Part A – Exploratory Data Analysis

It seems as if Benign patients have Bland Chromatin levels centered around the range of 0-4, whereas Malignant patients have Bland Chromatin levels ranging from 0-10, with the majority being greater than 5.

The separation between Benign and Malignant patient's Bar Nuclei is pretty clear. Benign patients tend to have a Bar Nuclei of 1, while Malignant patients are usually around 10.

It seems as if Benign patients have a Clump Thickness mainly from 0-5, while Malignant patients tend to have a Clump Thickness from 5-10. So the separation is pretty clear.

Benign patients usually have Marginal Adhesion around 1, while Malignant patients have Marginal Adhesion levels that are more spread out from 0-10.

Benign patients usually have Mitoses around 1, while Malignant patients have Mitoses that are more spread out from 0-10.

Benign patients usually have Normal Nucleoli around 1, while Malignant patients have Normal Nucleoli that are more spread out from 0-10.

Benign patients usually have Single Epithelial Cell sizes ranging from 0 to 2, while Malignant patients have Single Epithelial Cell sizes that are more evenly distributed, ranging from 0 to 10.

Benign patients usually have Uniformity of Cell Shapes ranging from 0- 3, while Malignant patients usually have Uniformity of Cell Shapes that are more evenly distributed, ranging from 0 to 10.

Benign patients usually have Uniformity of Cell Sizes ranging from 0- 3, while Malignant patients usually have Uniformity of Cell Sizes that are more evenly distributed, ranging from 0 to 10.

As for the total number of each record in the dataset, there seems to be more Benign samples than Malignant. For example, there are about 450 Benign samples while there are only about 250 Malignant samples.

Part B – Build the best models

In searching for the optimal model for the bcw dataset, I used the sklearn.model_selection class of GridSearchCV. This class has cross-validation built in, and allows you to set the number of folds upon initialization. The value I set for the number of cross-validation folds was 3. For input parameters to the GridSearchCV class, I used the following:

Kernel: linear, poly, rbf, sigmoid
C: .01, .1, 1, 10
Degree: 2, 3, 4, 5
Gamma: .01, .1, 1, 10
Coef0: -10, 0, 10, 20

The resulting two best models emerged from these test parameters:

```
Fitting 3 folds for each of 1024 candidates, totalling 3072 fits
[Parallel(n_jobs=1)]: Done 3072 out of 3072 | elapsed:  1.1min finished

Best Model Score: 0.9808917197452229
Best Model Params: {'kernel': 'rbf', 'degree': 2, 'C': 1, 'gamma': 0.01, 'coef0': -10}
Best Model Confusion Matrix: [[102, 2], [1, 52]]

Second Best Model Score 0.9746835443037974
Second Best Model Params: {'kernel': 'poly', 'degree': 3, 'C': 0.01, 'gamma': 0.01, 'coef0': -10}
Second Best Model Confusion Matrix: [[102, 3], [1, 52]]


Process finished with exit code 0
```

Best: kernel=rbf, degree=2, C=1, gamma=0.01, coef0=01
Second Best: kernel=poly, degree=3, C=0.01, gamma=0.01, coef0=-10

Part C – Confusion Matrix

The output of my program included the confusion matrixes for both of the top two models:

```
Fitting 3 folds for each of 1024 candidates, totalling 3072 fits
[Parallel(n_jobs=1)]: Done 3072 out of 3072 | elapsed:  1.1min finished

Best Model Score: 0.9808917197452229
Best Model Params: {'kernel': 'rbf', 'degree': 2, 'C': 1, 'gamma': 0.01, 'coef0': -10}
Best Model Confusion Matrix: [[102, 2], [1, 52]]

Second Best Model Score 0.9746835443037974
Second Best Model Params: {'kernel': 'poly', 'degree': 3, 'C': 0.01, 'gamma': 0.01, 'coef0': -10}
Second Best Model Confusion Matrix: [[102, 3], [1, 52]]


Process finished with exit code 0
```

Best:

[102, 2]

[1, 52]

Second Best:

[102, 3]

[1, 52]

The best model committed 2 false negatives and 1 false positive, while the second best model committed 3 false negatives and 1 false positive. Therefore, it seems as if both models were balanced in the type of errors they committed, but the rbf kernel was slightly more accurate.

After running this test multiple times, it almost seemed as if the best and second best kernels were chosen arbitrarily. Therefore, I am concluding that there is not any particular type of kernel that is preferable for this type of data.

Part D – Short Questions

Did Cross Validation help your accuracy of model over unseen data?

Cross validation did help with the accuracy of my model selection process. Each model was split into three separate sets and test scores were recorded for each one. After that, the average of those scores were used to determine the actual test score for the model. This process helps to illustrate how each model would generalize to independent datasets and will most definitely improve the model selection process.

What is a confusion matrix?

A confusion matrix is a method of scoring a model. The matrix is set up with counts of true positives, false positives, false negatives, and true negatives. The sum of the true positives and true negatives can then be divided by the total number of records to yield the accuracy of the model under question.