In [4]:
```
#importing dependencies
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

In [28]:
```
wine_df = load_wine()
```

In [32]:
```
wine_df
```

Out[32]:

```
{'data': array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+00,
        1.065e+03],
       [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
        1.050e+03],
       [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
        1.185e+03],
       ...,
       [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
        8.350e+02],
       [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
        8.400e+02],
       [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
        5.600e+02]]),
'target': 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,
       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, 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]),
'frame': None,
'target_names': array(['class_0', 'class_1', 'class_2'], dtype='<U7'),
'DESCR': '.. _wine_dataset:\n\nWine recognition dataset\n------------------------\n\n**Data Set Characteristics:**\n\n    :Number of Instances: 178\n    :Number of Attributes: 13 numeric, predictive attributes and the class\n    :Attribute Information:\n \t\t- Alcohol\n \t\t- Malic acid\n \t\t- Ash\n\t\t- Alcalinity of ash  \n \t\t- Magnesium\n\t\t- Total phenols\n \t\t- Flavanoids\n \t\t- Nonflavanoid phenols\n \t\t- Proanthocyanins\n\t\t- Color intensity\n \t\t- Hue\n \t\t- OD280/OD315 of diluted wines\n \t\t- Proline\n\n    - class_0\n    - class_1\n    - class_2\n\t\t\n    :Summary Statistics:\n\n    ============================= ==== ===== ======= =====\n    \n    Min   Max   Mean     SD\n    ============================= ==== ===== ======= =====\n    Alcohol:                      11.0  14.8    13.0  0.8\n    Malic Acid:                   0.74  5.80  2.34  1.12\n    Ash:                          1.36  3.23    2.36  0.27\n    Alcalinity of Ash:            10.6  30.0    19.5  3.3\n    Magnesium:                    70.0 162.0    99.7  14.3\n    Total Phenols:                0.98  3.88    2.29  0.63\n    Flavanoids:                   0.34  5.08    2.03  1.00\n    Nonflavanoid Phenols:         0.13  0.66    0.36  0.12\n    Proanthocyanins:              0.41  3.58    1.59  0.57\n    Colour Intensity:             1.3   13.0    5.1   2.3\n    Hue:                          0.48  1.71    0.96  0.23\n    OD280/OD315 of diluted wines: 1.27  4.00    2.61  0.71\n    Proline:                      278   1680    746   315\n    ============================= ==== ===== ======= =====\n\n    :Missing Attribute Values: None\n    :Class Distribution: class_0 (59), class_1 (71), class_2 (48)\n    :Creator: R.A. Fisher\n    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n    :Date: July, 1988\n\nThis is a copy of UCI ML Wine recognition datasets.\nhttps://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data\n\nThe data is the results of a chemical analysis of wines grown in the same\nregion in Italy by three different cultivators. There are thirteen different\nmeasurements taken for different constituents found in the three types of\nwine.\n\nOriginal Owners: \n\nForina, M. et al, PARVUS - \nAn Extendible Package for Data Exploration, Classification and Correlation. \nInstituto di Pharmaceutical and Food Analysis and Technologies,\nVia Brigata Salerno, 16147 Genoa, Italy.\n\nCitation:\n\nLichman, M. (2013). UCI Machine Learning Repository\n[https://archive.ics.uci.edu/ml]. Irvine, CA: University of California,\nSchool of Information and Computer Science. \n\n.. topic:: References\n\n (1) S. Aeberhard, D. Coomans and O. de Vel, \n "Comparison of Classifiers in High Dimensional Settings, \n Tech. Rep. no. 92-02, (1992), Dept. of Computer Science and Dept. of \n Mathematics and Statistics, James Cook University of North Queensland. \n (Also submitted to Technometrics). \n\n The data was used with many others for comparing various \n classifiers. The classes are separable, though only RDA \n has achieved 100% correct classification. \n (RDA : 100%, QDA 99.4%, LDA 98.9%, 1NN 96.1% (z-transformed data)) \n (All results using the leave-one-out technique) \n\n (2) S. Aeberhard, D. Coomans and O. de Vel, \n "THE CLASSIFICATION PERFORMANCE OF RDA" \n Tech. Rep. no. 92-01, (1992), Dept. of Computer Science and Dept. of \n Mathematics and Statistics, James Cook University of North Queensland. \n (Also submitted to Journal of Chemometrics).\n',
'feature_names': ['alcohol',
 'malic_acid',
 'ash',
 'alcalinity_of_ash',
 'magnesium',
 'total_phenols',
 'flavanoids',
 'nonflavanoid_phenols',
 'proanthocyanins',
 'color_intensity',
 'hue',
 'od280/od315_of_diluted_wines',
 'proline']}
```


In [39]:
```
X_train,X_test,y_train,y_test = train_test_split(wine_df.data,wine_df.target,test_size=0.4,random_state=0)
```

In [40]:
```
X_train
```

Out[40]:
```
array([[1.324e+01, 3.900e+00, 2.29e+00, ..., 8.20e-01, 3.000e+00,
        6.800e+02],
       [1.410e+01, 2.160e+00, 2.300e+00, ..., 1.25e+00, 3.170e+00,
        1.515e+03],
       [1.161e+01, 1.350e+00, 2.700e+00, ..., 9.60e-01, 3.260e+00,
        2.095e+03],
       ...,
       [1.242e+01, 1.61e+00, 2.190e+00, ..., 1.060+00, 2.96e+00,
        3.450e+02],
       [1.390e+01, 1.68e+00, 2.12e+00, ..., 9.10e-01, 3.33e+00,
        9.85e+02],
       [1.416e+01, 2.51e+00, 2.48e+00, ..., 6.20e-01, 1.71e+00,
        6.60e+02]])
```

In [41]:
```
X_test
```

Out[41]:
```
array([[1.374e+01, 1.67e+00, 2.25e+00, 1.64e+01, 1.18e+02, 2.60e+00,
        2.900e+00, 2.10e-01, 1.62e+00, 5.85e+00, 9.20e-01, 3.20e+00,
        1.06e+03],
       [1.27e+01, 2.67e+00, 2.48e+00, 2.20e+01, 1.12e+02, 1.48e+00,
        1.36e+00, 2.40e-01, 1.26e+00, 1.08e+01, 4.80e-01, 1.47e+00,
        4.80e+02],
       [1.237+01, 1.13e+00, 2.16e+00, 1.90e+01, 8.70e+01, 3.50e+00,
        3.10e+00, 1.90e-01, 1.87e+00, 4.45e+00, 1.22e+00, 2.87e+00,
        4.20e+02],
       [1.356e+01, 1.73e+00, 2.46e+00, 2.05e+01, 1.16e+02, 2.96e+00,
        2.78e+00, 2.00e-01, 2.45e+00, 6.25e+00, 9.80e-01, 3.03e+00,
        1.12e+03],
       [1.305e+01, 5.80e+00, 2.13e+00, 2.15e+01, 8.60e+01, 2.62e+00,
        2.65e+00, 3.00e-01, 2.01e+00, 2.60e+00, 7.30e-01, 3.10e+00,
        3.80e+02],
       [1.156e+01, 2.05e+00, 3.23e+00, 2.85e+01, 1.19e+02, 3.18e+00,
        5.08e+00, 4.70e-01, 1.87e+00, 6.00e+00, 9.30e-01, 3.69e+00,
        4.65e+02],
       [1.40e+01, 2.15e+00, 2.61e+00, 1.76e+01, 1.21e+02, 2.60e+00,
        2.51e+00, 3.10e-01, 1.25e+00, 5.05e+00, 1.06e+00, 3.58e+00,
        1.29e+03],
       [1.236e+01, 3.83e+00, 2.38e+00, 2.10e+01, 8.80e+01, 2.30e+00,
        9.20e-01, 5.00e-01, 1.64e+00, 7.65e+00, 5.60e-01, 1.58e+00,
        5.20e+02],
       [1.225e+01, 1.73e+00, 2.12e+00, 1.90e+01, 8.80e+01, 1.65e+00,
        2.03e+00, 3.70e-01, 1.63e+00, 3.40e+00, 1.00e+00, 3.17e+00,
        5.10e+02],
       [1.28e+01, 1.83e+00, 2.32e+00, 1.85e+01, 8.10e+01, 1.60e+00,
        1.50e+00, 5.20e-01, 1.64e+00, 2.40e+00, 1.08e+00, 2.27e+00,
        4.80e+02],
       [1.336e+01, 2.56e+00, 2.35e+00, 2.00e+01, 8.90e+01, 1.40e+00,
        5.00e-01, 3.70e-01, 6.40e-01, 5.60e+00, 7.00e-01, 2.47e+00,
        7.80e+02],
       [1.388e+01, 5.04e+00, 2.23e+00, 2.00e+01, 8.00e+01, 9.80e-01,
        3.40e-01, 4.00e-01, 6.80e-01, 4.90e+00, 5.80e-01, 1.33e+00,
        4.15e+02],
       [1.42e+01, 1.76e+00, 2.45e+00, 1.52e+01, 1.12e+02, 3.27e+00,
        3.39e+00, 3.40e-01, 1.97e+00, 6.75e+00, 1.05e+00, 2.85e+00,
        1.45e+03],
       [1.237e+01, 1.07e+00, 2.10e+00, 1.85e+01, 8.80e+01, 3.52e+00,
        3.75e+00, 2.40e-01, 1.95e+00, 4.50e+00, 1.04e+00, 2.77e+00,
        6.60e+02],
       [1.358e+01, 2.58e+00, 2.69e+00, 2.45e+01, 1.05e+02, 1.55e+00,
        8.40e-01, 3.90e-01, 1.54e+00, 8.66e+00, 7.40e-01, 1.80e+00,
        7.50e+02],
       [1.200e+01, 9.20e-01, 2.00e+00, 1.90e+01, 8.60e+01, 2.42e+00,
        2.26e+00, 3.00e-01, 1.43e+00, 2.50e+00, 1.38e+00, 3.12e+00,
        2.78e+02],
       [1.376e+01, 1.53e+00, 2.70e+00, 1.95e+01, 1.32e+02, 2.95e+00,
        2.74e+00, 5.00e-01, 1.35e+00, 5.40e+00, 1.25e+00, 3.00e+00,
        1.23e+03],
       [1.419e+01, 1.59e+00, 2.48e+00, 1.65e+01, 1.08e+02, 3.30e+00,
        3.93e+00, 3.20e-01, 1.86e+00, 8.70e+00, 1.23e+00, 2.82e+00,
        1.68e+03],
       [1.264e+01, 1.36e+00, 2.02e+00, 1.68e+01, 1.00e+02, 2.02e+00,
        1.41e+00, 5.30e-01, 6.20e-01, 5.75e+00, 9.80e-01, 1.59e+00,
        4.50e+02],
       [1.383e+01, 1.65e+00, 2.60e+00, 1.72e+01, 9.40e+01, 2.45e+00,
        2.99e+00, 2.20e-01, 2.29e+00, 5.60e+00, 1.24e+00, 3.37e+00,
        1.26e+03],
       [1.311e+01, 1.01e+00, 1.70e+00, 1.50e+01, 7.80e+01, 2.98e+00,
        3.18e+00, 2.60e-01, 2.28e+00, 5.30e+00, 1.12e+00, 3.18e+00,
        5.02e+02],
       [1.305e+01, 1.65e+00, 2.55e+00, 1.80e+01, 9.80e+01, 2.45e+00,
        2.43e+00, 2.90e-01, 1.44e+00, 4.25e+00, 1.12e+00, 2.51e+00,
        1.105e+03],
       [1.324e+01, 2.59e+00, 2.87e+00, 2.10e+01, 1.18e+02, 2.80e+00,
        2.69e+00, 3.90e-01, 1.82e+00, 4.32e+00, 1.04e+00, 2.93e+00,
        7.35e+02],
       [1.251e+01, 1.73e+00, 1.98e+00, 2.05e+01, 8.50e+01, 2.20e+00,
        1.92e+00, 3.20e-01, 1.48e+00, 2.94e+00, 1.04e+00, 3.57e+00,
        6.72e+02],
       [1.233e+01, 1.10e+00, 2.28e+00, 1.60e+01, 1.01e+02, 2.05e+00,
        1.09e+00, 6.30e-01, 4.10e-01, 3.27e+00, 1.25e+00, 1.67e+00,
        8.80e+02],
       [1.252e+01, 2.43e+00, 2.17e+00, 2.10e+01, 8.80e+01, 2.55e+00,
        2.27e+00, 2.60e-01, 1.22e+00, 2.00e+00, 9.00e-01, 2.78e+00,
        3.25e+02],
       [1.243e+01, 1.53e+00, 2.29e+00, 2.15e+01, 8.60e+01, 2.74e+00,
        3.15e+00, 3.90e-01, 1.77e+00, 3.94e+00, 6.90e-01, 2.84e+00,
        5.52e+02],
       [1.216e+01, 1.61e+00, 2.31e+00, 2.28e+01, 9.00e+01, 1.78e+00,
        1.69e+00, 4.30e-01, 1.56e+00, 2.45e+00, 1.33e+00, 2.26e+00,
        4.95e+02],
       [1.17+01, 2.68e+00, 2.92e+00, 2.00e+01, 1.03e+02, 1.75e+00,
        2.03e+00, 6.00e-01, 1.05e+00, 3.80e+00, 1.23e+00, 2.50e+00,
        6.07e+02],
       [1.378+01, 2.76e+00, 2.30e+00, 2.20e+01, 9.00e+01, 1.35e+00,
        6.80e-01, 4.10e-01, 1.03e+00, 9.58e+00, 7.00e-01, 1.68e+00,
        6.15e+02],
       [1.339+01, 1.77e+00, 2.62e+00, 1.61e+01, 9.30e+01, 2.85e+00,
        2.94e+00, 3.40e-01, 1.45e+00, 4.80e+00, 9.20e-01, 3.22e+00,
        1.19e+03],
       [1.422+01, 1.70e+00, 2.30e+00, 1.63e+01, 1.18e+02, 3.20e+00,
        3.00e+00, 2.60e-01, 2.03e+00, 6.38e+00, 9.40e-01, 3.31e+00,
        9.70e+02],
       [1.24+01, 4.43e+00, 2.38e+00, 2.20e+01, 8.00e+01, 2.10e+00,
        1.75e+00, 4.20e-01, 1.35e+00, 2.60e+00, 7.90e-01, 2.57e+00,
        5.80e+02],
       [1.421e+01, 4.04e+00, 2.44e+00, 1.89e+01, 1.11e+02, 2.85e+00,
        2.65e+00, 3.00e-01, 1.25e+00, 5.24e+00, 8.70e-01, 3.33e+00,
        1.080e+03],
       [1.483+01, 1.64e+00, 2.17e+00, 1.40e+01, 9.70e+01, 2.80e+00,
        2.98e+00, 2.90e-01, 1.98e+00, 5.20e+00, 1.08e+00, 2.85e+00,
        1.045e+03],
       [1.305e+01, 1.77e+00, 2.10e+00, 1.70e+01, 1.07e+02, 3.00e+00,
        3.00e+00, 2.80e-01, 2.03e+00, 5.04e+00, 8.80e-01, 3.35e+00,
        8.85e+02],
       [1.29e+01, 3.26e+00, 2.54e+00, 2.00e+01, 1.07e+02, 1.83e+00,
        5.60e-01, 5.00e-01, 8.00e-01, 5.80e+00, 9.60e-01, 1.82e+00,
        6.80e+02],
       [1.26e+01, 1.53e+00, 2.26e+00, 2.07e+01, 8.00e+01, 1.38e+00,
        1.46e+00, 5.80e-01, 1.62e+00, 3.05e+00, 9.60e-01, 2.06e+00,
        4.95e+02],
       [1.162+01, 1.99e+00, 2.28e+00, 1.80e+01, 9.80e+01, 3.02e+00,
        2.26e+00, 1.70e-01, 1.35e+00, 3.25e+00, 1.16e+00, 2.96e+00,
        3.45e+02],
       [1.34e+01, 3.91e+00, 2.48e+00, 2.30e+01, 1.02e+02, 1.80e+00,
        7.50e-01, 4.30e-01, 1.41e+00, 7.30e+00, 7.00e-01, 1.56e+00,
        7.50e+02],
       [1.35e+01, 1.81e+00, 2.61e+00, 2.00e+01, 9.60e+01, 2.53e+00,
        2.61e+00, 2.80e-01, 1.66e+00, 3.52e+00, 1.12e+00, 3.82e+00,
        8.45e+02],
       [1.373+01, 1.50e+00, 2.70e+00, 2.25e+01, 1.01e+02, 3.00e+00,
        3.25e+00, 2.90e-01, 2.38e+00, 5.70e+00, 1.19e+00, 2.71e+00,
        1.285e+03],
       [1.229e+01, 2.83e+00, 2.22e+00, 1.80e+01, 8.80e+01, 2.45e+00,
        2.25e+00, 2.50e-01, 1.99e+00, 2.15e+00, 1.15e+00, 3.30e+00,
        2.90e+02],
       [1.26e+01, 1.34e+00, 1.90e+00, 1.85e+01, 8.80e+01, 1.45e+00,
        1.36e+00, 2.90e-01, 1.35e+00, 2.45e+00, 1.04e+00, 2.77e+00,
        5.62e+02],
       [1.141e+01, 7.40e-01, 2.50e+00, 2.10e+01, 8.80e+01, 2.48e+00,
        2.01e+00, 4.20e-01, 1.44e+00, 3.08e+00, 1.10e+00, 2.31e+00,
        4.34e+02],
       [1.344e+01, 3.10e+00, 2.56e+00, 1.52e+01, 1.16e+02, 2.70e+00,
        3.03e+00, 1.70e-01, 1.66e+00, 5.10e+00, 9.60e-01, 3.36e+00,
        8.45e+02],
       [1.26e+01, 2.46e+00, 2.20e+00, 1.85e+01, 9.40e+01, 1.62e+00,
        6.60e-01, 6.30e-01, 9.40e-01, 7.10e+00, 7.30e-01, 1.58e+00,
        6.95e+02],
       [1.196e+01, 1.09e+00, 2.30e+00, 2.10e+01, 1.01e+02, 2.38e+00,
        2.14e+00, 1.30e-01, 1.65e+00, 3.21e+00, 9.90e-01, 3.13e+00,
        8.86e+02],
       [1.225e+01, 3.88e+00, 2.20e+00, 1.85e+01, 1.12e+02, 1.38e+00,
        7.00e-01, 2.90e-01, 1.14e+00, 9.21e+00, 6.50e-01, 2.00e+00,
        8.55e+02],
       [1.439+01, 1.92e+00, 2.72e+00, 2.00e+01, 1.20e+02, 2.80e+00,
        3.14e+00, 3.30e-01, 1.97e+00, 6.20e+00, 1.07e+00, 2.65e+00,
        1.280e+03],
       [1.288e+01, 2.99e+00, 2.40e+00, 2.00e+01, 1.04e+02, 1.30e+00,
        1.22e+00, 2.40e-01, 8.30e-01, 5.40e+00, 7.40e-01, 1.42e+00,
        5.30e+02],
       [1.349e+01, 3.59e+00, 2.19e+00, 1.95e+01, 8.80e+01, 1.62e+00,
        4.80e-01, 5.80e-01, 8.80e-01, 5.70e+00, 8.10e-01, 1.82e+00,
        6.80e+02],
       [1.356e+01, 1.71e+00, 2.31e+00, 1.62e+01, 1.17e+02, 3.15e+00,
        3.29e+00, 3.40e-01, 2.34e+00, 6.13e+00, 9.50e-01, 3.38e+00,
        7.95e+02],
       [1.434e+01, 1.68e+00, 2.70e+00, 2.50e+01, 9.80e+01, 2.80e+00,
        1.31e+00, 5.30e-01, 2.70e+00, 1.30e+01, 5.70e-01, 1.96e+00,
        6.60e+02],
       [1.371e+01, 1.86e+00, 2.36e+00, 1.66e+01, 1.01e+02, 2.61e+00,
        2.88e+00, 2.70e-01, 1.69e+00, 3.80e+00, 1.11e+00, 4.00e+00,
        1.035e+03],
       [1.222e+01, 1.29e+00, 1.94e+00, 1.90e+01, 9.20e+01, 2.36e+00,
        2.04e+00, 3.90e-01, 2.08e+00, 2.70e+00, 8.60e-01, 3.02e+00,
        3.12e+02],
       [1.327e+01, 4.28e+00, 2.26e+00, 2.00e+01, 1.20e+02, 1.59e+00,
        6.90e-01, 4.30e-01, 1.35e+00, 1.02e+01, 5.90e-01, 1.56e+00,
        8.35e+02],
       [1.316e+01, 3.57e+00, 2.15e+00, 2.10e+01, 1.02e+02, 1.50e+00,
        5.50e-01, 4.30e-01, 1.30e+00, 4.00e+00, 6.00e-01, 1.68e+00,
        8.30e+02],
       [1.386e+01, 1.51e+00, 2.67e+00, 2.50e+01, 8.60e+01, 2.95e+00,
        2.86e+00, 2.10e-01, 1.87e+00, 3.38e+00, 1.36e+00, 3.16e+00,
        4.10e+02],
       [1.285e+01, 3.27e+00, 2.58e+00, 2.20e+01, 1.06e+02, 1.65e+00,
        6.00e-01, 6.00e-01, 9.60e-01, 5.58e+00, 8.70e-01, 2.11e+00,
        5.70e+02],
       [1.384e+01, 4.12e+00, 2.38e+00, 1.95e+01, 8.90e+01, 1.80e+00,
        8.30e-01, 4.80e-01, 1.56e+00, 9.01e+00, 5.70e-01, 1.64e+00,
        4.80e+02],
       [1.333e+01, 1.72e+00, 2.14e+00, 1.70e+01, 9.40e+01, 2.40e+00,
        2.19e+00, 2.70e-01, 1.35e+00, 3.95e+00, 1.02e+00, 2.77e+00,
        1.285e+03],
       [1.305e+01, 3.86e+00, 2.32e+00, 2.25e+01, 8.50e+01, 1.65e+00,
        1.59e+00, 6.10e-01, 1.62e+00, 4.80e+00, 8.40e-01, 2.01e+00,
        5.15e+02],
       [1.251e+01, 1.24e+00, 2.25e+00, 1.75e+01, 8.50e+01, 2.00e+00,
        5.80e-01, 6.00e-01, 1.25e+00, 5.45e+00, 7.50e-01, 1.51e+00,
        6.50e+02],
       [1.229e+01, 1.41e+00, 1.98e+00, 1.60e+01, 8.50e+01, 2.55e+00,
        2.50e+00, 2.90e-01, 1.77e+00, 2.90e+00, 1.23e+00, 2.74e+00,
        4.28e+02],
       [1.277e+01, 3.43e+00, 1.98e+00, 1.60e+01, 8.00e+01, 1.63e+00,
        1.25e+00, 4.30e-01, 8.30e-01, 3.40e+00, 7.00e-01, 2.12e+00,
        6.72e+02],
       [1.29e+01, 3.45e+00, 2.35e+00, 1.85e+01, 1.06e+02, 1.39e+00,
        7.00e-01, 4.00e-01, 9.40e-01, 5.28e+00, 6.80e-01, 1.75e+00,
        6.75e+02],
       [1.367+01, 1.25e+00, 1.92e+00, 1.80e+01, 9.40e+01, 2.10e+00,
        1.79e+00, 3.20e-01, 7.30e-01, 3.80e+00, 1.23e+00, 2.46e+00,
        6.30e+02],
       [1.316e+01, 2.36e+00, 2.67e+00, 1.86e+01, 1.01e+02, 2.80e+00,
        3.24e+00, 3.00e-01, 2.81e+00, 5.68e+00, 1.03e+00, 3.17e+00,
        1.185e+03],
       [1.237+01, 9.40e-01, 1.36e+00, 1.06e+01, 8.80e+01, 1.98e+00,
        5.70e-01, 2.80e-01, 4.20e-01, 1.95e+00, 1.05e+00, 1.82e+00,
        5.20e+02],
       [1.247+01, 1.52e+00, 2.20e+00, 1.90e+01, 1.62e+02, 2.50e+00,
        2.27e+00, 3.20e-01, 3.28e+00, 2.60e+00, 1.16e+00, 2.63e+00,
        9.37e+02],
       [1.181+01, 2.12e+00, 2.74e+00, 2.15e+01, 1.34e+02, 1.60e+00,
        9.90e-01, 1.40e-01, 1.56e+00, 2.50e+00, 9.50e-01, 2.26e+00,
        6.25e+02]])
```

pipeline for logistic regression

In [43]:
```
pipeline_lr=Pipeline([('scalar1',StandardScaler()),
                      ('pca1',PCA(n_components=2)),
                      ('lr_classifier',LogisticRegression(random_state=0))])
```

pipeline for Decision Tree

In [44]:
```
pipeline_dt=Pipeline([('scalar2',StandardScaler()),
                      ('pca2',PCA(n_components=2)),
                      ('dt_classifier',DecisionTreeClassifier())])
```

pipeline for RandomForest

In [45]:
```
pipeline_randomforest=Pipeline([('scalar3',StandardScaler()),
                                ('pca3',PCA(n_components=2)),
                                ('rf_classifier',RandomForestClassifier())])
```

In [46]:
```
## LEts make the list of pipelines
pipelines = [pipeline_lr, pipeline_dt, pipeline_randomforest]
```

In [47]:
```
best_accuracy=0.0
best_classifier=0
best_pipeline=""
```

In [48]:
```
# Dictionary of pipelines and classifier types for ease of reference
pipe_dict = {0: 'Logistic Regression', 1: 'Decision Tree', 2: 'RandomForest'}

# Fit the pipelines
for pipe in pipelines:
    pipe.fit(X_train, y_train)
```

In [49]:
```
for i,model in enumerate(pipelines):
    print("{} Test Accuracy: {}".format(pipe_dict[i],model.score(X_test,y_test)))
```

```
Logistic Regression Test Accuracy: 0.9583333333333334
Decision Tree Test Accuracy: 0.9583333333333334
RandomForest Test Accuracy: 0.9583333333333334
```

In [51]:
```
for i,model in enumerate(pipelines):
    if model.score(X_test,y_test)>best_accuracy:
        best_accuracy=model.score(X_test,y_test)
        best_pipeline=model
        best_classifier=i
print('Classifier with best accuracy for wine data set:{}'.format(pipe_dict[best_classifier]))
```

```
Classifier with best accuracy for wine data set:Logistic Regression
```

In [ ]:
```

```