Part 1

We're group 1, and we've designed a program that utilizes the MLPClassifier to determine a specific wine based upon its characteristics like location, chemical make-up and color.

Imports

The first step is to import the necessary assets into the program.

The "pandas" software library is a very useful tool for data manipulation and analysis in python. It offers data structures and operations for manipulating numerical tables and time series.

```
import pandas as pd
```

The "sklearn" modules are all needed to implement the Multi-layer Perceptron Classifier, or MLPClassifier, for short. This classifier is a supervised learning algorithm that learns a function by training on a dataset.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

The pickle module uses a fundamental algorithm for serializing and de-serializing a Python object structure. If an object is 'pickled', it is converted into a byte stream and then can later be 'unpickled' and converted back to its original object form.

```
import pickle
```

Defining Variables

Using panda, data is imported from an excel file and stored in the variable 'wine'. This is the dataset that the MLPClassifier will train on.

```
wine = pd.read_csv('wine.data', names = ["Cultivator", "Alchol", "Malic_Acid", "Ash",
"Alcalinity_of_Ash", "Magnesium", "Total_phenols", "Falvanoids",
"Nonflavanoid_phenols", "Proanthocyanins", "Color_intensity", "Hue", "OD280",
"Proline"])
```

Our x and y values are then taken from the variable 'wine', with X using the "drop" function to remove the 'Cultivator' row from the dataset.

```
X = wine.drop('Cultivator',axis=1)
y = wine['Cultivator']
```

We then split the data into two sets: Training and Testing. This operation is performed by using the sklearn module 'train_test _split'.

```
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

Using the sklearn "StandardScaler", "fit", and "transform" functions, we normalize the data's features by removing the mean and scaling the values to the unit variance.

```
scaler = StandardScaler(copy=True, with_mean=True, with_std=True)
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Using the MLPClassifier

Now that we've initialized the important variables, we can implement the MLPClassifier on our dataset.

```
models = [MLPClassifier(hidden_layer_sizes=(100,100,100),max_iter=1000)
for i in range(10)]
```

We first generate multiple models using list comprehensions. The "predict(x)" function returns the predicted labels of y given the unlabeled observations x.

```
[obj.fit(X_train,y_train) for obj in models]
model_predictions = [obj.predict(X_test) for obj in models]
```

In order to choose the best model, we generate scores for each one, index them, and then determine the top score using the "max" function.

```
scores = [accuracy_score(y_test,prediction) for prediction in model_predictions]
index_of_best = scores.index(max(scores))
print scores
print index_of_best
```

The optimal perceptron is then stored in a variable, and predictions are made for this classifier that we will be able to compare to the actual values, i.e. is the wine correctly identified or not?

```
optimal_perceptron = models[index_of_best]
predictions = optimal_perceptron.predict(X_test)
```

To evaluate our classifer, we check to see if the actual values, "y_test.values", are equivalent to the predicted values, "predictions". If they are, the result is a passing grade, and if not then they're given a failing grade.

```
evaluations = []
for a,b in zip(y_test.values,predictions):
    if a==b:
        evaluations.append('PASS')
    else:
        evaluations.append('FAIL')
```

Showing the Results

The final step of the program is to show the results to see how well our classifier performed. We use the "DataFrame" function from the pandas module to format a nice table to display the results.

```
results = pd.DataFrame({'Actual':y_test.values,'Prediction':predictions,
'Result':evaluations})
print results
print '\n'
print classification_report(y_test,predictions)
print "Cumulative Accuracy: %.02f%%\n" % (float(accuracy_score(y_test,predictions))*100
```

We also give the option to save all the data with a y/n input from the user.

```
while True:
    choice = raw_input('Save network? (y/n) >> ')
    if choice == 'y':
        with open('wine_detection_network.pickle', 'wb') as f:
            pickle.dump(optimal_perceptron, f)
        break
    elif choice == 'n':
        break
    else:
        print "invalid choice, please try again >> "
```

Here's an example output of the program showing the PASS/FAIL table and the classifier's precision and performance.

```
  Actual  Prediction Result
0        2           2    PASS
1        1           1    PASS
2        2           2    PASS
3        3           3    PASS
4        3           3    PASS
5        3           3    PASS
6        3           3    PASS
7        3           3    PASS
8        2           2    PASS
9        3           3    PASS
10       2           2    PASS
11       1           1    PASS
12       2           2    PASS
13       2           2    PASS
14       3           3    PASS
15       3           3    PASS
16       2           2    PASS
17       1           1    PASS
```

```
18       2            2   PASS
19       2            2   PASS
20       2            2   PASS
21       1            1   PASS
22       2            2   PASS
23       1            1   PASS
24       3            3   PASS
25       3            3   PASS
26       3            3   PASS
27       2            2   PASS
28       2            2   PASS
29       2            2   PASS
30       1            1   PASS
31       1            1   PASS
32       3            3   PASS
33       2            2   PASS
34       1            1   PASS
35       3            3   PASS
36       2            2   PASS
37       1            1   PASS
38       3            3   PASS
39       2            2   PASS
40       1            1   PASS
41       3            3   PASS
42       2            2   PASS
43       2            2   PASS
44       2            2   PASS


             precision    recall  f1-score   support

          1       1.00      1.00      1.00        10
          2       1.00      1.00      1.00        20
          3       1.00      1.00      1.00        15

avg / total       1.00      1.00      1.00        45

Cumulative Accuracy: 100.00%
```