## Approach 1: Traditional Machine Learning

**Description of Data**

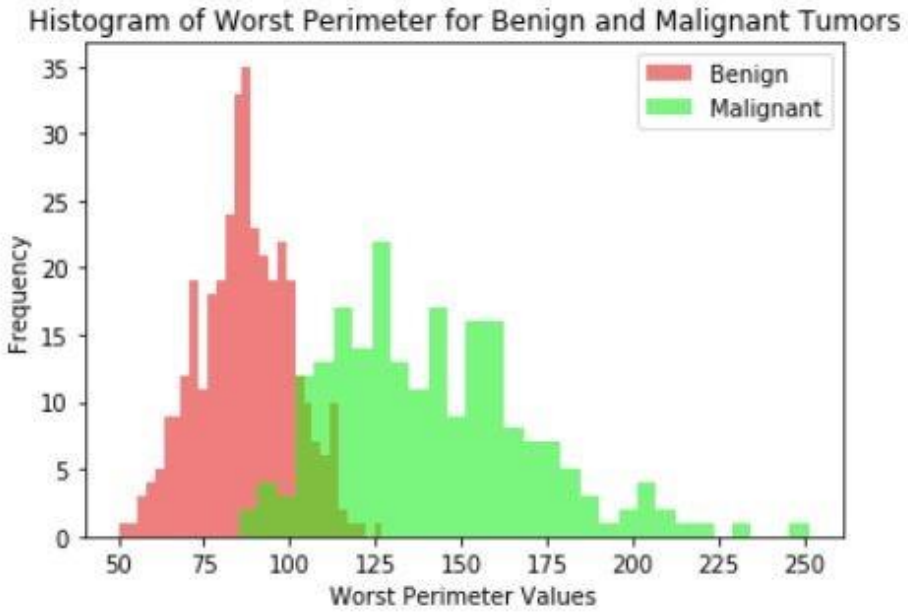
The dataset represents features of a set of cell nuclei of digitized images of fine needle aspiration (FNA) done on breast tissue. The dataset used in this study consists of 32 attributes in the UCI Machine Learning store. Some of the features included in the dataset are radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, fractal size for each cell nucleus.

**Summary of Data Statistics**

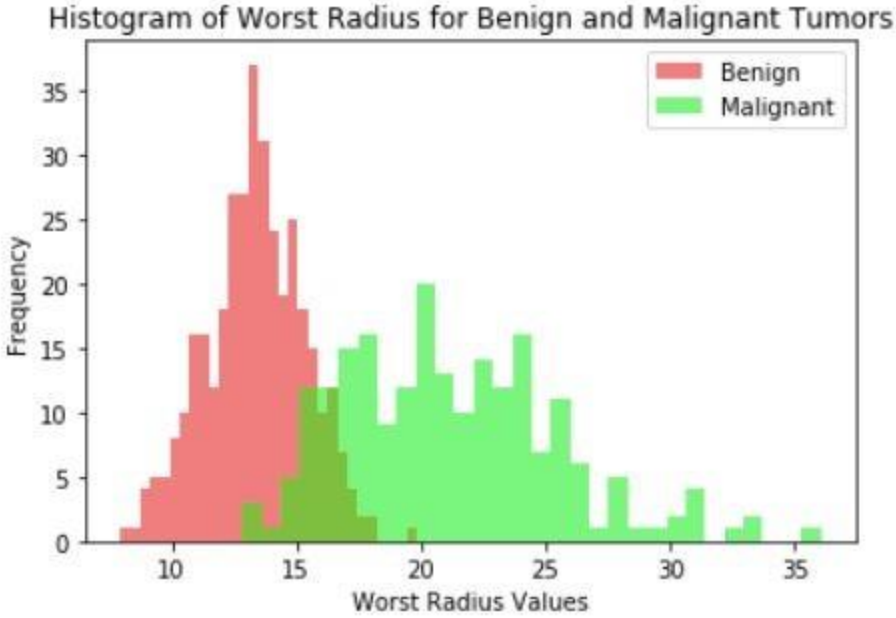
We gathered statistics on three parameters, worst radius (wradius), worst texture (wtexture), and worst perimeter (wperimeter). Based on the summary statistics, the worst perimeter had the largest standard deviation although this would be more meaningful if we would have normalized the data. We also generated histograms and box plots of these (below). It is worth noting that wtexture had the greatest overlap in the histogram.

Summary statistics on wradius, wtexture, wperimeter.

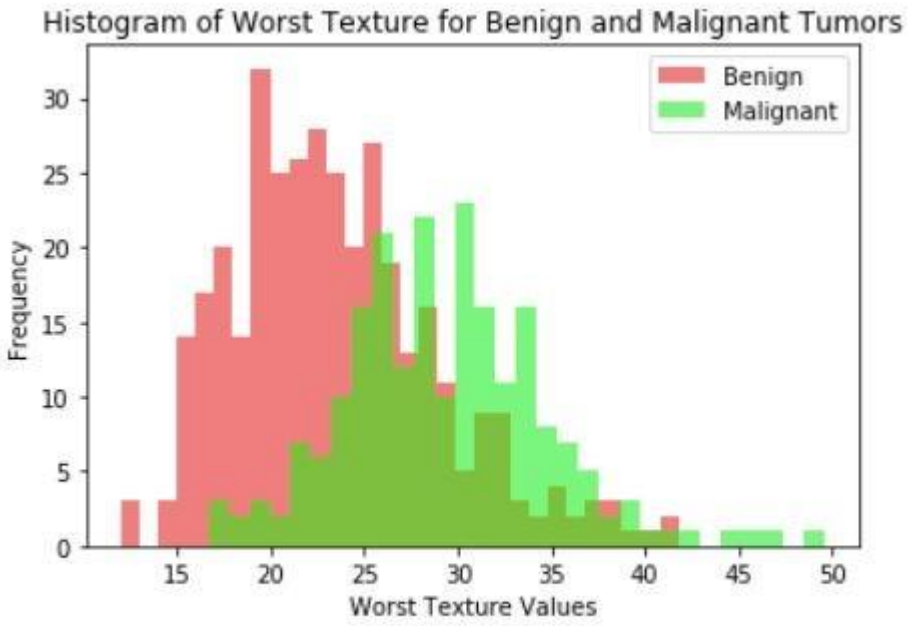
| Statistics | wradius | wtexture | wperimeter |
|---|---|---|---|
| Mean | 16.269190 | 25.677223 | 107.261213 |
| Standard Deviation | 4.833242 | 6.146258 | 33.602542 |
| Minimum | 7.930000 | 12.020000 | 50.410000 |
| 25% | 13.010000 | 21.080000 | 84.110000 |
| 50% | 14.970000 | 25.410000 | 97.660000 |
| 75% | 18.790000 | 29.720000 | 125.400000 |
| Maximum | 36.040000 | 49.540000 | 251.200000 |



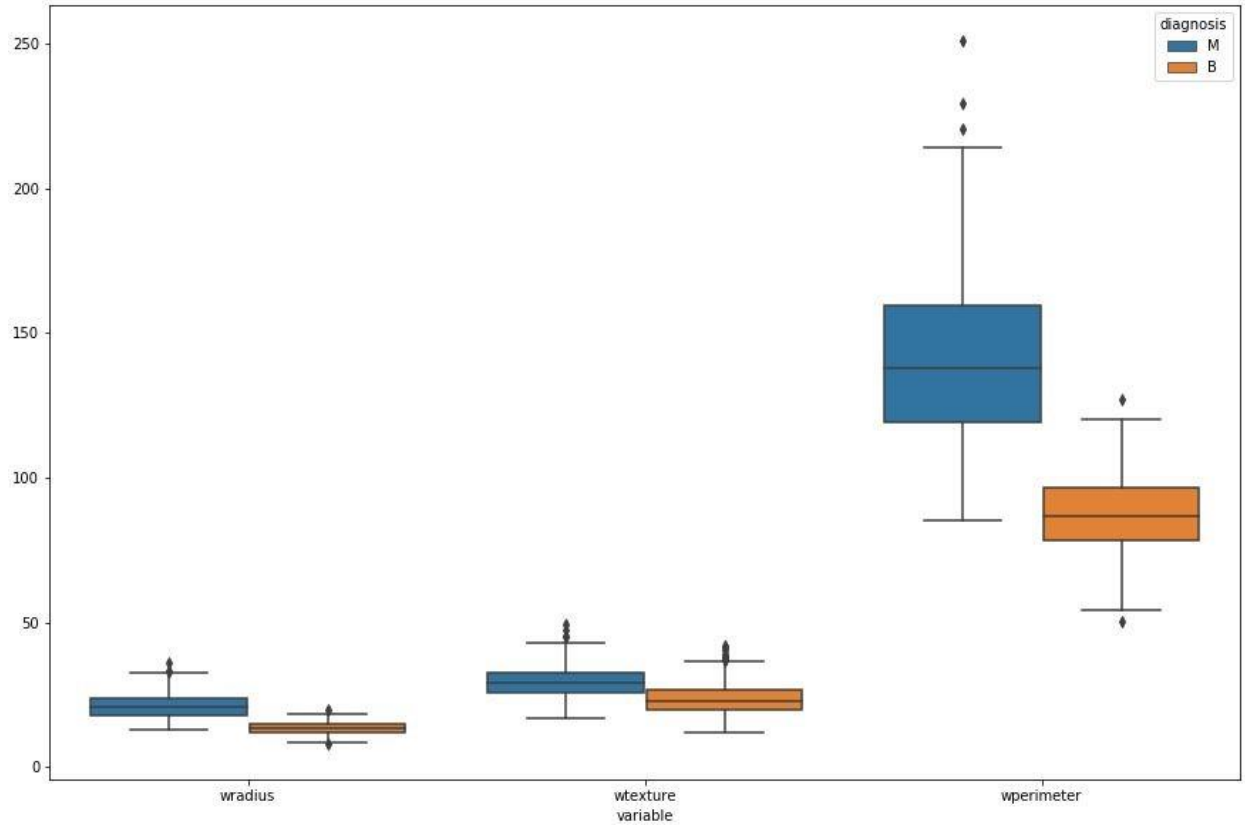

```
Most frequent malignant worst perimeter is:  123.85666666666665
```




```
Most frequent malignant radius mean is:  19.8
```




```
Most frequent malignant worst texture is:  29.818
```




**Description and Selection Process of the Best Features**

We used a heatmap with Pearson's correlation and removed all parameters that had a correlation value of less than .5. This amounted to 15 parameters which are listed below with

their correlations to the classifier. We decided that because we were left with only 15 parameters this was enough to start with creating the classifiers.

```
mtexture              0.415185
msmoothness           0.358560
msymmetry             0.330499
mfractaldimension     0.012838
setexture             0.008303
sesmoothness          0.067016
secompactness         0.292999
seconcavity           0.253730
seconcavepoints       0.408042
sesymmetry            0.006522
sefractaldimension    0.077972
wtexture              0.456903
wsmoothness           0.421465
wsymmetry             0.416294
wfractaldimension     0.323872
Name: diagnosis, dtype: float64
```

**Description and Comparison of the Classifier Methods**

The methods that we used were Random Forest and Support Vector Machine.

Random Forest

The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction is more accurate than that of any individual tree. After each tree is built, all of the data are run down the tree, and proximities are computed for each pair of cases. If two cases occupy the same terminal node, their proximity is increased by one. At the end of the run, the proximities are normalized by dividing by the number of trees. Proximities are used in replacing missing data, locating outliers, and producing illuminating low-dimensional views of the data. Random forests are extremely flexible and have very high accuracy.

Support Vector Machine

A support vector machine classifier is created by computing the most optimal hyperplane between the training data points. The best types of datasets to use these are generally small and with minimal noise. This is because the training time can be inefficient if there is a large amount of data. Similar to Neural Networks, these can also perform well with non linear boundaries.

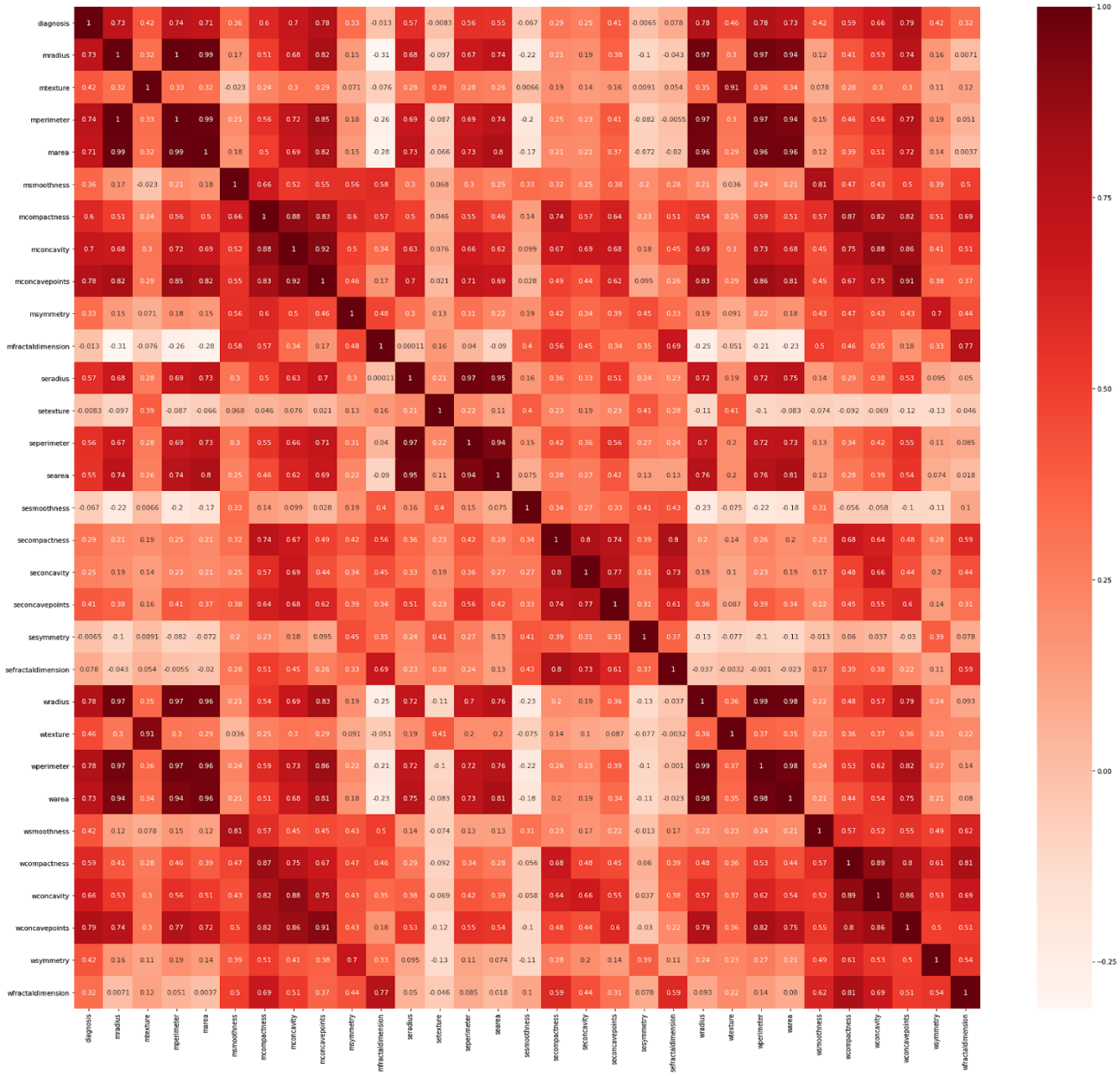
Comparison of Methods

Random forests are generally used for multiclass problems whereas SVMs are used for those with two classes (such as this problem). Random forests are usually preferred if there are many rows to process as the training complexity for SVMs are higher than the training time complexity for random forests. Both methods are sensitive to parameter input.

**Results**

Both the Random Forest and Support Vector Machine gave an accuracy of above 90%.

Pearson's Correlation Matrix



## Support Vector Machine

In [3]:
```
# 1.3 Support Vector Machine

# create test and training variables
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=21)

svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)
y_pred = svclassifier.predict(X_test)

print("Support Vector Machine Results")
print(classification_report(y_test,y_pred))

cm = pd.DataFrame(confusion_matrix(y_test, y_pred), columns=["T", "F"], index=["P", "N"])
print("Accuracy = %.2f%%" % ((cm.iloc[1, 1] + cm.iloc[0, 0]) / cm.values.sum() * 100))
cm
```

```
Support Vector Machine Results
              precision    recall  f1-score   support

           0       0.96      0.92      0.94        73
           1       0.86      0.93      0.89        41

   micro avg       0.92      0.92      0.92       114
   macro avg       0.91      0.92      0.92       114
weighted avg       0.92      0.92      0.92       114

Accuracy = 92.11%
```

Out[3]:

| | T | F |
|---|---|---|
| **P** | 67 | 6 |
| **N** | 3 | 38 |

## Random Forest

In [4]:
```
# 1.4 Random Forest

# Import the model we are using
from sklearn.ensemble import RandomForestClassifier
# Instantiate model with 1000 decision trees
rf = RandomForestClassifier(n_estimators = 1000, random_state = 42)
# Train the model on training data
rf.fit(X_train, y_train);
y_pred = rf.predict(X_test)

print("Random Forest Results")
cm = pd.DataFrame(confusion_matrix(y_test, y_pred), columns=["T", "F"], index=["P", "N"])
a = (cm.values.sum())
b = (cm.iloc[1, 1] + cm.iloc[0, 0])
print("Accuracy = %.2f%%" % ((b / a) * 100))
print(cm)
```

```
Random Forest Results
107
114
Accuracy = 93.86%
    T   F
P  69   4
N   3  38
```

## Approach 2: Deep Learning

### Description of Data

The dataset used contained breast histopathy images from a Kaggle dataset (https://www.kaggle.com/paultimothymooney/predicting-idc-in-breast-cancer-histology-images/data). It contained 162 images of slide images of breast cancer imaging split up into 50x50x3 image patches. The images are categorized by those that contain Invasive Ductal Carcinoma (IDC) and those that do not.

### Description of Used Method

Tensorflow was used on Google Colab. Because there were so many images (277,524 patches) we decided only to use 202 images to train (half with IDC and half without) and 20 images to test. We used each patch of an image as if it were a unique sample and did not take into account that issues may arise because many of these patches are from the same patients. We used a Convolution Neural Network for this problem with 10 epochs.

### Results

After running the code several times, a range of accuracy between 71%-81.82% was output. There were never any false positives produced.

```
Train on 202 samples
Epoch 1/10
202/202 [==============================] - 0s 2ms/sample - loss: 0.6914 - acc: 0.4950
Epoch 2/10
202/202 [==============================] - 0s 691us/sample - loss: 0.6447 - acc: 0.5594
Epoch 3/10
202/202 [==============================] - 0s 687us/sample - loss: 0.5412 - acc: 0.7228
Epoch 4/10
202/202 [==============================] - 0s 712us/sample - loss: 0.5095 - acc: 0.7624
Epoch 5/10
202/202 [==============================] - 0s 695us/sample - loss: 0.4386 - acc: 0.7970
Epoch 6/10
202/202 [==============================] - 0s 700us/sample - loss: 0.4131 - acc: 0.7921
Epoch 7/10
202/202 [==============================] - 0s 706us/sample - loss: 0.3564 - acc: 0.8564
Epoch 8/10
202/202 [==============================] - 0s 712us/sample - loss: 0.4973 - acc: 0.7525
Epoch 9/10
202/202 [==============================] - 0s 687us/sample - loss: 0.4722 - acc: 0.7970
Epoch 10/10
202/202 [==============================] - 0s 702us/sample - loss: 0.5191 - acc: 0.7525
    T  F
P  11  0
N   4  7
Accuracy = 81.82%
```

## Discussion

### Difference between Approaches

The main distinction between machine learning and deep learning is that machine learning allows for a more hands-on approach because the parameters are determined without the use of a black box method. Furthermore, because there are more factors to process, the deep learning technique took longer to compute in this situation. However, using the machine learning technique on a smaller number of photos took longer. Although both machine learning classifiers had increased accuracy, it is impossible to conclude whether they are better for this goal because they were employed on separate datasets.