iris_cat_encoded = encoder.fit_transform(iris_cat)
iris_cat_encoded
```

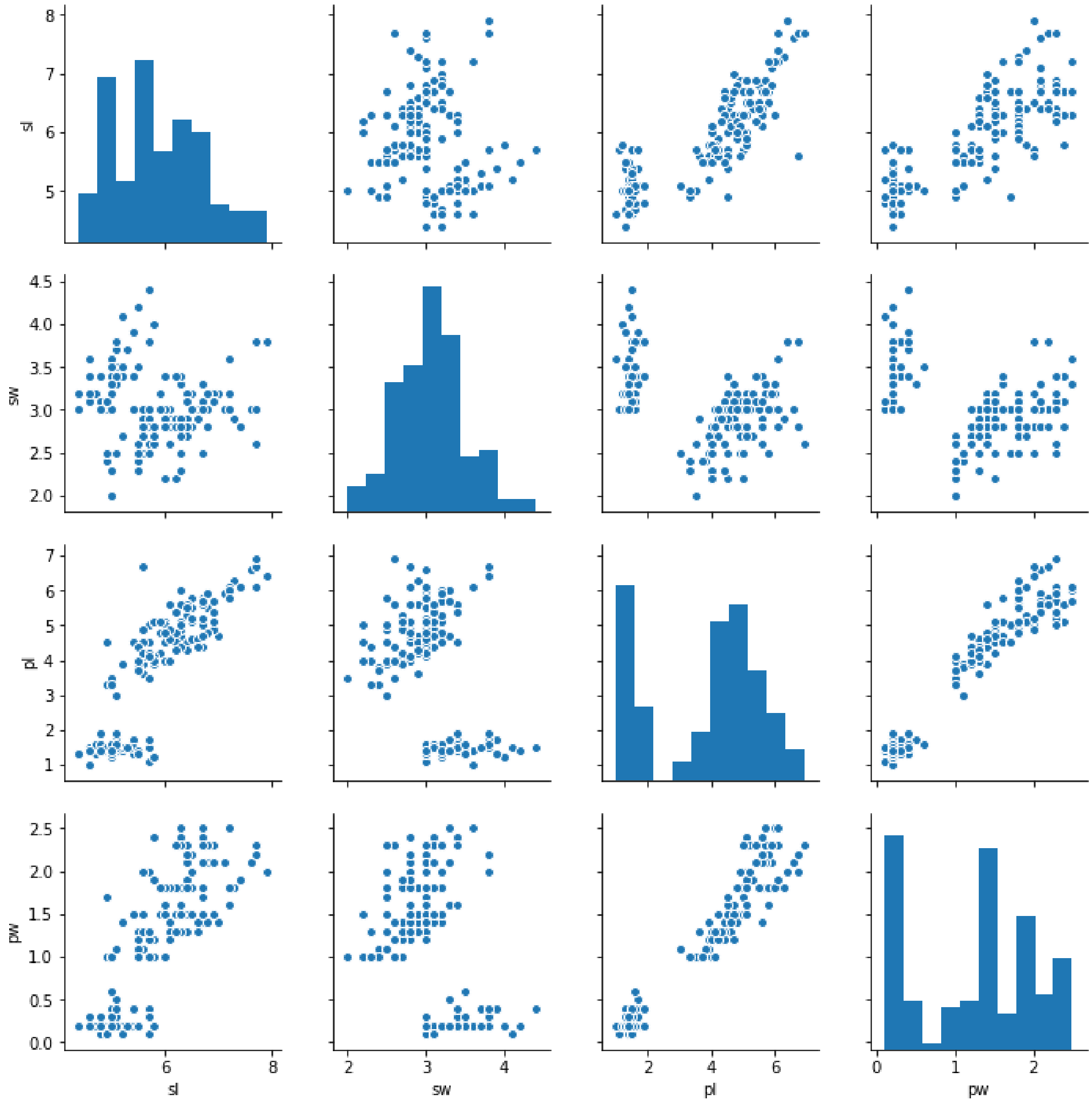
Out[357]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1,
       1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
 2, 2,
       2, 2, 2, 2, 2, 2])
```

In [358]:
```
encoder.classes_
```

Out[358]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=objec
t)
```

In [359]:
```
iris.to_csv('iris-data-clean.csv', index=False)
iris_clean = pd.read_csv('iris-data-clean.csv')
sns.pairplot(iris_clean)
```

Out[359]:
```
<seaborn.axisgrid.PairGrid at 0x7f072d8573c8>
```



6) Using Percepron.py learning algorithm.

6) a. modifying the data such that the data points with Iris-setosa is given the label 1 and the rest as -1

In [361]:
```
import pandas as pd
import matplotlib.pyplot as plot
from pandas.plotting import scatter_matrix
from sklearn.model_selection import train_test_split
import numpy as np

def load_data():
    path = os.path.join("/home/raj/Downloads/iris-data-clean.csv")
    mycsv= pd.read_csv(path)
    return mycsv

iris2 = load_data()

iris2['c'].replace(["Iris-setosa", "Iris-versicolor", "Iris-virginica"],
-1, -1], inplace=True)
```

In [362]:
```
iris2.head(5)
```

Out[362]:

	sl	sw	pl	pw	c
0	5.1	3.5	1.4	0.2	1
1	4.9	3.0	1.4	0.2	1
2	4.7	3.2	1.3	0.2	1
3	4.6	3.1	1.5	0.2	1
4	5.0	3.6	1.4	0.2	1

Above is the example for the label matrix that has 1 for iris- setosa and below is the label for iris-vergicia.

In [363]:
```
iris2.tail()
```

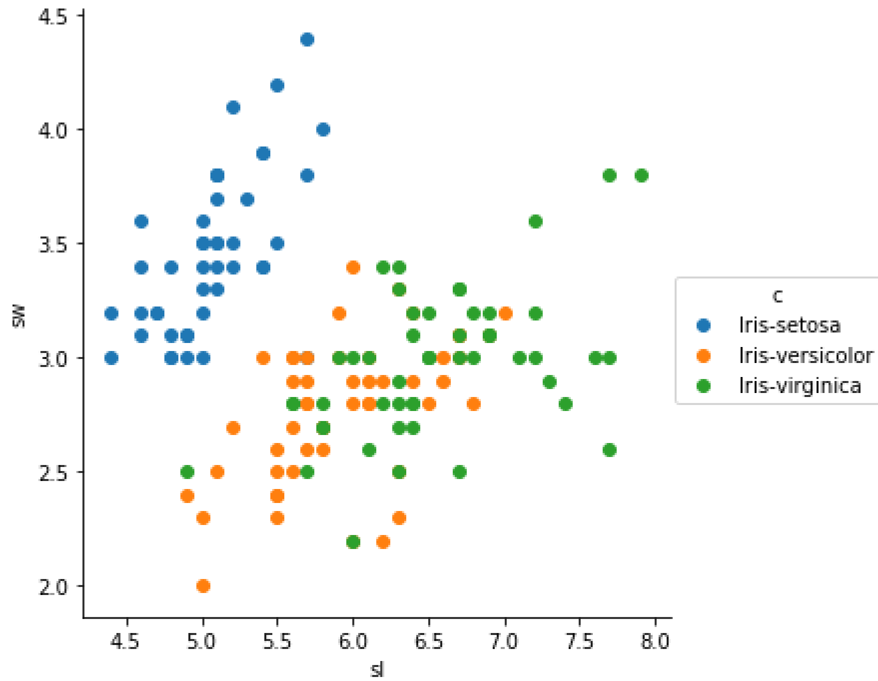
Out[363]:

	sl	sw	pl	pw	c
139	6.7	3.0	5.2	2.3	-1
140	6.3	2.5	5.0	2.3	-1
141	6.5	3.0	5.2	2.0	-1
142	6.2	3.4	5.4	2.3	-1
143	5.9	3.0	5.1	1.8	-1

6)b. is the data linearly separable ? yes it can be shown from the below plot. as the data is separable the percepton will work, by drawing a straight line in between we can say the data is linearly seperable.

In [364]:
```
iris2['c'].value_counts()
sns.FacetGrid(iris, hue='c', size=5)\
    .map(plot.scatter, "sl", "sw")\
    .add_legend()
```

Out[364]: <seaborn.axisgrid.FacetGrid at 0x7f072b7cae80>



6) c. explain the percerptron.py

In perceptron.py file first numpy is imported then class Perceptron is made. Then in constructor for rate and niter is madewith o.o1 and 10 respectively. Then in function fit training data is pased through X,y and weights are defined.In the next line cotains misclassification number identification. To do this for loop is used using delta_w and creating equation,err +=int(delta_w!=0.0). Next to calculate the netinput weight list are defined and for predicted value unit step function to find the prediction

6) d. traning data and testing data, theoritically the traning data is 80 % and testing is 20%.

In [365]:
```
train, test= train_test_split(iris2, test_size=0.2, random_state = 200)
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 115 entries, 100 to 26
Data columns (total 5 columns):
sl     115 non-null float64
sw     115 non-null float64
pl     115 non-null float64
pw     115 non-null float64
c      115 non-null int64
dtypes: float64(4), int64(1)
memory usage: 5.4 KB
```

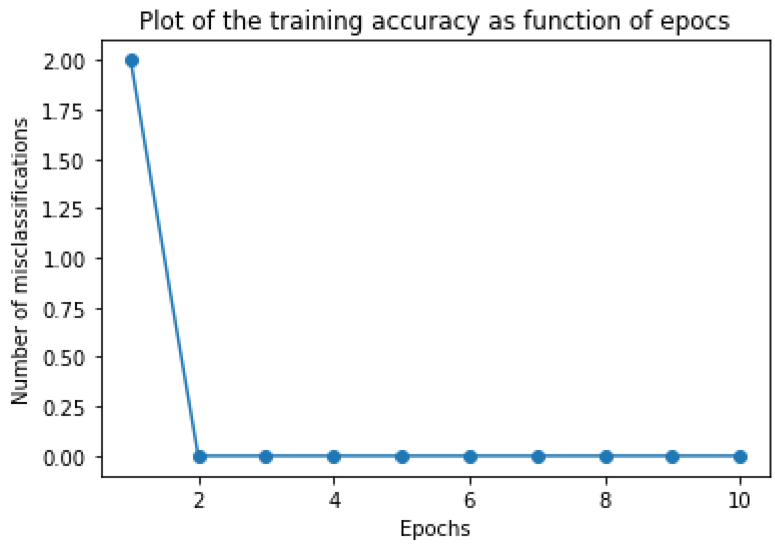
In [366]:
```
X=train.iloc[0:,[0,1,2,3]].values
y=train.iloc[0:,4].values
```

In [367]:
```
X_test=test.iloc[0:,[0,1,2,3]].values
y_test=test.iloc[0:,4].values
```

In [368]:
```
import numpy as np
class Perceptron(object):
     def __init__(self, rate=0.01, niter=10):
          self.rate = rate
          self.niter = niter

     def fit(self, X, y):
          self.weight = np.zeros(1+X.shape[1])
          self.errors = []

          for i in range(self.niter):
              err = 0
              for xi, target in zip(X, y):
                  delta_w = self.rate * (target -self.predict(xi))
                  self.weight[1:] += delta_w * xi
                  self.weight[0] += delta_w
                  err += int(delta_w != 0.0)
              self.errors.append(err)
          return self
     def net_input(self, X):
          return np.dot(X, self.weight[1:]) + self.weight[0]

     def predict(self, X):
          return np.where(self.net_input(X) >= 0.0, 1, -1)

percptn = Perceptron(0.1, 10)
percptn.fit(X, y)
```

Out[368]:
```
<__main__.Perceptron at 0x7f072b6fe438>
```

In [369]:
```
plt.plot(range(1, len(percptn.errors) + 1), percptn.errors, marker='o')
plt.xlabel('Epochs')
plt.ylabel('Number of misclassifications')
plt.title('Plot of the training accuracy as function of epocs')
plt.show()
```



In [370]:
```
X_test=test.iloc[0:,[0,1,2,3]].values
y_test=test.iloc[0:,4].values
percptn.predict(X_test)
```

Out[370]:
```
array([-1, -1, -1, -1,  1, -1,  1,  1, -1, -1,  1,  1, -1, -1, -1,  1,
 -1,
        1,  1, -1, -1, -1, -1,  1, -1, -1, -1, -1,  1])
```

6)f. does the algorithm converege? the algorithm do connverge and theres
 no niter in the code.

6) g. Obtain the accuaracy as athe fucntion epcos and the plot is as above.

6) h.

In [372]:
```
X=test.iloc[0:144,:2].values
y=test.iloc[0:144,4].values
prediction=percptn.predict(X_test)
from sklearn.metrics import accuracy_score
print("accuracy of the perceptron function is: ", accuracy_score(y_test,
```

```
accuracy of the perceptron function is:  1.0
```

In [374]:
```
train, test = train_test_split(iris2, test_size=0.2, random_state = 200)
```

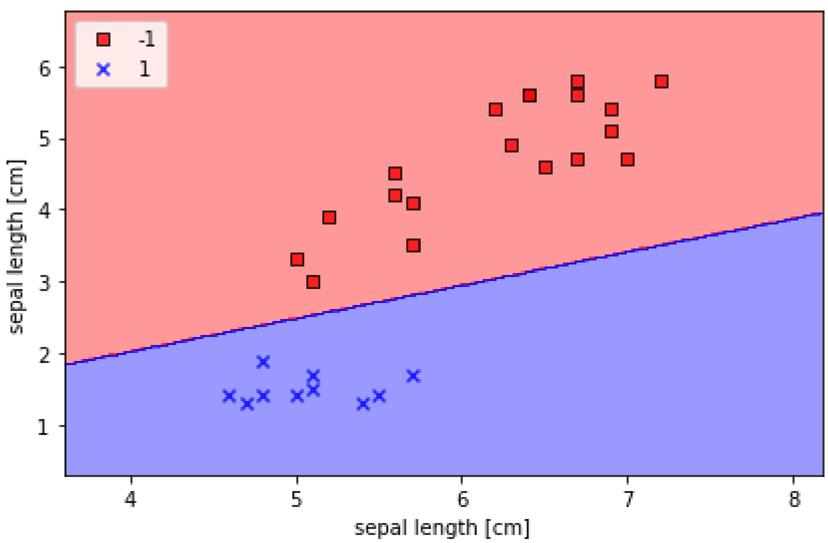
In [375]:
```
X=train.iloc[0:,[0,2]].values
y=train.iloc[0:,4].values
```

In [376]:
```
import numpy as np
class Perceptron(object):
     def __init__(self, rate=0.01, niter=10):
          self.rate = rate
          self.niter = niter

     def fit(self, X, y):
          self.weight = np.zeros(1+X.shape[1])
          self.errors = []

          for i in range(self.niter):
               err = 0
               for xi, target in zip(X, y):
                    delta_w = self.rate * (target -self.predict(xi))
                    self.weight[1:] += delta_w * xi
                    self.weight[0] += delta_w
                    err += int(delta_w != 0.0)
               self.errors.append(err)
          return self
     def net_input(self, X):
          return np.dot(X, self.weight[1:]) + self.weight[0]

     def predict(self, X):
          return np.where(self.net_input(X) >= 0.0, 1, -1)

percptn = Perceptron(0.1, 10)
percptn.fit(X, y)
```

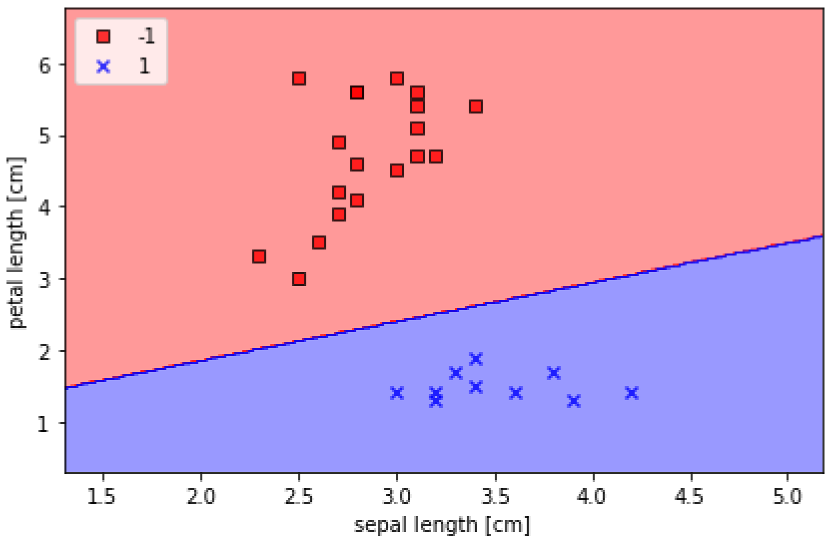
Out[376]:
```
<__main__.Perceptron at 0x7f072b68a2b0>
```

In [377]:
```
X=test.iloc[0:,[0,2]].values
y=test.iloc[0:,4].values
```

In [379]:
```
plotlib.colors import ListedColormap

_decision_regions(x_ts,y_ts,classifier,resolution=0.02):
kers = ('s','x','o','^','v')
ors = ('red','blue','lightgreen','gray','cyan')
p = ListedColormap(colors[:len(np.unique(y))])
min, x1_max = X[:,0].min() - 1, X[:,0].max() +1
min, x2_max = X[:,1].min() - 1, X[:,1].max() +1
, xx2 =np.meshgrid(np.arange(x1_min,x1_max,resolution),
arange(x2_min,x2_max,resolution))
 classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
 Z.reshape(xx1.shape)
.contourf(xx1,xx2,Z,alpha=0.4,cmap=cmap)
.xlim(xx1.min(), xx1.max())
.ylim(xx2.min(), xx2.max())
 idx, cl in enumerate(np.unique(y)):
 plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],alpha=0.8, c=cmap(idx),edge
ision_regions(X, y, classifier=percptn)
el('sepal length [cm]')
el('sepal length [cm]')
nd(loc='upper left')

t_layout()
()
```



In [383]:
```
X=train.iloc[0:,[1,2]].values
yr=train.iloc[0:,4].values
percptn = Perceptron(0.1,10)
percptn.fit(X, y)

X=test.iloc[0:,[1,2]].values
y=test.iloc[0:,4].values

plot_decision_regions(X,y, classifier=percptn)
plt.xlabel('sepal length [cm]')
plt.ylabel('petal length [cm]')
plt.legend(loc='upper left')

plt.tight_layout()
plt.show()
```



In [ ]: