KNN classification implementation ...

..iris dataset..

In [56]:
```
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()
```

In [4]:
```
iris.feature_names
```

Out[4]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [5]:
```
iris.target_names
```

Out[5]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [10]:
```
#Load in dataframe...
df = pd.DataFrame(iris.data,columns=iris.feature_names)
```

In [11]:
```
df
```

Out[11]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
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 [12]:
```
# show first 5 rows..
df.head()
```

Out[12]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
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

In [13]:
```
# show last 5 rows...
df.tail()
```

Out[13]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
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

In [16]:
```
df.shape
```

Out[16]:
```
(150, 4)
```

In [28]:
```
df['target'] = iris.target   # 0 means = setosha
df.head()
```

Out[28]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

In [23]:
```
df[df.target==1].head() #1=versicolor
```

Out[23]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
50	7.0	3.2	4.7	1.4	1
51	6.4	3.2	4.5	1.5	1
52	6.9	3.1	4.9	1.5	1
53	5.5	2.3	4.0	1.3	1
54	6.5	2.8	4.6	1.5	1

In [24]:
```
df[df.target==2].head() #2=virginica
```

Out[24]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
100	6.3	3.3	6.0	2.5	2
101	5.8	2.7	5.1	1.9	2
102	7.1	3.0	5.9	2.1	2
103	6.3	2.9	5.6	1.8	2
104	6.5	3.0	5.8	2.2	2

In [26]:
```
# create a new columns...
df['flower_name'] = df.target.apply(lambda x: iris.target_names[x])
df.head()
```

Out[26]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

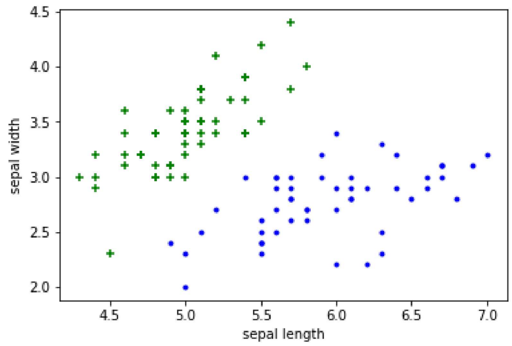
In [29]:
```
## create dataframe for visulazi clusttering...

df0 = df[:50] #setosa

df1 = df[50:100] #versicolor

df2 = df[100:] #virginica
```

In [30]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

sepal length vs sepal width (setosa vs versicolor)

In [33]:
```
plt.xlabel('sepal length')
plt.ylabel('sepal width')
plt.scatter(df0['sepal length (cm)'] , df0['sepal width (cm)'] ,color="green",marker="+")
plt.scatter(df1['sepal length (cm)'] , df1['sepal width (cm)'] ,color="blue",marker=".")
```

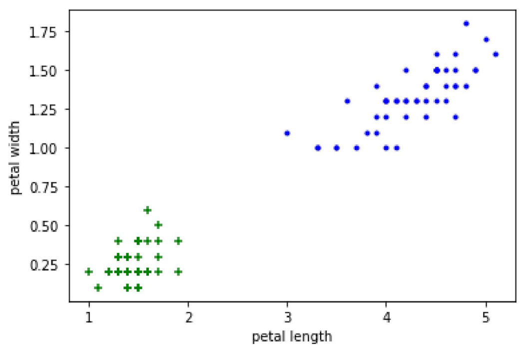
Out[33]: <matplotlib.collections.PathCollection at 0x1cc76d597f0>



petal length vs petal width (setosa vs versicolor)

In [34]:
```
plt.xlabel('petal length')
plt.ylabel('petal width')
plt.scatter(df0['petal length (cm)'] , df0['petal width (cm)'] ,color="green",marker="+")
plt.scatter(df1['petal length (cm)'] , df1['petal width (cm)'] ,color="blue",marker=".")
```

Out[34]: <matplotlib.collections.PathCollection at 0x1cc76d90190>



Train Test split..

In [ ]:
```
from sklearn
```

In [35]:
```
from sklearn.model_selection import train_test_split
```

In [37]:
```
x = df.drop(['target','flower_name'],axis='columns')
y = df.target
```

In [38]:
```
X_train, X_test, y_train, y_test = train_test_split(x,y ,test_size=0.2,random_state=1)
```

In [39]:
```
len(X_train)
```

Out[39]: 120

In [41]:
```
len(y_test)
```

Out[41]: 30

create KNN classifier..

In [44]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=10)
```

In [45]:
```
knn.fit(X_train,y_train)
```

Out[45]:
```
KNeighborsClassifier(n_neighbors=10)
```

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 [48]:
```
knn.score(X_test,y_test)
```

Out[48]:
```
0.9333333333333333
```

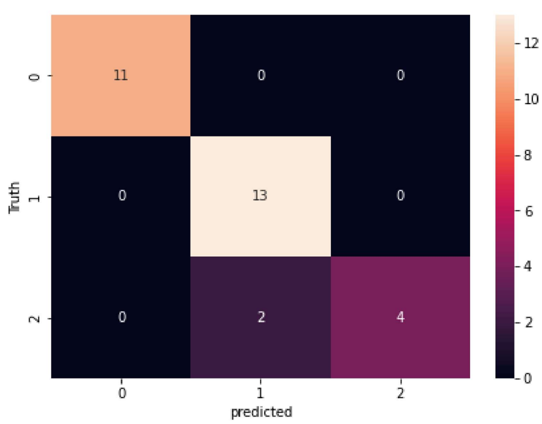
confusion matrix ...

In [52]:
```
# confusion matrix ...
from sklearn.metrics import confusion_matrix
y_pred = knn.predict(X_test)
cm = confusion_matrix(y_test , y_pred)
cm
```

Out[52]:
```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  2,  4]], dtype=int64)
```

In [53]:
```
# visualize cunfusion matrix...
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(7,5))
sn.heatmap(cm , annot=True)
plt.xlabel('predicted')
plt.ylabel('Truth')
```

Out[53]:
```
Text(42.0, 0.5, 'Truth')
```



classification report...

In [55]:
```
# classification report...
from sklearn.metrics import classification_report

print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       0.87      1.00      0.93        13
           2       1.00      0.67      0.80         6

    accuracy                           0.93        30
   macro avg       0.96      0.89      0.91        30
weighted avg       0.94      0.93      0.93        30
```

In [ ]: