Import libraries

In [2]:

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

In [3]:

```
iris = pd.read_csv("F:\dataset\iris.csv")
```

In [4]:

```
iris.head()
```

Out[4]:

	SL	SW	PL	PW	Classification
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

In [5]:

```
iris.shape
```

Out[5]:

```
(150, 5)
```

In [6]:

```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   SL              150 non-null    float64
 1   SW              150 non-null    float64
 2   PL              150 non-null    float64
 3   PW              150 non-null    float64
 4   Classification  150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [7]:

```
iris.isnull()
```

Out[7]:

	SL	SW	PL	PW	Classification
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

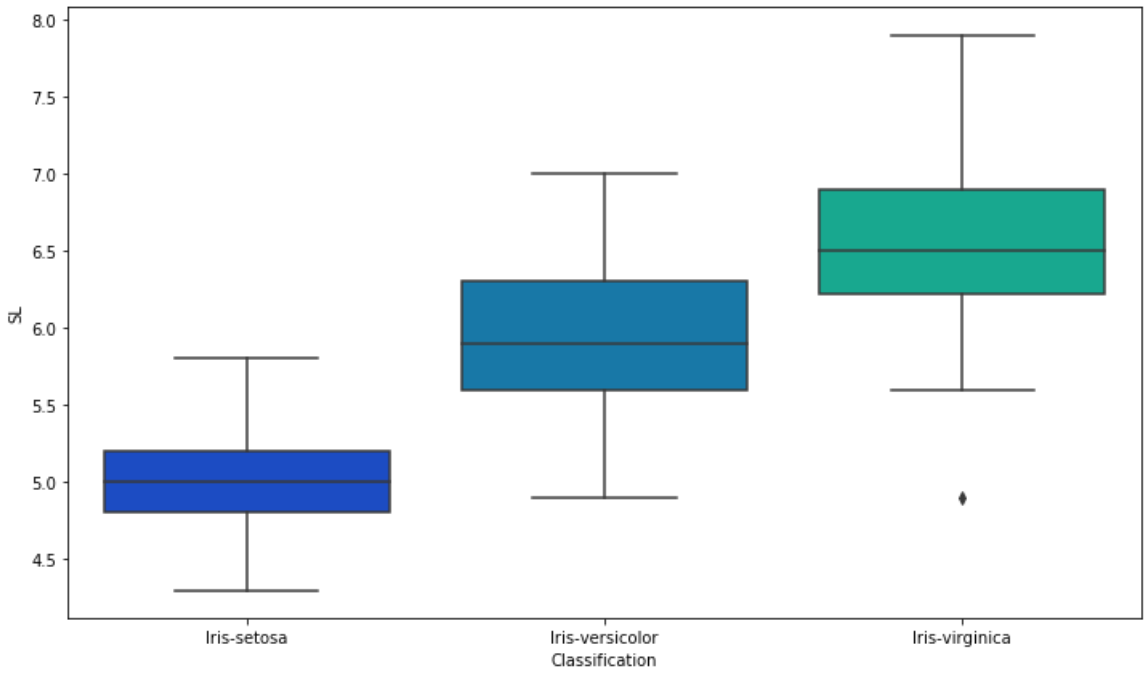
150 rows × 5 columns

In [8]:

```
plt.figure(figsize=(12, 7))
sns.boxplot(x='Classification',y='SL',data=iris,palette='winter')
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x295b69b45c8>
```

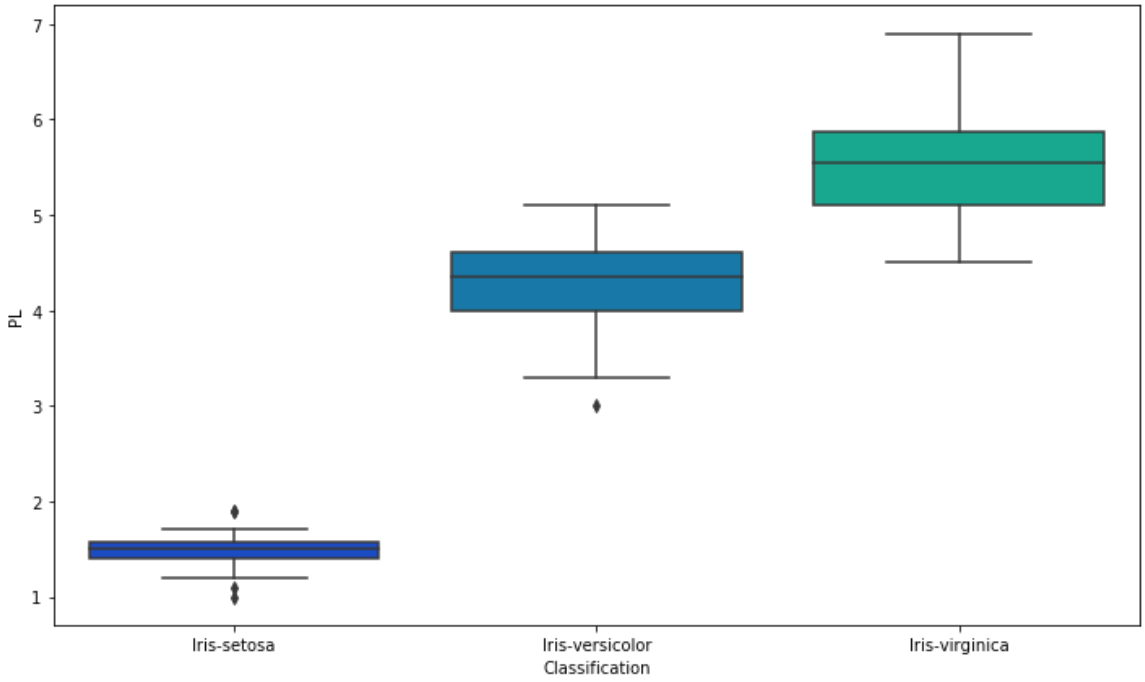


In [9]:

```
plt.figure(figsize=(12, 7))
sns.boxplot(x='Classification',y='PL',data=iris,palette='winter')
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x295b81e5bc8>
```

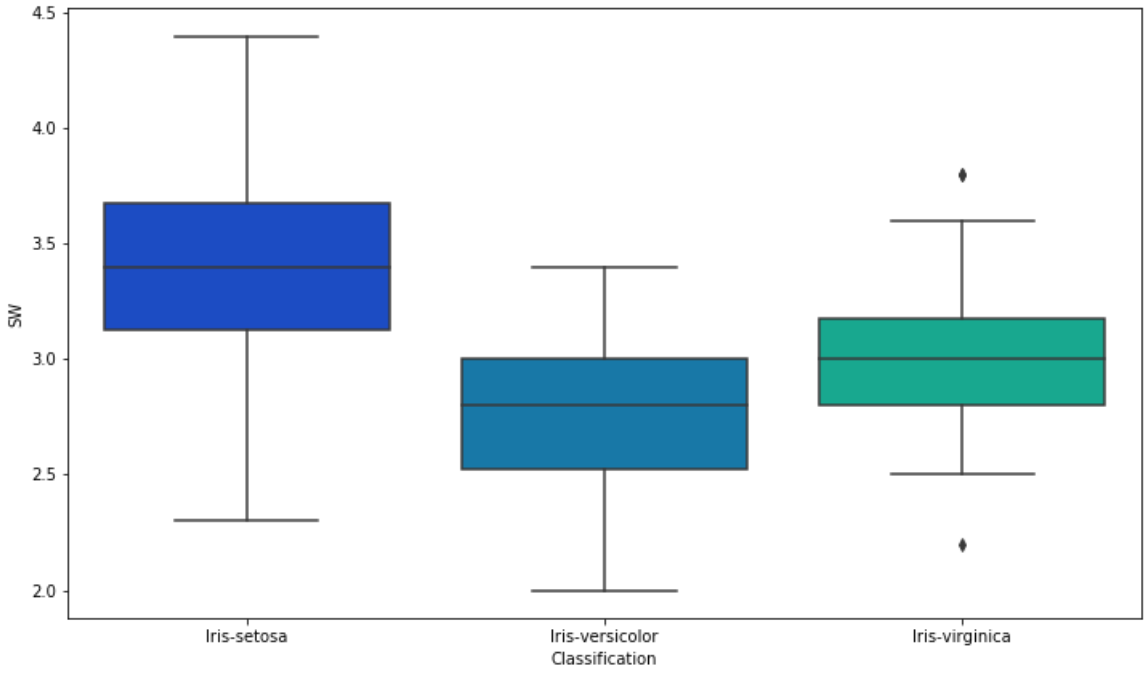


In [10]:

```
plt.figure(figsize=(12, 7))
sns.boxplot(x='Classification',y='SW',data=iris,palette='winter')
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x295b8243288>
```

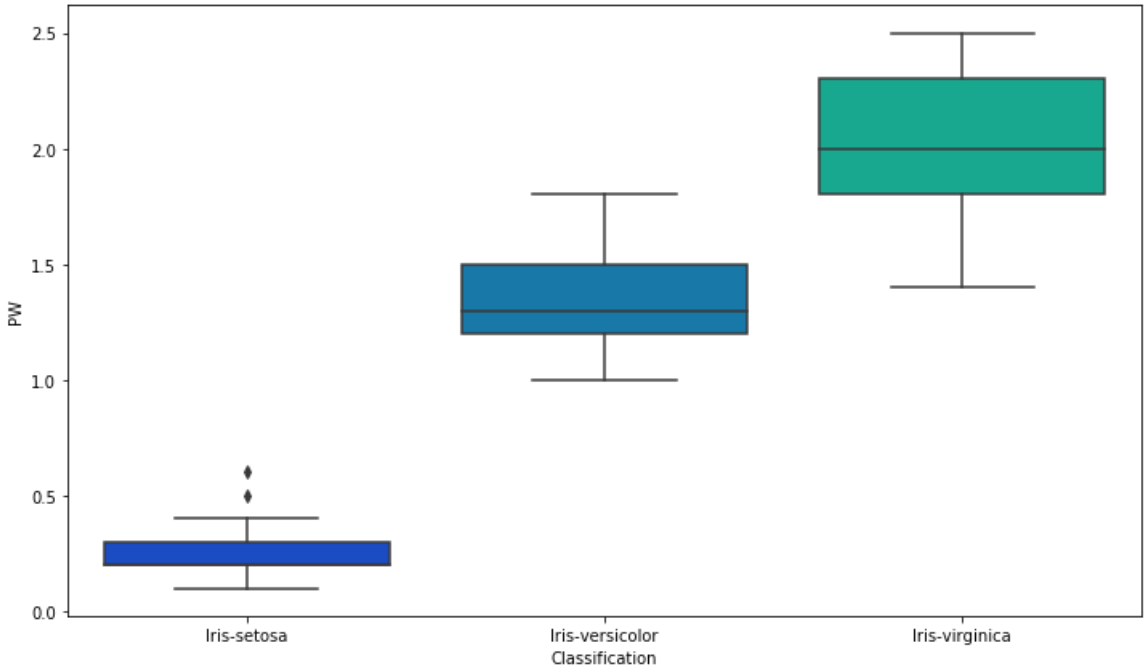


In [11]:

```
plt.figure(figsize=(12, 7))
sns.boxplot(x='Classification',y='PW',data=iris,palette='winter')
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x295b85e0bc8>
```

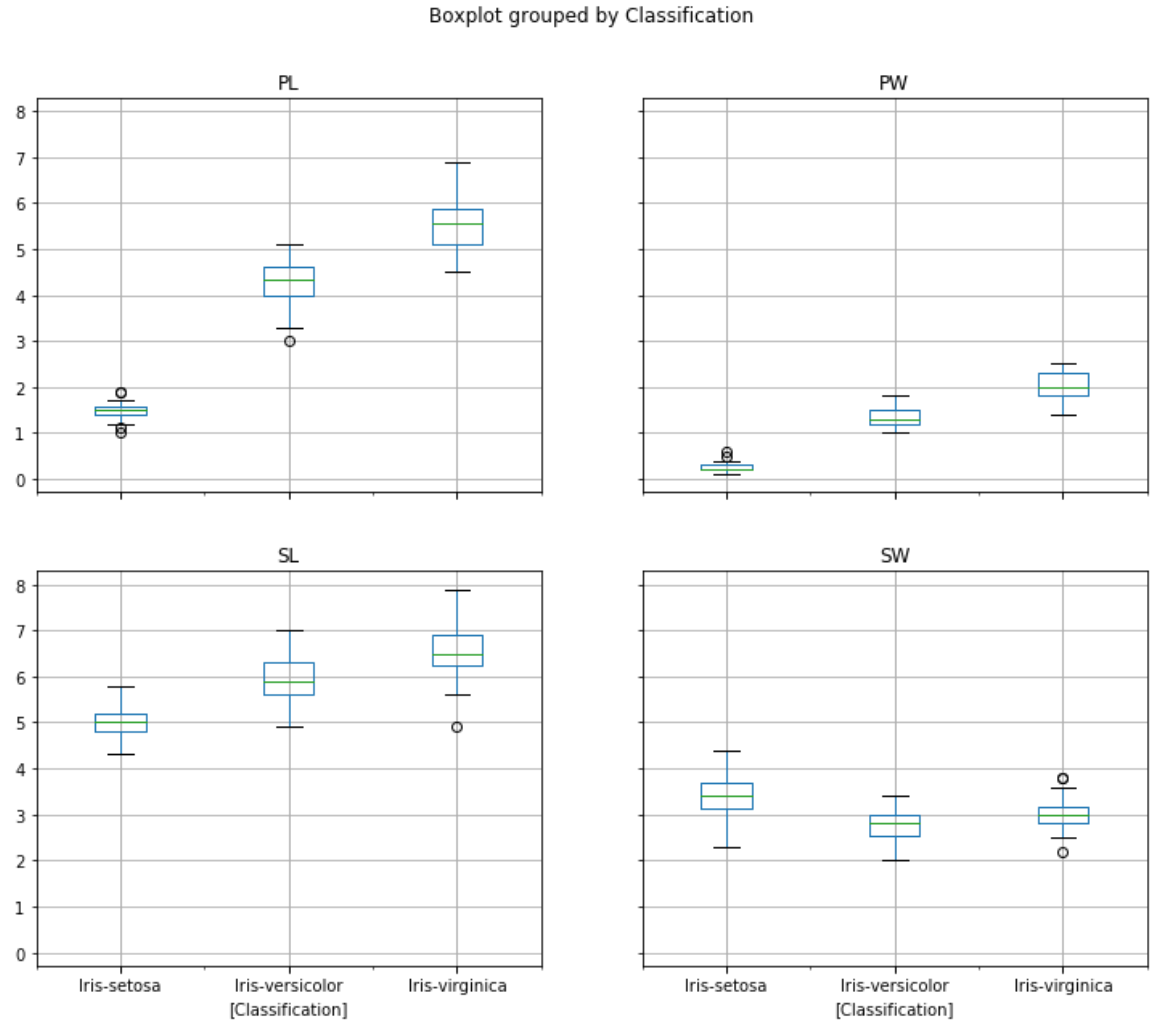


In [12]:

```
iris.boxplot(by="Classification", figsize=(12, 10))
```

Out[12]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000295B86C118
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000295B86EFCC
8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000295B86C14C
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000295B876128
8>]],
      dtype=object)
```



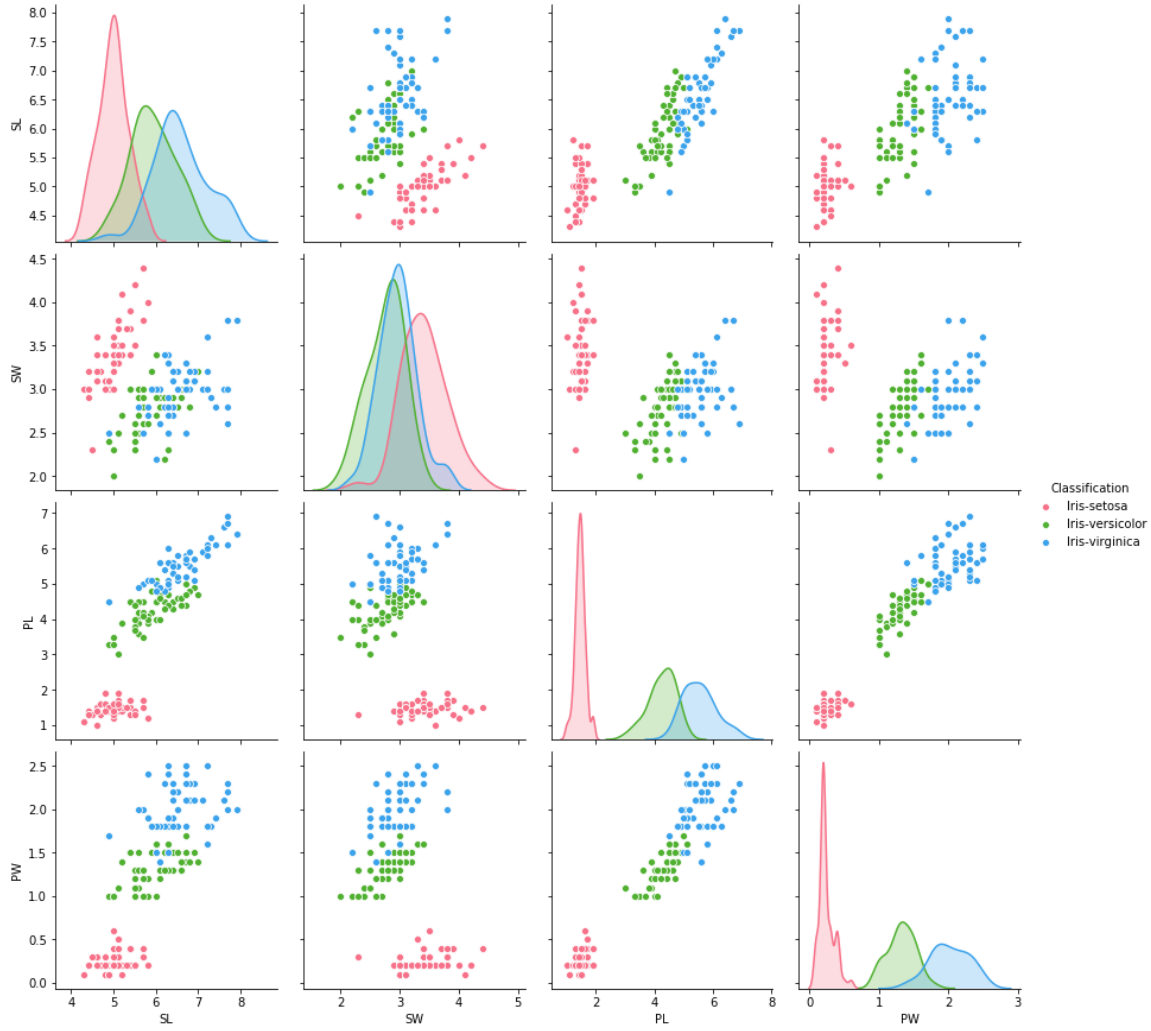
In [13]:

```
sns.pairplot(iris, hue="Classification", palette="husl", size=3, diag_kind="kde")
```

```
C:\Users\Pratima Dhar\anaconda3\lib\site-packages\seaborn\axisgrid.py:207
9: UserWarning: The `size` parameter has been renamed to `height`; please
update your code.
  warnings.warn(msg, UserWarning)
```

Out[13]:

```
<seaborn.axisgrid.PairGrid at 0x295b8c10948>
```



In [14]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(iris.drop('Classification',axis=1),
                                                    iris['Classification'], test_size=
0.20,
                                                    random_state=10)
```

In [15]:

```
print(X_train,y_train)
```

```
      SL   SW   PL   PW
58   6.6  2.9  4.6  1.3
97   6.2  2.9  4.3  1.3
129  7.2  3.0  5.8  1.6
114  5.8  2.8  5.1  2.4
146  6.3  2.5  5.0  1.9
..   ...  ...  ...  ...
113  5.7  2.5  5.0  2.0
64   5.6  2.9  3.6  1.3
15   5.7  4.4  1.5  0.4
125  7.2  3.2  6.0  1.8
9    4.9  3.1  1.5  0.1

[120 rows x 4 columns] 58     Iris-versicolor
97     Iris-versicolor
129     Iris-virginica
114     Iris-virginica
146     Iris-virginica
           ...
113     Iris-virginica
64     Iris-versicolor
15         Iris-setosa
125     Iris-virginica
9          Iris-setosa
Name: Classification, Length: 120, dtype: object
```

In [16]:

```
print(X_test,y_test)
```

```
      SL   SW   PL   PW
87   6.3  2.3  4.4  1.3
111  6.4  2.7  5.3  1.9
10   5.4  3.7  1.5  0.2
91   6.1  3.0  4.6  1.4
49   5.0  3.3  1.4  0.2
60   5.0  2.0  3.5  1.0
72   6.3  2.5  4.9  1.5
67   5.8  2.7  4.1  1.0
39   5.1  3.4  1.5  0.2
55   5.7  2.8  4.5  1.3
66   5.6  3.0  4.5  1.5
142  5.8  2.7  5.1  1.9
53   5.5  2.3  4.0  1.3
1    4.9  3.0  1.4  0.2
19   5.1  3.8  1.5  0.3
112  6.8  3.0  5.5  2.1
85   6.0  3.4  4.5  1.6
38   4.4  3.0  1.3  0.2
21   5.1  3.7  1.5  0.4
35   5.0  3.2  1.2  0.2
102  7.1  3.0  5.9  2.1
132  6.4  2.8  5.6  2.2
126  6.2  2.8  4.8  1.8
24   4.8  3.4  1.9  0.2
61   5.9  3.0  4.2  1.5
2    4.7  3.2  1.3  0.2
95   5.7  3.0  4.2  1.2
90   5.5  2.6  4.4  1.2
76   6.8  2.8  4.8  1.4
117  7.7  3.8  6.7  2.2 87     Iris-versicolor
111     Iris-virginica
10         Iris-setosa
91     Iris-versicolor
49         Iris-setosa
60     Iris-versicolor
72     Iris-versicolor
67     Iris-versicolor
39         Iris-setosa
55     Iris-versicolor
66     Iris-versicolor
142     Iris-virginica
53     Iris-versicolor
1          Iris-setosa
19         Iris-setosa
112     Iris-virginica
85     Iris-versicolor
38         Iris-setosa
21         Iris-setosa
35         Iris-setosa
102     Iris-virginica
132     Iris-virginica
126     Iris-virginica
24         Iris-setosa
61     Iris-versicolor
2          Iris-setosa
95     Iris-versicolor
90     Iris-versicolor
76     Iris-versicolor
117     Iris-virginica
Name: Classification, dtype: object
```

In [17]:

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=4)
knn.fit(X_train,y_train)
```

Out[17]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                     weights='uniform')
```

In [18]:

```
pred1=knn.predict(X_test)
```

In [19]:

```
print(pred1)
```

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

In [20]:

```
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,knn.predict(X_test)))
```

```
[[10  0  0]
 [ 0 12  1]
 [ 0  0  7]]
```

In [21]:

```
print(classification_report(y_test,knn.predict(X_test)))
```

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.88      1.00      0.93         7

       accuracy                           0.97        30
      macro avg       0.96      0.97      0.96        30
   weighted avg       0.97      0.97      0.97        30
```

In [22]:

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import recall_score , precision_score , roc_auc_score ,roc_curve
from sklearn.metrics import confusion_matrix
```

In [23]:

```
cv_scores = []
neighbors = list(np.arange(3,50,2))
for n in neighbors:
    knn = KNeighborsClassifier(n_neighbors = n,algorithm = 'brute')

    cross_val = cross_val_score(knn,X_train,y_train,cv = 5 , scoring = 'accuracy')
    cv_scores.append(cross_val.mean())

error = [1-x for x in cv_scores]
optimal_n = neighbors[ error.index(min(error)) ]
knn_optimal = KNeighborsClassifier(n_neighbors = optimal_n,algorithm = 'brute')
knn_optimal.fit(X_train,y_train)
pred = knn_optimal.predict(X_test)
acc = accuracy_score(y_test,pred)*100
print("The accuracy for optimal k = {0} using brute is {1}".format(optimal_n,acc))
```

```
The accuracy for optimal k = 7 using brute is 96.66666666666667
```

In [24]:

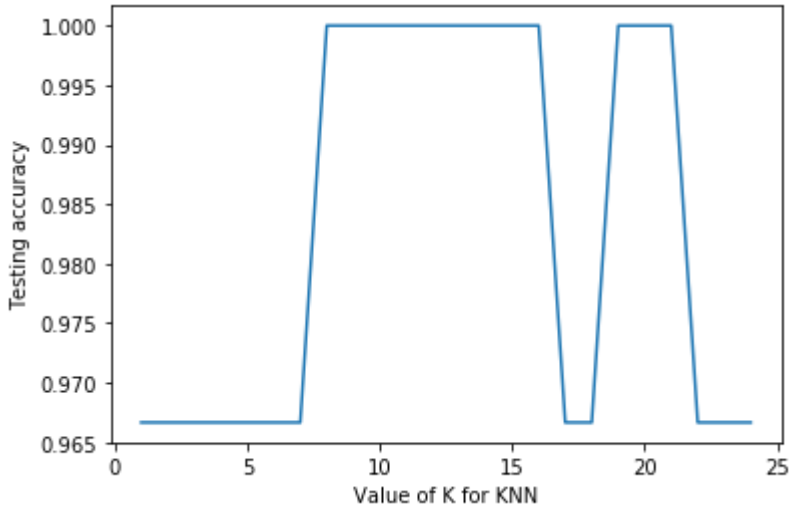
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
k_range=range(1,25)
scores={}
scores_list=[]
for k in k_range:
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train,y_train)
    pred2=knn.predict(X_test)
    scores[k]=metrics.accuracy_score(y_test,pred2)
    scores_list.append(metrics.accuracy_score(y_test,pred2))
```

In [25]:

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.plot(k_range,scores_list)
plt.xlabel('Value of K for KNN')
plt.ylabel('Testing accuracy')
```

Out[25]:

```
Text(0, 0.5, 'Testing accuracy')
```



In [26]:

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=13)
knn.fit(X_train,y_train)
```

Out[26]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=13, p=2,
                     weights='uniform')
```

In [27]:

```
pred3=knn.predict(X_test)
```

In [28]:

```
print(pred3)
```

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

In [29]:

```
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,knn.predict(X_test)))
```

```
[[10  0  0]
 [ 0 13  0]
 [ 0  0  7]]
```

In [30]:

```
print(classification_report(y_test,knn.predict(X_test)))
```

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00        13
 Iris-virginica       1.00      1.00      1.00         7

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

In [31]:

```
to_drop=['SL','SW']
iris1=iris.drop(to_drop,axis=1)
iris1.head()
X_train, X_test, y_train, y_test = train_test_split(iris1.drop('Classification',axis=1
),
                                                    iris1['Classification'], test_size=
0.20,
                                                    random_state=21)
```

In [32]:

```
print(X_train,y_train)
```

```
      PL   PW
41   1.3  0.3
131  6.4  2.0
70   4.8  1.8
46   1.6  0.2
126  4.8  1.8
..   ...  ...
120  5.7  2.3
112  5.5  2.1
48   1.5  0.2
4    1.4  0.2
56   4.7  1.6

[120 rows x 2 columns] 41         Iris-setosa
131     Iris-virginica
70     Iris-versicolor
46         Iris-setosa
126     Iris-virginica
           ...
120     Iris-virginica
112     Iris-virginica
48         Iris-setosa
4          Iris-setosa
56     Iris-versicolor
Name: Classification, Length: 120, dtype: object
```

In [33]:

```
print(X_test,y_test)
```

```
      PL   PW
92   4.0  1.2
44   1.9  0.4
7    1.5  0.2
21   1.5  0.4
95   4.2  1.2
75   4.4  1.4
20   1.7  0.2
121  4.9  2.0
26   1.6  0.4
19   1.5  0.3
81   3.7  1.0
88   4.1  1.3
143  5.9  2.3
117  6.7  2.2
23   1.7  0.5
77   5.0  1.7
138  4.8  1.8
73   4.7  1.2
14   1.2  0.2
142  5.1  1.9
123  4.9  1.8
62   4.0  1.0
83   5.1  1.6
74   4.3  1.3
42   1.3  0.2
60   3.5  1.0
40   1.3  0.3
45   1.4  0.3
87   4.4  1.3
124  5.7  2.1 92     Iris-versicolor
44          Iris-setosa
7           Iris-setosa
21          Iris-setosa
95      Iris-versicolor
75      Iris-versicolor
20          Iris-setosa
121      Iris-virginica
26          Iris-setosa
19          Iris-setosa
81      Iris-versicolor
88      Iris-versicolor
143      Iris-virginica
117      Iris-virginica
23          Iris-setosa
77      Iris-versicolor
138      Iris-virginica
73      Iris-versicolor
14          Iris-setosa
142      Iris-virginica
123      Iris-virginica
62      Iris-versicolor
83      Iris-versicolor
74      Iris-versicolor
42          Iris-setosa
60      Iris-versicolor
40          Iris-setosa
45          Iris-setosa
87      Iris-versicolor
124      Iris-virginica
Name: Classification, dtype: object
```

In [34]:

```
from sklearn.neighbors import KNeighborsClassifier
knn2=KNeighborsClassifier(n_neighbors=5)
knn2.fit(X_train,y_train)
```

Out[34]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [35]:

```
pred4=knn2.predict(X_test)
```

In [36]:

```
print(pred4)
```

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

In [37]:

```
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,knn2.predict(X_test)))
```

```
[[11  0  0]
 [ 0 10  2]
 [ 0  0  7]]
```

In [38]:

```
print(classification_report(y_test,knn2.predict(X_test)))
```

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.83      0.91        12
 Iris-virginica       0.78      1.00      0.88         7

       accuracy                           0.93        30
      macro avg       0.93      0.94      0.93        30
   weighted avg       0.95      0.93      0.93        30
```

In [39]:

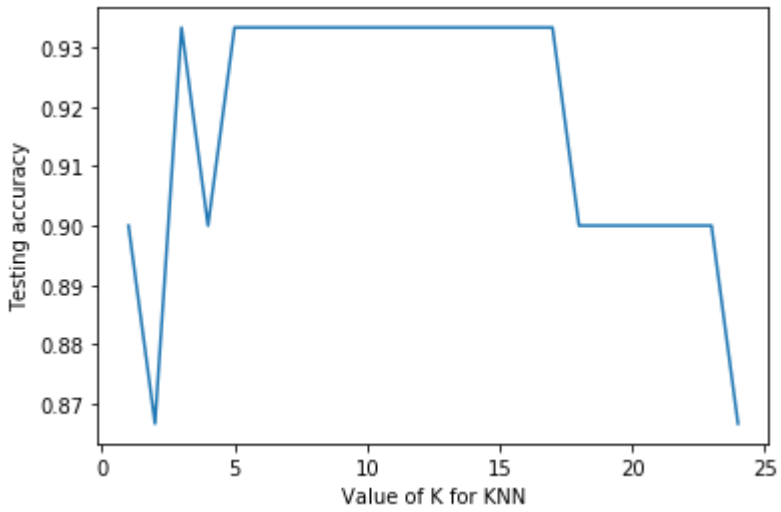
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
k_range=range(1,25)
scores={}
scores_list=[]
for k in k_range:
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train,y_train)
    pred5=knn.predict(X_test)
    scores[k]=metrics.accuracy_score(y_test,pred5)
    scores_list.append(metrics.accuracy_score(y_test,pred5))
```

In [40]:

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.plot(k_range,scores_list)
plt.xlabel('Value of K for KNN')
plt.ylabel('Testing accuracy')
```

Out[40]:

```
Text(0, 0.5, 'Testing accuracy')
```



In [41]:

```
from sklearn.neighbors import KNeighborsClassifier
knn3=KNeighborsClassifier(n_neighbors=15)
knn3.fit(X_train,y_train)
```

Out[41]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=15, p=2,
                     weights='uniform')
```

In [42]:

```
pred6=knn3.predict(X_test)
```

In [43]:

```
print(pred6)
```

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

In [44]:

```
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,knn3.predict(X_test)))
```

```
[[11  0  0]
 [ 0 10  2]
 [ 0  0  7]]
```

In [45]:

```
print(classification_report(y_test,knn3.predict(X_test)))
```

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.83      0.91        12
 Iris-virginica       0.78      1.00      0.88         7

       accuracy                           0.93        30
      macro avg       0.93      0.94      0.93        30
   weighted avg       0.95      0.93      0.93        30
```