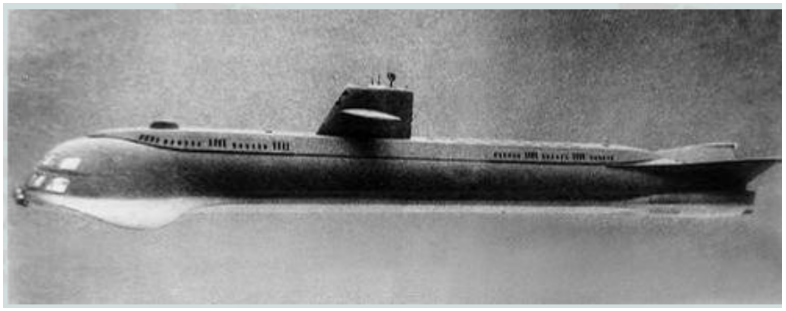
Project: Machine Learning Mine Versus Rock



You are the sonar operator on the SeaView Naval submarine. The path of your next mission will take you through an old mine field that has never been cleared. You know that your improbable Cobra shaped submarine is barely sea worthy and cannot stand a mine explosion. Experience tells you mines blowing up within 30 meters of the ship are a problem. These mines use magnetic detection to tell when a ship is near. You are concerned because the SeaView is particular vulnerable, and there is no good way to escape a submarine if it is stricken by a mine. (http://vttbots.com/Page3.html)

Given all this motivation you have some training data that may allow you to detect the difference between the mines and surrounding rocks. It is critical that you detect as many mines as possible. Your test data is in file sonar_all_data_2.csv, available in the project assignment. There you will find multiple observations of sonar data for mines and rocks each with 60 time samples which either indicate the presence of a mine (actually a hollow pipe) or a rock (actually a rock).

Split the data to train using 70% of the data set to detect a mine versus a rock and the apply this to the 30% test dataset, as usual. You will use a neural network to train and test. However, this project will focus on PCA analysis. Perform learning on the data set using 1 through 60 principal components. That is, use just the top component, then the top two, then the top three, and so on. Determine the number of components that yields the best results on the test data.

To create a neural network, use the MLPClassifier, which is a multi-level Perceptron. You may want to start with something like this:

```
model = MLPClassifier( hidden_layer_sizes=(100), activation='logistic', max_iter=2000, alpha=0.00001,
                       solver='adam', tol=0.0001 )
```

Feel free to modify the parameters or add others to tune your results.

Each time you run, the above will start with a different random seed. How much difference does this make to your results? Is it better to pick a random seed and use that seed for each number of components and then try again with a different seed? (random_state=new_seed is how you set the seed.)

It is likely that you will have convergence issues that will result in warnings. To suppress the warnings, use this code:

```
from warnings import filterwarnings
filterwarnings('ignore')
```

To create a confusion matrix:

```
from sklearn.metrics import confusion_matrix
cmat = confusion_matrix(actual,predicted)
```

Deliverables:

1) Your python code which should be well organized and documented and in a file called proj2.py.
2) For each number of components used, print the number of components and the accuracy achieved. (Use test accuracy.)
3) At the end, print the maximum accuracy along with the number of components that achieved that accuracy.
4) Plot accuracy versus the number of components.
5) Print the confusion matrix which results from the analysis which resulted in the maximum accuracy.
6) Write a one paragraph conclusion of this work containing reference to results, and conclusions you draw about your chances of surviving a real mine field. What does the confusion matrix tell you? In addition, discuss why that number of components was the maximum as opposed to some other number and the general shape of the plot you created. Finally, explain how you chose the parameters for the MLPClassifier. This paragraph should be in a file called proj2.pdf.