In [1]:

```python
# Author : Amir Shokri
# github link : https://github.com/amirshnll/iris
# dataset link : http://archive.ics.uci.edu/ml/datasets/Iris
# email : amirsh.nll@gmail.com
```

In [1]:

```python
import pandas as pd
```

In [5]:

```python
data = pd.read_csv('iris.csv', header=None)
data
```

Out[5]:

	0	1	2	3	4
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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [4]:

```python
data.describe()
```

Out[4]:

	0	1	2	3
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [6]:

```python
x = data[data.columns[:4]]
Y = data[data.columns[4]]
```

In [7]:

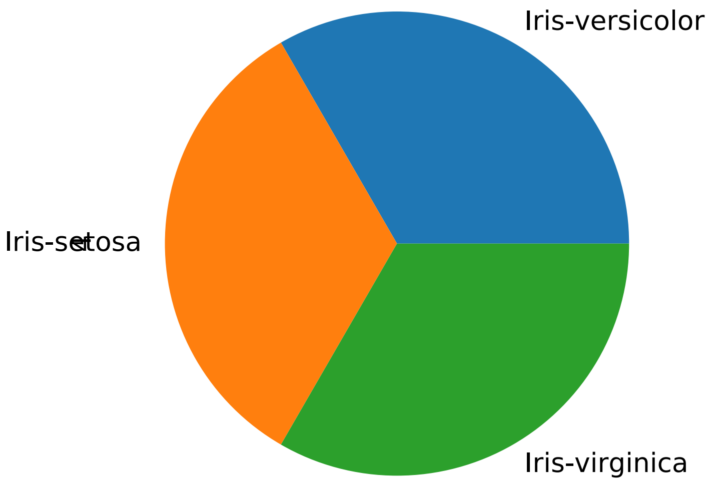
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_x = scaler.fit_transform(x)
```

In [9]:

```
Y.value_counts().plot.pie()
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f0814984c8>
```



In [11]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(scaled_x, Y, test_size=0.3
, random_state=0)
```

In [12]:

```
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
```

In [14]:

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(accuracy_score(y_test, y_pred))
print(f1_score(y_test, y_pred, average='micro'))
```

```
0.9777777777777777
0.9777777777777777
```

In [15]:

```
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(accuracy_score(y_test, y_pred))
print(f1_score(y_test, y_pred, average='micro'))
```

```
0.9777777777777777
0.9777777777777777
```

In [16]:

```
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(accuracy_score(y_test, y_pred))
print(f1_score(y_test, y_pred, average='micro'))
```

```
1.0
1.0
```

In [17]:

```
from sklearn.neural_network import MLPClassifier
clf = MLPClassifier(hidden_layer_sizes=(100,))
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(accuracy_score(y_test, y_pred))
print(f1_score(y_test, y_pred, average='micro'))
```

```
0.9777777777777777
0.9777777777777777
```

In [19]:

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(accuracy_score(y_test, y_pred))
print(f1_score(y_test, y_pred, average='micro'))
```

```
0.9777777777777777
0.9777777777777777
```

In [ ]: