In [1]:
```
import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn import metrics
from sklearn.model_selection import train_test_split
from warnings import simplefilter


# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)
```

In [2]:
```
# Dataset Path
DATASET_PATH = "dataset/glass.data"
```

In [3]:
```
glass_data_headers = ["Id", "RI", "Na", "Mg", "Al", "Si", "K", "Ca", "Ba", "Fe", "glass-type"]
glass_data = pd.read_csv(DATASET_PATH, names=glass_data_headers)
```

In [4]:
```
print ("Number of observations :: ", len(glass_data.index))
print ("Number of columns :: ", len(glass_data.columns))
print ("Headers :: ", glass_data.columns.values)
#print ("Target :: ", glass_data[glass_data_headers[-1]])
```

```
Number of observations ::  214
Number of columns ::  11
Headers ::  ['Id' 'RI' 'Na' 'Mg' 'Al' 'Si' 'K' 'Ca' 'Ba' 'Fe' 'glass-type']
```

In [5]:
```
print ("glass_data_RI :: ", list(glass_data["RI"][:10]))
print ("glass_data_target :: ", np.array([1, 1, 1, 2, 2, 3, 4, 5, 6, 7]))
```

```
glass_data_RI ::  [1.52101, 1.5176100000000001, 1.5161799999999999, 1.51766, 1.51742, 1.51596, 1.5174299999999998, 1.51756, 1.5
1918, 1.51755]
glass_data_target ::  [1 1 1 2 2 3 4 5 6 7]
```

In [6]:
```
train_x, test_x, train_y, test_y = train_test_split(glass_data[glass_data_headers[:-1]],
                                                    glass_data[glass_data_headers[-1]],
                                                    train_size=0.7)
```

In [7]:
```
# Train multi-classification model with logistic regression
lr = linear_model.LogisticRegression()
lr.fit(train_x, train_y)
```

Out[7]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [8]:
```
# Train multinomial logistic regression model
mul_lr = linear_model.LogisticRegression(multi_class='multinomial', solver='newton-cg').fit(train_x, train_y)
```

In [9]:
```
print ("Logistic regression Train Accuracy :: ", metrics.accuracy_score(train_y, lr.predict(train_x)))
print ("Logistic regression Test Accuracy :: ", metrics.accuracy_score(test_y, lr.predict(test_x)))

print ("Multinomial Logistic regression Train Accuracy :: ", metrics.accuracy_score(train_y, mul_lr.predict(train_x)))
print ("Multinomial Logistic regression Test Accuracy :: ", metrics.accuracy_score(test_y, mul_lr.predict(test_x)))
```

```
Logistic regression Train Accuracy ::  0.8993288590604027
Logistic regression Test Accuracy ::  0.8615384615384616
Multinomial Logistic regression Train Accuracy ::  1.0
Multinomial Logistic regression Test Accuracy ::  0.9846153846153847
```