Final Project Submission

Please fill out:

- Student name: LaShanni Butler
- Student pace: Part time
- Scheduled project review date/time: 7/26/19 @ 1:45pm PST
- Instructor name: Jeff Herman
- Blog post URL:

Problem Statement and Goal: Analyze dataset to identify which patients have breast cancer and use ML and DL models to determine which model best predicts breast cancer malignancy.

1) Loading Libraries/dataset

In [1]:
```
#Importing libraries
import numpy as np # linear algebra
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from pandas.tools import plotting
from scipy import stats
plt.style.use("ggplot")

#Machine learning modeling & confusion matrices
from sklearn.model_selection import train_test_split
from sklearn.model_selection import train_test_split,cross_val_score,KFold,cr
from sklearn.metrics import accuracy_score, classification_report, precision_
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import ExtraTreesClassifier,RandomForestClassifier,AdaB
from sklearn.neighbors  import KNeighborsClassifier
from sklearn import svm,tree
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from sklearn.model_selection import GridSearchCV
from sklearn import tree
from sklearn.neighbors import NearestNeighbors
from sklearn.exceptions import DataConversionWarning
import warnings
warnings.filterwarnings(action='ignore', category=DataConversionWarning)
warnings.filterwarnings('ignore')

#Deep learning modeling
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from keras.utils import np_utils
from keras.layers import Dropout

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import Pipeline
```

```
Using TensorFlow backend.
```

In [2]:
```
#loading Breast cancer dataset from UCI
df = pd.read_csv('data.csv')
```

In [3]:
```
#Taking a look at the dataset
df.head()
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
0	842302	M	17.99	10.38	122.80	1001.0	0
1	842517	M	20.57	17.77	132.90	1326.0	0
2	84300903	M	19.69	21.25	130.00	1203.0	0
3	84348301	M	11.42	20.38	77.58	386.1	0
4	84358402	M	20.29	14.34	135.10	1297.0	0

5 rows × 33 columns

In [4]:
```
df.tail()
```

Out[4]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
564	926424	M	21.56	22.39	142.00	1479.0	0
565	926682	M	20.13	28.25	131.20	1261.0	0
566	926954	M	16.60	28.08	108.30	858.1	0
567	927241	M	20.60	29.33	140.10	1265.0	0
568	92751	B	7.76	24.54	47.92	181.0	0

5 rows × 33 columns

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
id                         569 non-null int64
diagnosis                  569 non-null object
radius_mean                569 non-null float64
texture_mean               569 non-null float64
perimeter_mean             569 non-null float64
area_mean                  569 non-null float64
smoothness_mean            569 non-null float64
compactness_mean           569 non-null float64
concavity_mean             569 non-null float64
concave points_mean        569 non-null float64
symmetry_mean              569 non-null float64
fractal_dimension_mean     569 non-null float64
radius_se                  569 non-null float64
texture_se                 569 non-null float64
perimeter_se               569 non-null float64
area_se                    569 non-null float64
smoothness_se              569 non-null float64
compactness_se             569 non-null float64
concavity_se               569 non-null float64
concave points_se          569 non-null float64
symmetry_se                569 non-null float64
fractal_dimension_se       569 non-null float64
radius_worst               569 non-null float64
texture_worst              569 non-null float64
perimeter_worst            569 non-null float64
area_worst                 569 non-null float64
smoothness_worst           569 non-null float64
compactness_worst          569 non-null float64
concavity_worst            569 non-null float64
concave points_worst       569 non-null float64
symmetry_worst             569 non-null float64
fractal_dimension_worst    569 non-null float64
Unnamed: 32                0 non-null float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

Column description names:

- id: ID number
- diagnosis: The diagnosis of breast tissues (M = malignant, B = benign)
- radius_mean: mean of distances from center to points on the perimeter
- texture_mean: standard deviation of gray-scale values
- perimeter_mean: mean size of the core tumor
- area_mean
- smoothness_mean: mean of local variation in radius lengths
- compactness_mean: mean of perimeter^2 / area - 1.0
- concavity_mean: mean of severity of concave portions of the contour
- concave points_mean: mean for number of concave portions of the contour
- symmetry_mean

- fractal_dimension_mean: mean for "coastline approximation" - 1
- radius_se: standard error for the mean of distances from center to points on the perimeter
- texture_se: standard error for standard deviation of gray-scale values
- perimeter_se
- area_se
- smoothness_se: standard error for local variation in radius lengths
- compactness_se: standard error for perimeter^2 / area - 1.0
- concavity_se: standard error for severity of concave portions of the contour
- concave points_se: standard error for number of concave portions of the contour
- symmetry_se
- fractal_dimension_se: standard error for "coastline approximation" - 1
- radius_worst: "worst" or largest mean value for mean of distances from center to points on the perimeter
- texture_worst: "worst" or largest mean value for standard deviation of gray-scale values
- perimeter_worst
- area_worst
- smoothness_worst: "worst" or largest mean value for local variation in radius lengths
- compactness_worst: "worst" or largest mean value for perimeter^2 / area - 1.0
- concavity_worst:"worst" or largest mean value for severity of concave portions of the contour
- concave points_worst: "worst" or largest mean value for number of concave portions of the contour
- symmetry_worst
- fractal_dimension_worst: "worst" or largest mean value for "coastline approximation" - 1

In [6]:
```
#count number of rows and columns in dataset
df.shape
```

Out[6]:
```
(569, 33)
```

There are 569 rows (or patients) and 33 columns with features of the dataset

In [7]:
```
#Count of the number of empty/missing values in each column
df.isna().sum()
```

Out[7]:
```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
Unnamed: 32              569
dtype: int64
```

I noticed there is a column at the end that has null values, so I will drop that column. Additionally, I will address any other issues in the dataset that might affect the analyses.

In [8]:
```
#Dropping the Unnamed 32 column since there is missing data
#NOTE: This drops the column Unnamed
df = df.dropna(axis=1)
```

In [9]:
```
#Get the new count of the number of rows and columns
df.shape
```

Out[9]:
```
(569, 32)
```

In [10]:
```
#sanity check to see if columns were dropped
df.head()
```

Out[10]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
0	842302	M	17.99	10.38	122.80	1001.0	0
1	842517	M	20.57	17.77	132.90	1326.0	0
2	84300903	M	19.69	21.25	130.00	1203.0	0
3	84348301	M	11.42	20.38	77.58	386.1	0
4	84358402	M	20.29	14.34	135.10	1297.0	0

5 rows × 32 columns

2) EDA

In [11]:
```
df.columns
```

Out[11]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mea
n',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_s
e',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_s
e',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

In [12]:
```
df.dtypes
```

Out[12]:
```
id                         int64
diagnosis                 object
radius_mean              float64
texture_mean             float64
perimeter_mean           float64
area_mean                float64
smoothness_mean          float64
compactness_mean         float64
concavity_mean           float64
concave points_mean      float64
symmetry_mean            float64
fractal_dimension_mean   float64
radius_se                float64
texture_se               float64
perimeter_se             float64
area_se                  float64
smoothness_se            float64
compactness_se           float64
concavity_se             float64
concave points_se        float64
symmetry_se              float64
fractal_dimension_se     float64
radius_worst             float64
texture_worst            float64
perimeter_worst          float64
area_worst               float64
smoothness_worst         float64
compactness_worst        float64
concavity_worst          float64
concave points_worst     float64
symmetry_worst           float64
fractal_dimension_worst  float64
dtype: object
```

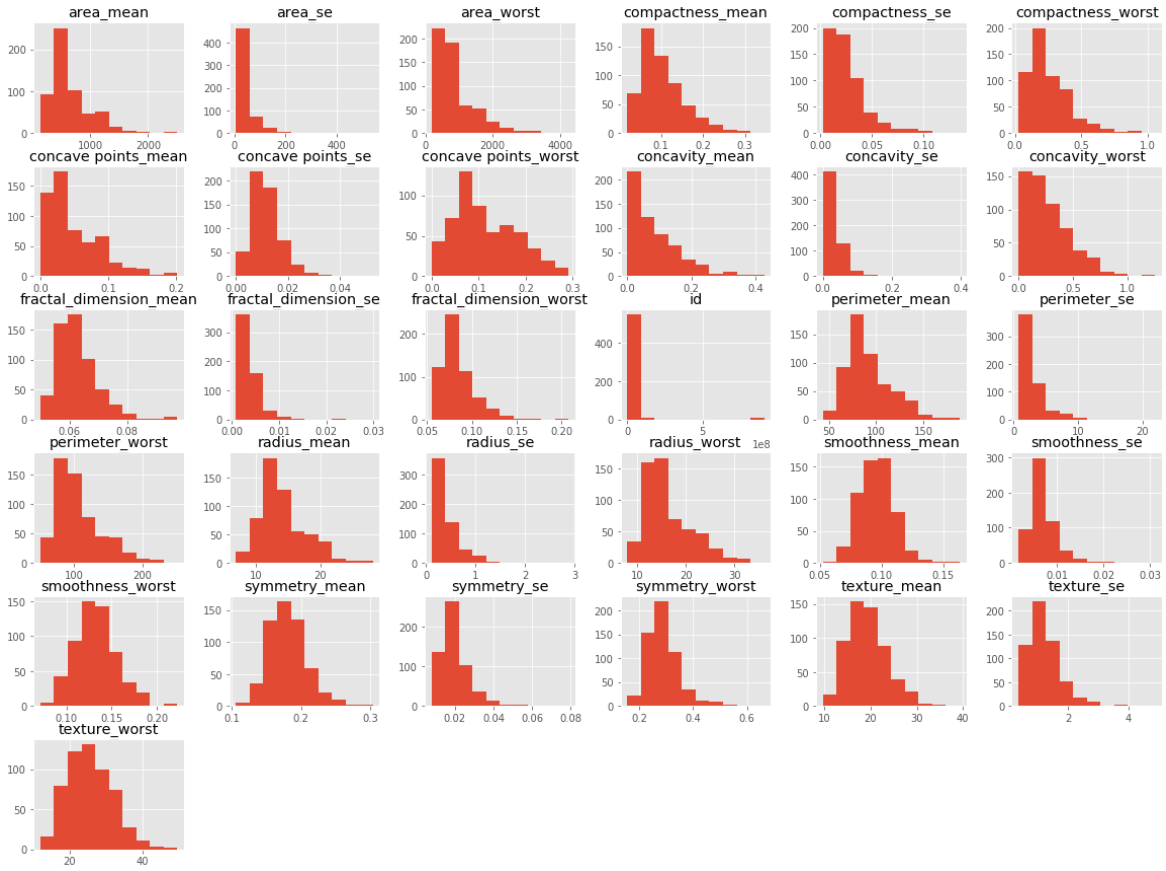
In [13]:
```
#Getting summary stats for dataset, and nothing stands out in terms of outlie
df.describe()
```

Out[13]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_n
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.00
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.09
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.01
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.05
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.08
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.09
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.10
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.16

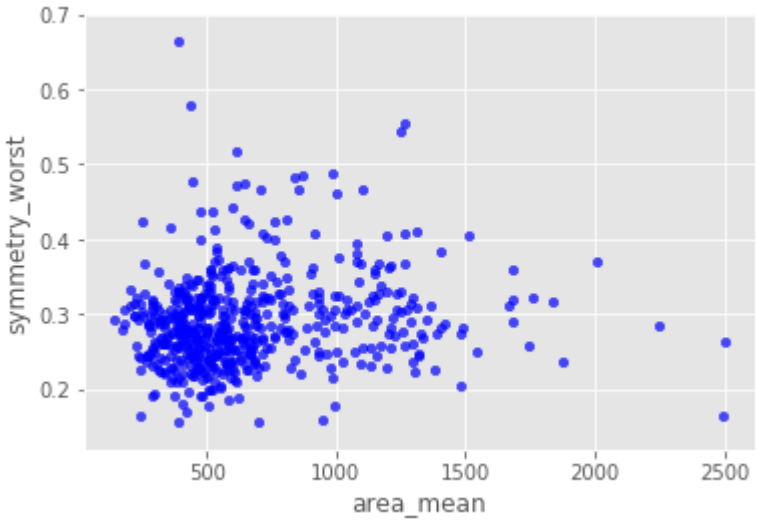
8 rows × 31 columns

In [14]:
```
#Generating historgrams to visualize the features in dataset
num_bins = 10
df.hist(bins=num_bins, figsize=(20,15))
plt.savefig('hr_histogram_plots')
plt.show()
```



In [15]:
```
df.plot(kind='Scatter', x='area_mean', y='symmetry_worst', alpha=0.7, color='
```

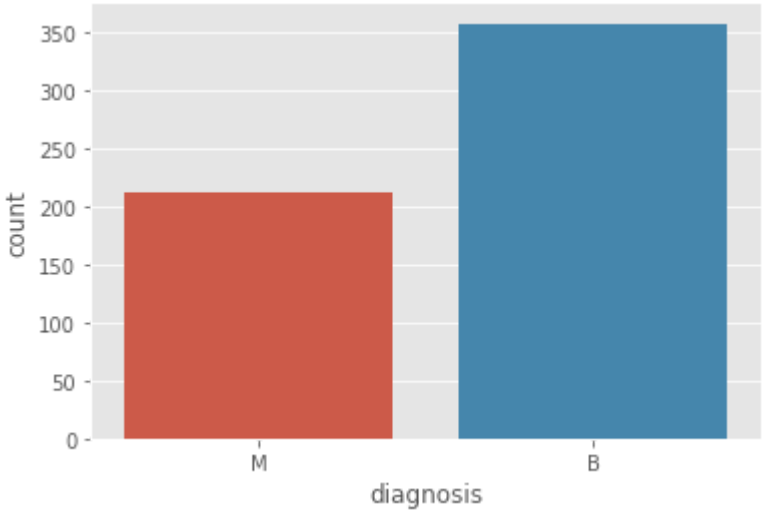
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x259f41f93c8>



In [16]:
```
#Get a count of the number of Malignant (M) (harmful) or Benign (B) cells (no
df['diagnosis'].value_counts()
```

Out[16]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

In [17]:
```
#Visualize this count
sns.countplot(df['diagnosis'],label='Count')
plt.figure(figsize=(8,10));
```



<Figure size 576x720 with 0 Axes>

In [18]:
```
#Visualize this count
#Transform/ Encode the column diagnosis
dictionary = {'M':1, 'B':0}#Create a dictionary file
df.diagnosis = [dictionary[item] for item in df.diagnosis] #Change all 'M' to
```

In [19]:
```
#Encoding categorical data values (Transforming categorical data/ Strings to
from sklearn.preprocessing import LabelEncoder
labelencoder_Y = LabelEncoder()
df.iloc[:,1]= labelencoder_Y.fit_transform(df.iloc[:,1].values)
print(labelencoder_Y.fit_transform(df.iloc[:,1].values))
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 1
 0 1 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0
 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 1
 0 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 1 0 1 0 0 1 0 1 1 1 1 0 0 1 1 0 0
 0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0
 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1
 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0
 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 0
 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 1
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 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 0]
```

In [20]:
```
#Running a scatterplot  in which one variable in the same data row is matched
sns.pairplot(df, hue='diagnosis')
```

Out[20]: <seaborn.axisgrid.PairGrid at 0x259f3382860>

In [21]:
```
#Scatterplot  with just a sample of the columns
sns.pairplot(df.iloc[:,1:7], hue='diagnosis') #plot a sample of the columns
```

Out[21]: <seaborn.axisgrid.PairGrid at 0x2599a314668>

At quick glance: The malignant tumor were much larger in terms of perimeter, area, radius than the benign ones. Also the malignant tumors generally had more texture to them.

In [22]:
```
#Print the first 5 rows of the new modified data set
#double checking that 1 and 0 are now in the diagnosis column
df.head(5)
```

Out[22]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
0	842302	1	17.99	10.38	122.80	1001.0	0
1	842517	1	20.57	17.77	132.90	1326.0	0
2	84300903	1	19.69	21.25	130.00	1203.0	0
3	84348301	1	11.42	20.38	77.58	386.1	0
4	84358402	1	20.29	14.34	135.10	1297.0	0

5 rows × 32 columns

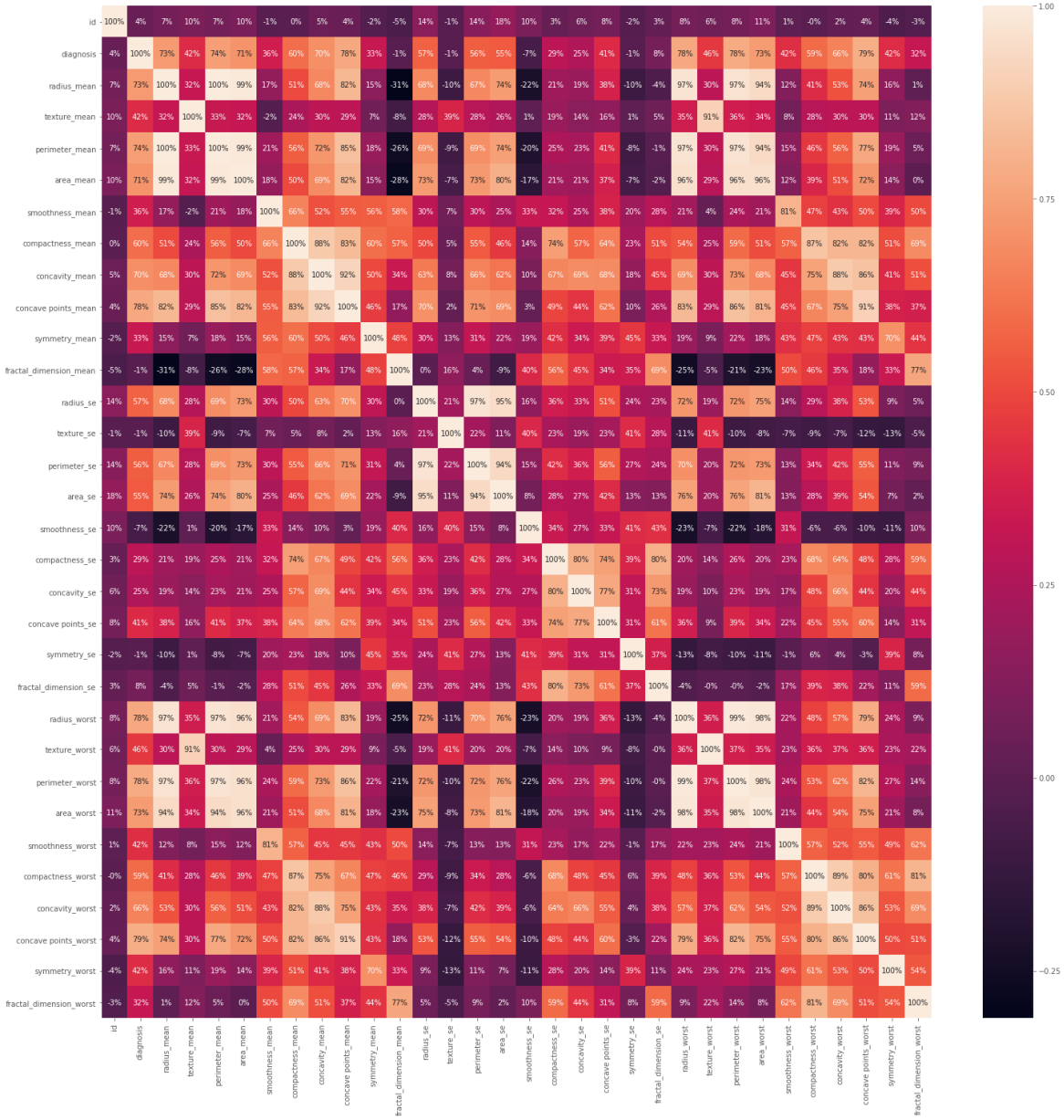
In [23]:
```
#Get the correlation of the columns
df.corr()
```

Out[23]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	ar
id	1.000000	0.039769	0.074626	0.099770	0.073159	
diagnosis	0.039769	1.000000	0.730029	0.415185	0.742636	
radius_mean	0.074626	0.730029	1.000000	0.323782	0.997855	
texture_mean	0.099770	0.415185	0.323782	1.000000	0.329533	
perimeter_mean	0.073159	0.742636	0.997855	0.329533	1.000000	
area_mean	0.096893	0.708984	0.987357	0.321086	0.986507	
smoothness_mean	-0.012968	0.358560	0.170581	-0.023389	0.207278	
compactness_mean	0.000096	0.596534	0.506124	0.236702	0.556936	
concavity_mean	0.050080	0.696360	0.676764	0.302418	0.716136	
concave points_mean	0.044158	0.776614	0.822529	0.293464	0.850977	
symmetry_mean	-0.022114	0.330499	0.147741	0.071401	0.183027	
fractal_dimension_mean	-0.052511	-0.012838	-0.311631	-0.076437	-0.261477	-
radius_se	0.143048	0.567134	0.679090	0.275869	0.691765	
texture_se	-0.007526	-0.008303	-0.097317	0.386358	-0.086761	-
perimeter_se	0.137331	0.556141	0.674172	0.281673	0.693135	
area_se	0.177742	0.548236	0.735864	0.259845	0.744983	
smoothness_se	0.096781	-0.067016	-0.222600	0.006614	-0.202694	-
compactness_se	0.033961	0.292999	0.206000	0.191975	0.250744	
concavity_se	0.055239	0.253730	0.194204	0.143293	0.228082	
concave points_se	0.078768	0.408042	0.376169	0.163851	0.407217	
symmetry_se	-0.017306	-0.006522	-0.104321	0.009127	-0.081629	-
fractal_dimension_se	0.025725	0.077972	-0.042641	0.054458	-0.005523	-
radius_worst	0.082405	0.776454	0.969539	0.352573	0.969476	
texture_worst	0.064720	0.456903	0.297008	0.912045	0.303038	
perimeter_worst	0.079986	0.782914	0.965137	0.358040	0.970387	
area_worst	0.107187	0.733825	0.941082	0.343546	0.941550	
smoothness_worst	0.010338	0.421465	0.119616	0.077503	0.150549	
compactness_worst	-0.002968	0.590998	0.413463	0.277830	0.455774	
concavity_worst	0.023203	0.659610	0.526911	0.301025	0.563879	
concave points_worst	0.035174	0.793566	0.744214	0.295316	0.771241	
symmetry_worst	-0.044224	0.416294	0.163953	0.105008	0.189115	
fractal_dimension_worst	-0.029866	0.323872	0.007066	0.119205	0.051019	

32 rows × 32 columns

In [24]:
```
#Visualize correlation via heatmap
plt.figure(figsize=(25,25))  #change the size of the heatmap
sns.heatmap(df.corr(), annot=True, fmt='.0%');
```



Using diagnosis as the feature, I see there are a few features that don't strongly correlate:

- fractal dimension mean
- texture se
- smoothness se
- symmetry se

So I think dropping them will further help with the predictive modeling

In [25]:
```
df.drop(['fractal_dimension_mean','texture_se','smoothness_se','symmetry_se']
```

3) Predictive Modeling

Splitting data into testing and training set

In [26]:
```
#Split the data into independent 'X' and dependent 'Y' variables
#Independent data will tell me the features that can detect if patient has ca
#Dependent data will tell me if a patient has cancer or not
X = df.iloc[:, 2:31].values # I started from index  2 to 31, essentially remo
Y = df.iloc[:, 1].values #Get the target variable 'diagnosis' located at inde
```

In [81]:
```
# Split the dataset into 75% Training set and 25% Testing set
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, r
```

In [82]:
```
# Scale the data to bring all features to the same level of magnitude
# This means the data will be within a specific range for example 0 -100 or 0

#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

3a) Looking at Machine Learning Models

In [83]:
```
#defining confusion matrix plotting function
#sourced from: http://scikit-learn.org/stable/auto_examples/model_selection/p

from sklearn.metrics import confusion_matrix
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):


    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)



    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment='center',
                 color='white' if cm[i, j] > thresh else 'black')


    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
#Using Begign and Malignant for 0 and 1
class_names = ['Benign','Malignant']
```

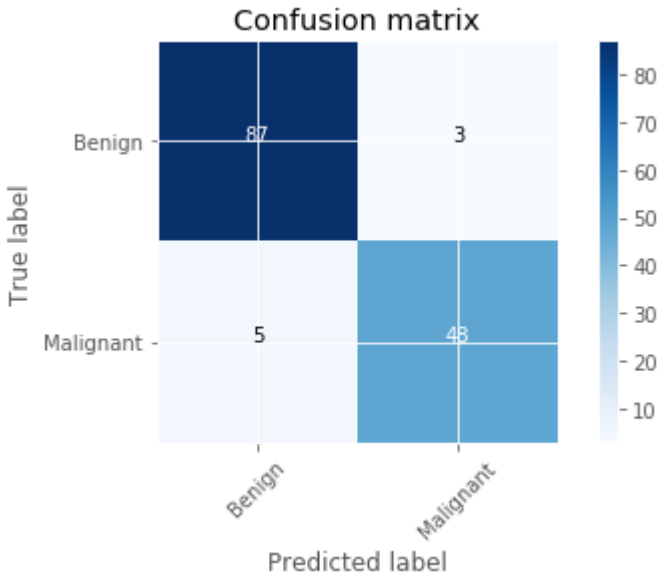
Predictive Models used:

- kNN
- Logistic Regression
- SVM
- Decision Tree
- Random Forest

In [84]:

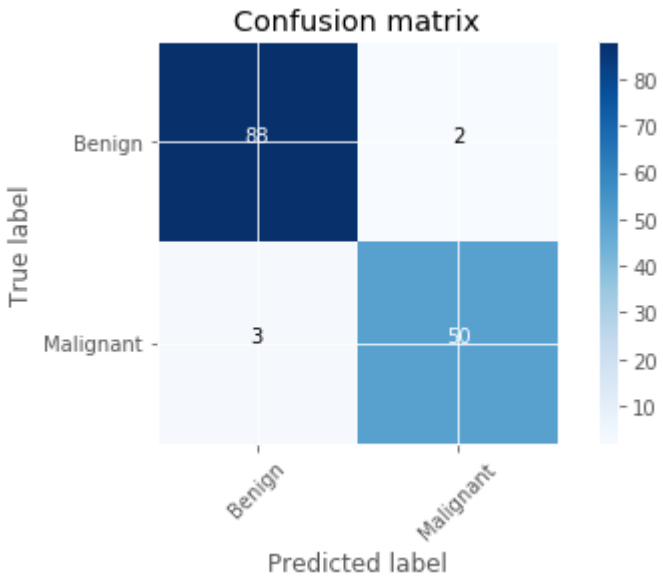
```
# Defining the kNNClassifier with 5 neighbors
knn = KNeighborsClassifier(n_neighbors = 5)

#Fitting the classifier to the training set
knn.fit(X_train,Y_train)
print ('kNN Accuracy is %2.2f' % accuracy_score(Y_test, knn.predict(X_test)))

#The cross validation score is obtained for kNN using 10 folds
#Cross-validation is used to split the data into training and test sets to ev

score_knn = cross_val_score(knn, X, Y, cv=10).mean()
print('Cross Validation Score = %2.2f' % score_knn)
y_pred= knn.predict(X_test)
print(classification_report(Y_test, y_pred))

#Defining the confusion matrix
cm = confusion_matrix(Y_test,y_pred)

#Plotting the confusion matrix
plot_confusion_matrix(cm, classes=class_names, title='Confusion matrix')
```

```
kNN Accuracy is 0.94
Cross Validation Score = 0.93
              precision    recall  f1-score   support

           0       0.95      0.97      0.96        90
           1       0.94      0.91      0.92        53

   micro avg       0.94      0.94      0.94       143
   macro avg       0.94      0.94      0.94       143
weighted avg       0.94      0.94      0.94       143
```



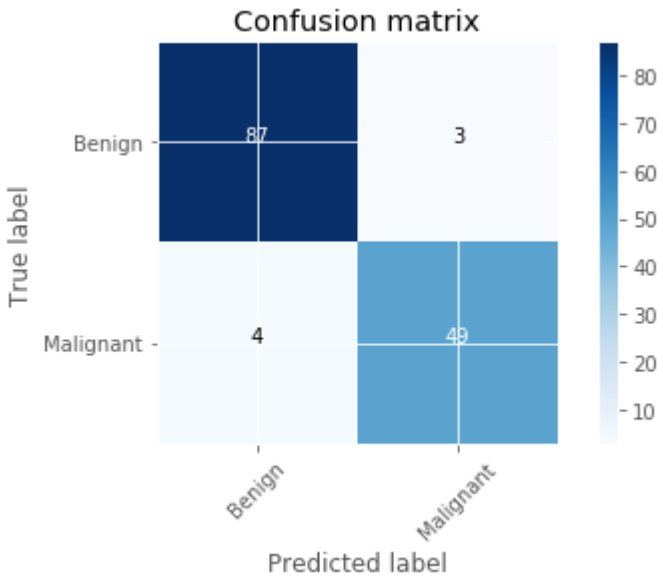
In [85]:
```
#Logistic Regression Classifier
LR = LogisticRegression()
LR.fit(X_train,Y_train)
print ('Logistic Accuracy is %2.2f' % accuracy_score(Y_test, LR.predict(X_tes
score_LR = cross_val_score(LR, X, Y, cv=10).mean()
print('Cross Validation Score = %2.2f' % score_LR)
y_pred = LR.predict(X_test)
print(classification_report(Y_test, y_pred))
# Confusion matrix for LR
cm = confusion_matrix(Y_test,y_pred)
plot_confusion_matrix(cm, classes=class_names, title='Confusion matrix')
```

```
Logistic Accuracy is 0.97
Cross Validation Score = 0.95
              precision    recall  f1-score   support

           0       0.97      0.98      0.97        90
           1       0.96      0.94      0.95        53

   micro avg       0.97      0.97      0.97       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.96      0.97      0.96       143
```



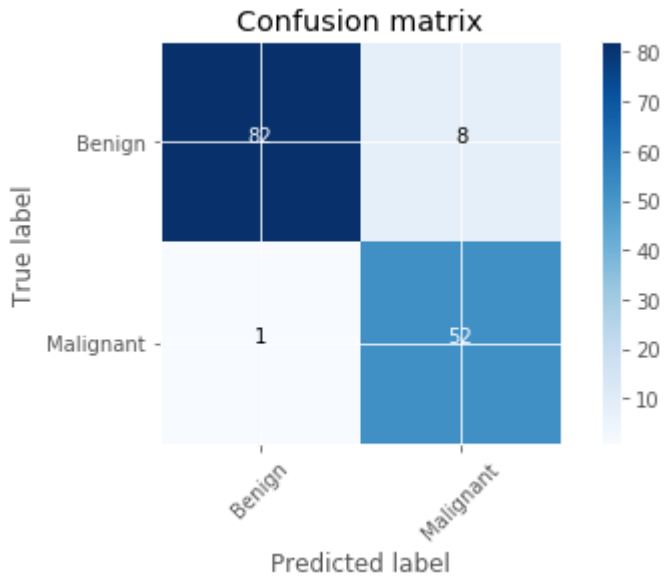
In [86]:
```
#SVM Classifier
SVM = svm.SVC()
SVM.fit(X_train, Y_train)
print ('SVM Accuracy is %2.2f' % accuracy_score(Y_test, SVM.predict(X_test)))
score_svm = cross_val_score(SVM, X, Y, cv=10).mean()
print('Cross Validation Score = %2.2f' % score_svm)
y_pred = SVM.predict(X_test)
print(classification_report(Y_test,y_pred))
#Confusion matrix for SVM
cm = confusion_matrix(Y_test,y_pred)
plot_confusion_matrix(cm, classes=class_names, title='Confusion matrix')
```

```
SVM Accuracy is 0.95
Cross Validation Score = 0.63
              precision    recall  f1-score   support

           0       0.96      0.97      0.96        90
           1       0.94      0.92      0.93        53

   micro avg       0.95      0.95      0.95       143
   macro avg       0.95      0.95      0.95       143
weighted avg       0.95      0.95      0.95       143
```



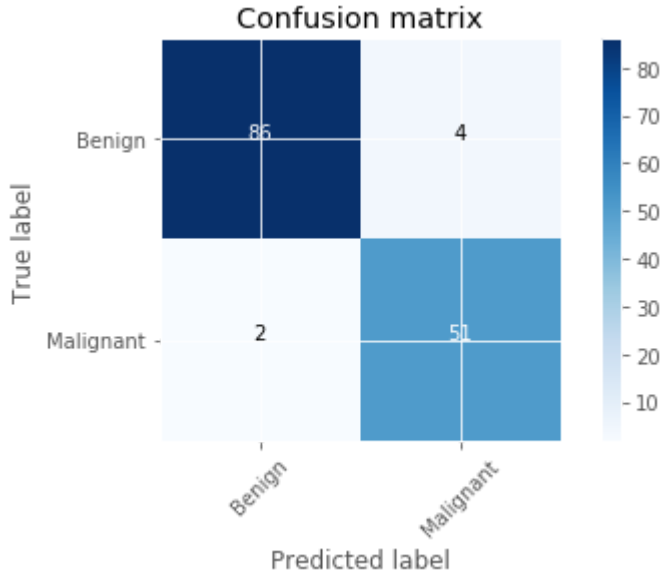
In [87]:
```
# Decision Tree Classifier
DT = tree.DecisionTreeClassifier(random_state = 0,class_weight='balanced',
    min_weight_fraction_leaf=0.01)
DT = DT.fit(X_train,Y_train)
print ('Decision Tree Accuracy is %2.2f' % accuracy_score(Y_test, DT.predict(
score_DT = cross_val_score(DT, X, Y, cv=10).mean()
print('Cross Validation Score = %2.2f' % score_DT)
y_pred = DT.predict(X_test)
print(classification_report(Y_test, y_pred))
# Confusion Matrix for Decision Tree
cm = confusion_matrix(Y_test,y_pred)
plot_confusion_matrix(cm, classes=class_names, title='Confusion matrix')
```

```
Decision Tree Accuracy is 0.94
Cross Validation Score = 0.95
              precision    recall  f1-score   support

           0       0.99      0.91      0.95        90
           1       0.87      0.98      0.92        53

   micro avg       0.94      0.94      0.94       143
   macro avg       0.93      0.95      0.93       143
weighted avg       0.94      0.94      0.94       143
```



In [88]:
```
#Random Forest Classifier
rfc = RandomForestClassifier(n_estimators=1000, max_depth=None, min_samples_s
rfc.fit(X_train, Y_train)
print ('Random Forest Accuracy is %2.2f' % accuracy_score(Y_test, rfc.predict
score_rfc = cross_val_score(rfc, X, Y, cv=10).mean()
print('Cross Validation Score = %2.2f' % score_rfc)
y_pred = rfc.predict(X_test)
print(classification_report(Y_test,y_pred ))
#Confusion Matrix for Random Forest
cm = confusion_matrix(Y_test,y_pred)
plot_confusion_matrix(cm, classes=class_names, title='Confusion matrix')
```

```
Random Forest Accuracy is 0.96
Cross Validation Score = 0.96
              precision    recall  f1-score   support

           0       0.98      0.96      0.97        90
           1       0.93      0.96      0.94        53

   micro avg       0.96      0.96      0.96       143
   macro avg       0.95      0.96      0.96       143
weighted avg       0.96      0.96      0.96       143
```



So KNN, Logistic regression and SVM all showed 96% accuracy. A close second was Random Forest at 95%.

- K Nearest Neighbor: 94%
- Logistic regression: 97%
- Support Vector Machine: 95%
- Decision Tree: 94%
- Random Forest: 96%

3b) Now looking at Deep Learning Model

In [89]:
```
# Initialising the Artificial Neural Net (ANN)
classifier = Sequential()
```

In [90]:
```
# Adding the input layer and the first hidden layer
classifier.add(Dense(output_dim=16, init='uniform', activation='relu', input_
# Adding dropout to prevent overfitting
classifier.add(Dropout(p=0.1))
```

In [91]:
```
# Adding the second hidden layer
classifier.add(Dense(output_dim=16, init='uniform', activation='relu'))
# Adding dropout to prevent overfitting
classifier.add(Dropout(p=0.1))
```

In [92]:
```
# Adding the output layer
classifier.add(Dense(output_dim=1, init='uniform', activation='sigmoid'))
```

In [93]:
```
# Compiling the ANN
classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['ac
```

In [94]:
```
classifier.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 16)                432
_________________________________________________________________
dropout_5 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_8 (Dense)              (None, 16)                272
_________________________________________________________________
dropout_6 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 17
=================================================================
Total params: 721
Trainable params: 721
Non-trainable params: 0
_________________________________________________________________
```

In [95]:
```
# Fitting the ANN to the Training set
#I decided to run a few different epochs to determine which one would give th
#epoch=25
classifier.fit(X_train, Y_train, batch_size=100, epochs=25)
```

```
Epoch 1/25
426/426 [==============================] - 1s 2ms/step - loss: 0.6930 - ac
c: 0.5540
Epoch 2/25
426/426 [==============================] - 0s 42us/step - loss: 0.6915 - ac
c: 0.6620
Epoch 3/25
426/426 [==============================] - 0s 30us/step - loss: 0.6895 - ac
c: 0.6761
Epoch 4/25
426/426 [==============================] - 0s 27us/step - loss: 0.6866 - ac
c: 0.7207
Epoch 5/25
426/426 [==============================] - 0s 44us/step - loss: 0.6824 - ac
c: 0.7676
Epoch 6/25
426/426 [==============================] - 0s 34us/step - loss: 0.6763 - ac
c: 0.8216
Epoch 7/25
426/426 [==============================] - 0s 31us/step - loss: 0.6673 - ac
c: 0.8732
Epoch 8/25
426/426 [==============================] - 0s 37us/step - loss: 0.6543 - ac
c: 0.8967
Epoch 9/25
426/426 [==============================] - 0s 47us/step - loss: 0.6378 - ac
c: 0.9131
Epoch 10/25
426/426 [==============================] - 0s 35us/step - loss: 0.6147 - ac
c: 0.9202
Epoch 11/25
426/426 [==============================] - 0s 35us/step - loss: 0.5861 - ac
c: 0.9272
Epoch 12/25
426/426 [==============================] - 0s 36us/step - loss: 0.5496 - ac
c: 0.9413
Epoch 13/25
426/426 [==============================] - 0s 29us/step - loss: 0.5045 - ac
c: 0.9413
Epoch 14/25
426/426 [==============================] - 0s 42us/step - loss: 0.4623 - ac
c: 0.9413
Epoch 15/25
426/426 [==============================] - 0s 37us/step - loss: 0.4197 - ac
c: 0.9460
Epoch 16/25
426/426 [==============================] - 0s 35us/step - loss: 0.3724 - ac
c: 0.9460
Epoch 17/25
426/426 [==============================] - 0s 35us/step - loss: 0.3307 - ac
c: 0.9484
```

```
Epoch 18/25
426/426 [==============================] - 0s 30us/step - loss: 0.2989 - ac
c: 0.9484
Epoch 19/25
426/426 [==============================] - 0s 34us/step - loss: 0.2647 - ac
c: 0.9507
Epoch 20/25
426/426 [==============================] - 0s 42us/step - loss: 0.2395 - ac
c: 0.9507
Epoch 21/25
426/426 [==============================] - 0s 43us/step - loss: 0.2262 - ac
c: 0.9507
Epoch 22/25
426/426 [==============================] - 0s 40us/step - loss: 0.2019 - ac
c: 0.9554
Epoch 23/25
426/426 [==============================] - 0s 38us/step - loss: 0.1859 - ac
c: 0.9577
Epoch 24/25
426/426 [==============================] - 0s 34us/step - loss: 0.1723 - ac
c: 0.9601
Epoch 25/25
426/426 [==============================] - 0s 46us/step - loss: 0.1586 - ac
c: 0.9624
```

Out[95]: <keras.callbacks.History at 0x259a2065f98>

In [96]:
```
# Fitting the ANN to the Training set, epoch=50
classifier.fit(X_train, Y_train, batch_size=100, epochs=50)
```

```
Epoch 1/50
426/426 [==============================] - 0s 41us/step - loss: 0.1526 - ac
c: 0.9648
Epoch 2/50
426/426 [==============================] - 0s 41us/step - loss: 0.1375 - ac
c: 0.9648
Epoch 3/50
426/426 [==============================] - 0s 42us/step - loss: 0.1355 - ac
c: 0.9671
Epoch 4/50
426/426 [==============================] - 0s 31us/step - loss: 0.1272 - ac
c: 0.9671
Epoch 5/50
426/426 [==============================] - 0s 51us/step - loss: 0.1248 - ac
c: 0.9742
Epoch 6/50
426/426 [==============================] - 0s 41us/step - loss: 0.1253 - ac
c: 0.9742
Epoch 7/50
426/426 [==============================] - 0s 52us/step - loss: 0.1176 - ac
c: 0.9765
Epoch 8/50
426/426 [==============================] - 0s 37us/step - loss: 0.1098 - ac
c: 0.9789
Epoch 9/50
426/426 [==============================] - 0s 36us/step - loss: 0.1044 - ac
c: 0.9789
Epoch 10/50
426/426 [==============================] - 0s 34us/step - loss: 0.1015 - ac
c: 0.9789
Epoch 11/50
426/426 [==============================] - 0s 75us/step - loss: 0.0984 - ac
c: 0.9742
Epoch 12/50
426/426 [==============================] - 0s 64us/step - loss: 0.0994 - ac
c: 0.9789
Epoch 13/50
426/426 [==============================] - ETA: 0s - loss: 0.1079 - acc: 0.
970 - 0s 68us/step - loss: 0.0967 - acc: 0.9765
Epoch 14/50
426/426 [==============================] - 0s 37us/step - loss: 0.0915 - ac
c: 0.9789
Epoch 15/50
426/426 [==============================] - 0s 37us/step - loss: 0.0896 - ac
c: 0.9789
Epoch 16/50
426/426 [==============================] - 0s 36us/step - loss: 0.0880 - ac
c: 0.9789
Epoch 17/50
426/426 [==============================] - 0s 38us/step - loss: 0.0843 - ac
c: 0.9789
Epoch 18/50
426/426 [==============================] - 0s 45us/step - loss: 0.0857 - ac
```

```
c: 0.9812
Epoch 19/50
426/426 [==============================] - 0s 43us/step - loss: 0.0791 - ac
c: 0.9836
Epoch 20/50
426/426 [==============================] - 0s 40us/step - loss: 0.0836 - ac
c: 0.9789
Epoch 21/50
426/426 [==============================] - 0s 33us/step - loss: 0.0791 - ac
c: 0.9836
Epoch 22/50
426/426 [==============================] - 0s 31us/step - loss: 0.0747 - ac
c: 0.9836
Epoch 23/50
426/426 [==============================] - 0s 30us/step - loss: 0.0780 - ac
c: 0.9812
Epoch 24/50
426/426 [==============================] - 0s 49us/step - loss: 0.0703 - ac
c: 0.9859
Epoch 25/50
426/426 [==============================] - 0s 36us/step - loss: 0.0721 - ac
c: 0.9883
Epoch 26/50
426/426 [==============================] - 0s 28us/step - loss: 0.0734 - ac
c: 0.9859
Epoch 27/50
426/426 [==============================] - 0s 33us/step - loss: 0.0699 - ac
c: 0.9883
Epoch 28/50
426/426 [==============================] - 0s 31us/step - loss: 0.0696 - ac
c: 0.9883
Epoch 29/50
426/426 [==============================] - 0s 26us/step - loss: 0.0710 - ac
c: 0.9883
Epoch 30/50
426/426 [==============================] - 0s 33us/step - loss: 0.0715 - ac
c: 0.9883
Epoch 31/50
426/426 [==============================] - 0s 54us/step - loss: 0.0693 - ac
c: 0.9859
Epoch 32/50
426/426 [==============================] - 0s 37us/step - loss: 0.0701 - ac
c: 0.9883
Epoch 33/50
426/426 [==============================] - 0s 36us/step - loss: 0.0650 - ac
c: 0.9883
Epoch 34/50
426/426 [==============================] - 0s 40us/step - loss: 0.0702 - ac
c: 0.9883
Epoch 35/50
426/426 [==============================] - 0s 34us/step - loss: 0.0644 - ac
c: 0.9883
Epoch 36/50
426/426 [==============================] - 0s 34us/step - loss: 0.0644 - ac
c: 0.9883
Epoch 37/50
426/426 [==============================] - 0s 41us/step - loss: 0.0646 - ac
```

```
c: 0.9883
Epoch 38/50
426/426 [==============================] - 0s 35us/step - loss: 0.0666 - ac
c: 0.9859
Epoch 39/50
426/426 [==============================] - 0s 35us/step - loss: 0.0639 - ac
c: 0.9883
Epoch 40/50
426/426 [==============================] - 0s 43us/step - loss: 0.0641 - ac
c: 0.9883
Epoch 41/50
426/426 [==============================] - 0s 34us/step - loss: 0.0610 - ac
c: 0.9883
Epoch 42/50
426/426 [==============================] - 0s 25us/step - loss: 0.0591 - ac
c: 0.9883
Epoch 43/50
426/426 [==============================] - 0s 41us/step - loss: 0.0597 - ac
c: 0.9859
Epoch 44/50
426/426 [==============================] - 0s 37us/step - loss: 0.0572 - ac
c: 0.9883
Epoch 45/50
426/426 [==============================] - 0s 35us/step - loss: 0.0568 - ac
c: 0.9883
Epoch 46/50
426/426 [==============================] - 0s 34us/step - loss: 0.0607 - ac
c: 0.9883
Epoch 47/50
426/426 [==============================] - 0s 45us/step - loss: 0.0574 - ac
c: 0.9883
Epoch 48/50
426/426 [==============================] - 0s 46us/step - loss: 0.0605 - ac
c: 0.9883
Epoch 49/50
426/426 [==============================] - 0s 44us/step - loss: 0.0618 - ac
c: 0.9883
Epoch 50/50
426/426 [==============================] - 0s 41us/step - loss: 0.0530 - ac
c: 0.9906
```

Out[96]: <keras.callbacks.History at 0x2599f118e80>

In [97]:
```
# Fitting the ANN to the Training set, epoch=100
classifier.fit(X_train, Y_train, batch_size=100, epochs=100)
```

```
Epoch 1/100
426/426 [==============================] - 0s 37us/step - loss: 0.0551 - ac
c: 0.9883
Epoch 2/100
426/426 [==============================] - 0s 71us/step - loss: 0.0563 - ac
c: 0.9883
Epoch 3/100
426/426 [==============================] - 0s 50us/step - loss: 0.0592 - ac
c: 0.9883
Epoch 4/100
426/426 [==============================] - 0s 34us/step - loss: 0.0566 - ac
c: 0.9859
Epoch 5/100
426/426 [==============================] - 0s 29us/step - loss: 0.0548 - ac
c: 0.9906
Epoch 6/100
426/426 [==============================] - 0s 32us/step - loss: 0.0545 - ac
c: 0.9883
Epoch 7/100
426/426 [==============================] - 0s 44us/step - loss: 0.0598 - ac
c: 0.9883
Epoch 8/100
426/426 [==============================] - 0s 41us/step - loss: 0.0559 - ac
c: 0.9883
Epoch 9/100
426/426 [==============================] - 0s 29us/step - loss: 0.0523 - ac
c: 0.9906
Epoch 10/100
426/426 [==============================] - 0s 30us/step - loss: 0.0518 - ac
c: 0.9883
Epoch 11/100
426/426 [==============================] - 0s 35us/step - loss: 0.0477 - ac
c: 0.9906
Epoch 12/100
426/426 [==============================] - 0s 28us/step - loss: 0.0513 - ac
c: 0.9883
Epoch 13/100
426/426 [==============================] - 0s 43us/step - loss: 0.0564 - ac
c: 0.9883
Epoch 14/100
426/426 [==============================] - 0s 38us/step - loss: 0.0546 - ac
c: 0.9883
Epoch 15/100
426/426 [==============================] - 0s 34us/step - loss: 0.0510 - ac
c: 0.9906
Epoch 16/100
426/426 [==============================] - 0s 29us/step - loss: 0.0536 - ac
c: 0.9883
Epoch 17/100
426/426 [==============================] - 0s 55us/step - loss: 0.0524 - ac
c: 0.9883
Epoch 18/100
426/426 [==============================] - 0s 45us/step - loss: 0.0543 - ac
```

```
c: 0.9859
Epoch 19/100
426/426 [==============================] - 0s 36us/step - loss: 0.0512 - ac
c: 0.9906
Epoch 20/100
426/426 [==============================] - 0s 29us/step - loss: 0.0522 - ac
c: 0.9883
Epoch 21/100
426/426 [==============================] - 0s 55us/step - loss: 0.0548 - ac
c: 0.9906
Epoch 22/100
426/426 [==============================] - 0s 48us/step - loss: 0.0560 - ac
c: 0.9883
Epoch 23/100
426/426 [==============================] - 0s 52us/step - loss: 0.0538 - ac
c: 0.9883
Epoch 24/100
426/426 [==============================] - 0s 42us/step - loss: 0.0518 - ac
c: 0.9883
Epoch 25/100
426/426 [==============================] - 0s 50us/step - loss: 0.0500 - ac
c: 0.9906
Epoch 26/100
426/426 [==============================] - 0s 57us/step - loss: 0.0485 - ac
c: 0.9906
Epoch 27/100
426/426 [==============================] - 0s 49us/step - loss: 0.0497 - ac
c: 0.9906
Epoch 28/100
426/426 [==============================] - 0s 47us/step - loss: 0.0476 - ac
c: 0.9906
Epoch 29/100
426/426 [==============================] - 0s 42us/step - loss: 0.0472 - ac
c: 0.9906
Epoch 30/100
426/426 [==============================] - 0s 41us/step - loss: 0.0469 - ac
c: 0.9906
Epoch 31/100
426/426 [==============================] - 0s 41us/step - loss: 0.0481 - ac
c: 0.9906
Epoch 32/100
426/426 [==============================] - 0s 43us/step - loss: 0.0488 - ac
c: 0.9906
Epoch 33/100
426/426 [==============================] - 0s 64us/step - loss: 0.0507 - ac
c: 0.9906
Epoch 34/100
426/426 [==============================] - 0s 47us/step - loss: 0.0483 - ac
c: 0.9906
Epoch 35/100
426/426 [==============================] - 0s 38us/step - loss: 0.0476 - ac
c: 0.9883
Epoch 36/100
426/426 [==============================] - 0s 41us/step - loss: 0.0455 - ac
c: 0.9883
Epoch 37/100
426/426 [==============================] - 0s 40us/step - loss: 0.0462 - ac
```

```
c: 0.9883
Epoch 38/100
426/426 [==============================] - 0s 37us/step - loss: 0.0488 - ac
c: 0.9906
Epoch 39/100
426/426 [==============================] - 0s 59us/step - loss: 0.0448 - ac
c: 0.9906
Epoch 40/100
426/426 [==============================] - 0s 48us/step - loss: 0.0437 - ac
c: 0.9906
Epoch 41/100
426/426 [==============================] - 0s 36us/step - loss: 0.0482 - ac
c: 0.9906
Epoch 42/100
426/426 [==============================] - 0s 41us/step - loss: 0.0455 - ac
c: 0.9906
Epoch 43/100
426/426 [==============================] - 0s 37us/step - loss: 0.0444 - ac
c: 0.9906
Epoch 44/100
426/426 [==============================] - ETA: 0s - loss: 0.0190 - acc: 1.
000 - 0s 45us/step - loss: 0.0487 - acc: 0.9906
Epoch 45/100
426/426 [==============================] - 0s 44us/step - loss: 0.0449 - ac
c: 0.9906
Epoch 46/100
426/426 [==============================] - 0s 68us/step - loss: 0.0422 - ac
c: 0.9906
Epoch 47/100
426/426 [==============================] - ETA: 0s - loss: 0.0164 - acc: 1.
000 - 0s 45us/step - loss: 0.0437 - acc: 0.9906
Epoch 48/100
426/426 [==============================] - 0s 48us/step - loss: 0.0443 - ac
c: 0.9906
Epoch 49/100
426/426 [==============================] - 0s 42us/step - loss: 0.0458 - ac
c: 0.9906
Epoch 50/100
426/426 [==============================] - 0s 47us/step - loss: 0.0448 - ac
c: 0.9906
Epoch 51/100
426/426 [==============================] - 0s 56us/step - loss: 0.0418 - ac
c: 0.9906
Epoch 52/100
426/426 [==============================] - 0s 50us/step - loss: 0.0449 - ac
c: 0.9906
Epoch 53/100
426/426 [==============================] - 0s 51us/step - loss: 0.0449 - ac
c: 0.9906
Epoch 54/100
426/426 [==============================] - 0s 54us/step - loss: 0.0442 - ac
c: 0.9906
Epoch 55/100
426/426 [==============================] - 0s 51us/step - loss: 0.0420 - ac
c: 0.9906
Epoch 56/100
426/426 [==============================] - 0s 43us/step - loss: 0.0425 - ac
```

```
c: 0.9906
Epoch 57/100
426/426 [==============================] - 0s 40us/step - loss: 0.0448 - ac
c: 0.9906
Epoch 58/100
426/426 [==============================] - 0s 36us/step - loss: 0.0452 - ac
c: 0.9906
Epoch 59/100
426/426 [==============================] - 0s 42us/step - loss: 0.0392 - ac
c: 0.9906
Epoch 60/100
426/426 [==============================] - 0s 37us/step - loss: 0.0446 - ac
c: 0.9906
Epoch 61/100
426/426 [==============================] - 0s 79us/step - loss: 0.0426 - ac
c: 0.9906
Epoch 62/100
426/426 [==============================] - 0s 53us/step - loss: 0.0430 - ac
c: 0.9906
Epoch 63/100
426/426 [==============================] - 0s 48us/step - loss: 0.0441 - ac
c: 0.9906
Epoch 64/100
426/426 [==============================] - 0s 42us/step - loss: 0.0449 - ac
c: 0.9906
Epoch 65/100
426/426 [==============================] - 0s 37us/step - loss: 0.0435 - ac
c: 0.9906
Epoch 66/100
426/426 [==============================] - 0s 38us/step - loss: 0.0386 - ac
c: 0.9906
Epoch 67/100
426/426 [==============================] - 0s 47us/step - loss: 0.0415 - ac
c: 0.9906
Epoch 68/100
426/426 [==============================] - 0s 35us/step - loss: 0.0443 - ac
c: 0.9906
Epoch 69/100
426/426 [==============================] - 0s 42us/step - loss: 0.0414 - ac
c: 0.9906
Epoch 70/100
426/426 [==============================] - 0s 38us/step - loss: 0.0399 - ac
c: 0.9906
Epoch 71/100
426/426 [==============================] - 0s 40us/step - loss: 0.0390 - ac
c: 0.9906
Epoch 72/100
426/426 [==============================] - 0s 68us/step - loss: 0.0419 - ac
c: 0.9906
Epoch 73/100
426/426 [==============================] - 0s 71us/step - loss: 0.0390 - ac
c: 0.9906
Epoch 74/100
426/426 [==============================] - 0s 42us/step - loss: 0.0389 - ac
c: 0.9906
Epoch 75/100
426/426 [==============================] - 0s 48us/step - loss: 0.0387 - ac
```

```
c: 0.9906
Epoch 76/100
426/426 [==============================] - 0s 47us/step - loss: 0.0420 - ac
c: 0.9883
Epoch 77/100
426/426 [==============================] - 0s 61us/step - loss: 0.0350 - ac
c: 0.9883
Epoch 78/100
426/426 [==============================] - 0s 48us/step - loss: 0.0398 - ac
c: 0.9906
Epoch 79/100
426/426 [==============================] - 0s 41us/step - loss: 0.0405 - ac
c: 0.9906
Epoch 80/100
426/426 [==============================] - 0s 41us/step - loss: 0.0384 - ac
c: 0.9906
Epoch 81/100
426/426 [==============================] - 0s 38us/step - loss: 0.0389 - ac
c: 0.9906
Epoch 82/100
426/426 [==============================] - 0s 42us/step - loss: 0.0421 - ac
c: 0.9883
Epoch 83/100

426/426 [==============================] - 0s 47us/step - loss: 0.0376 - ac
c: 0.9883
Epoch 84/100
426/426 [==============================] - 0s 73us/step - loss: 0.0385 - ac
c: 0.9906
Epoch 85/100
426/426 [==============================] - 0s 41us/step - loss: 0.0405 - ac
c: 0.9906
Epoch 86/100
426/426 [==============================] - 0s 36us/step - loss: 0.0397 - ac
c: 0.9906
Epoch 87/100
426/426 [==============================] - 0s 42us/step - loss: 0.0389 - ac
c: 0.9906
Epoch 88/100
426/426 [==============================] - 0s 47us/step - loss: 0.0356 - ac
c: 0.9906
Epoch 89/100
426/426 [==============================] - 0s 50us/step - loss: 0.0385 - ac
c: 0.9906
Epoch 90/100
426/426 [==============================] - 0s 45us/step - loss: 0.0346 - ac
c: 0.9906
Epoch 91/100
426/426 [==============================] - 0s 43us/step - loss: 0.0418 - ac
c: 0.9906
Epoch 92/100
426/426 [==============================] - 0s 41us/step - loss: 0.0352 - ac
c: 0.9906
Epoch 93/100
426/426 [==============================] - 0s 45us/step - loss: 0.0357 - ac
c: 0.9906
Epoch 94/100
```

```
426/426 [==============================] - 0s 57us/step - loss: 0.0330 - ac
c: 0.9906
Epoch 95/100
426/426 [==============================] - 0s 47us/step - loss: 0.0371 - ac
c: 0.9906
Epoch 96/100
426/426 [==============================] - 0s 66us/step - loss: 0.0387 - ac
c: 0.9906
Epoch 97/100
426/426 [==============================] - 0s 43us/step - loss: 0.0385 - ac
c: 0.9906
Epoch 98/100
426/426 [==============================] - 0s 48us/step - loss: 0.0348 - ac
c: 0.9906
Epoch 99/100
426/426 [==============================] - 0s 54us/step - loss: 0.0375 - ac
c: 0.9906
Epoch 100/100
426/426 [==============================] - 0s 49us/step - loss: 0.0369 - ac
c: 0.9906
```

Out[97]: <keras.callbacks.History at 0x2599f13b518>

In [98]:
```
# Fitting the ANN to the Training set, epoch=150
#Looks like running 150 epochs gives the best results
classifier.fit(X_train, Y_train, batch_size=100, epochs=150)
```

```
Epoch 1/150
426/426 [==============================] - 0s 72us/step - loss: 0.0360 - ac
c: 0.9906
Epoch 2/150
426/426 [==============================] - 0s 49us/step - loss: 0.0360 - ac
c: 0.9906
Epoch 3/150
426/426 [==============================] - 0s 58us/step - loss: 0.0348 - ac
c: 0.9906
Epoch 4/150
426/426 [==============================] - 0s 52us/step - loss: 0.0368 - ac
c: 0.9906
Epoch 5/150
426/426 [==============================] - 0s 44us/step - loss: 0.0371 - ac
c: 0.9906
Epoch 6/150
426/426 [==============================] - 0s 41us/step - loss: 0.0371 - ac
c: 0.9906
Epoch 7/150
426/426 [==============================] - 0s 44us/step - loss: 0.0345 - ac
c: 0.9906
Epoch 8/150
426/426 [==============================] - 0s 50us/step - loss: 0.0342 - ac
c: 0.9906
Epoch 9/150
426/426 [==============================] - 0s 80us/step - loss: 0.0360 - ac
c: 0.9906
Epoch 10/150
426/426 [==============================] - 0s 48us/step - loss: 0.0340 - ac
c: 0.9906
Epoch 11/150
426/426 [==============================] - 0s 48us/step - loss: 0.0340 - ac
c: 0.9906
Epoch 12/150
426/426 [==============================] - 0s 41us/step - loss: 0.0359 - ac
c: 0.9906
Epoch 13/150
426/426 [==============================] - 0s 43us/step - loss: 0.0361 - ac
c: 0.9906
Epoch 14/150
426/426 [==============================] - 0s 91us/step - loss: 0.0357 - ac
c: 0.9906
Epoch 15/150
426/426 [==============================] - 0s 42us/step - loss: 0.0331 - ac
c: 0.9906
Epoch 16/150
426/426 [==============================] - 0s 45us/step - loss: 0.0300 - ac
c: 0.9906
Epoch 17/150
426/426 [==============================] - 0s 47us/step - loss: 0.0348 - ac
c: 0.9906
Epoch 18/150
```

```
426/426 [==============================] - 0s 49us/step - loss: 0.0353 - ac
c: 0.9906
Epoch 19/150
426/426 [==============================] - 0s 69us/step - loss: 0.0363 - ac
c: 0.9906
Epoch 20/150
426/426 [==============================] - 0s 44us/step - loss: 0.0330 - ac
c: 0.9906
Epoch 21/150
426/426 [==============================] - 0s 44us/step - loss: 0.0350 - ac
c: 0.9906
Epoch 22/150
426/426 [==============================] - 0s 36us/step - loss: 0.0317 - ac
c: 0.9906
Epoch 23/150
426/426 [==============================] - 0s 41us/step - loss: 0.0322 - ac
c: 0.9906
Epoch 24/150
426/426 [==============================] - 0s 40us/step - loss: 0.0351 - ac
c: 0.9883
Epoch 25/150
426/426 [==============================] - 0s 44us/step - loss: 0.0359 - ac
c: 0.9906
Epoch 26/150
426/426 [==============================] - 0s 55us/step - loss: 0.0313 - ac
c: 0.9906
Epoch 27/150
426/426 [==============================] - 0s 45us/step - loss: 0.0339 - ac
c: 0.9906
Epoch 28/150
426/426 [==============================] - 0s 56us/step - loss: 0.0342 - ac
c: 0.9906
Epoch 29/150
426/426 [==============================] - 0s 50us/step - loss: 0.0311 - ac
c: 0.9906
Epoch 30/150
426/426 [==============================] - 0s 42us/step - loss: 0.0323 - ac
c: 0.9906
Epoch 31/150
426/426 [==============================] - 0s 52us/step - loss: 0.0307 - ac
c: 0.9906
Epoch 32/150
426/426 [==============================] - 0s 44us/step - loss: 0.0345 - ac
c: 0.9883
Epoch 33/150
426/426 [==============================] - 0s 62us/step - loss: 0.0324 - ac
c: 0.9906
Epoch 34/150
426/426 [==============================] - 0s 49us/step - loss: 0.0315 - ac
c: 0.9906
Epoch 35/150
426/426 [==============================] - 0s 47us/step - loss: 0.0306 - ac
c: 0.9906
Epoch 36/150
426/426 [==============================] - 0s 57us/step - loss: 0.0323 - ac
c: 0.9883
Epoch 37/150
```

```
426/426 [==============================] - 0s 54us/step - loss: 0.0320 - ac
c: 0.9906
Epoch 38/150
426/426 [==============================] - 0s 55us/step - loss: 0.0318 - ac
c: 0.9906
Epoch 39/150
426/426 [==============================] - 0s 49us/step - loss: 0.0332 - ac
c: 0.9906
Epoch 40/150
426/426 [==============================] - 0s 50us/step - loss: 0.0297 - ac
c: 0.9906
Epoch 41/150
426/426 [==============================] - 0s 50us/step - loss: 0.0304 - ac
c: 0.9906
Epoch 42/150
426/426 [==============================] - 0s 52us/step - loss: 0.0284 - ac
c: 0.9906
Epoch 43/150
426/426 [==============================] - 0s 54us/step - loss: 0.0311 - ac
c: 0.9906
Epoch 44/150
426/426 [==============================] - 0s 43us/step - loss: 0.0323 - ac
c: 0.9906
Epoch 45/150
426/426 [==============================] - 0s 45us/step - loss: 0.0285 - ac
c: 0.9906
Epoch 46/150
426/426 [==============================] - 0s 52us/step - loss: 0.0287 - ac
c: 0.9906
Epoch 47/150
426/426 [==============================] - 0s 49us/step - loss: 0.0300 - ac
c: 0.9906
Epoch 48/150
426/426 [==============================] - 0s 36us/step - loss: 0.0287 - ac
c: 0.9906
Epoch 49/150
426/426 [==============================] - 0s 48us/step - loss: 0.0311 - ac
c: 0.9906
Epoch 50/150
426/426 [==============================] - 0s 86us/step - loss: 0.0301 - ac
c: 0.9906
Epoch 51/150
426/426 [==============================] - 0s 49us/step - loss: 0.0265 - ac
c: 0.9906
Epoch 52/150
426/426 [==============================] - 0s 51us/step - loss: 0.0284 - ac
c: 0.9906
Epoch 53/150
426/426 [==============================] - 0s 50us/step - loss: 0.0273 - ac
c: 0.9906
Epoch 54/150
426/426 [==============================] - 0s 47us/step - loss: 0.0296 - ac
c: 0.9906
Epoch 55/150
426/426 [==============================] - 0s 57us/step - loss: 0.0271 - ac
c: 0.9906
Epoch 56/150
```

```
426/426 [==============================] - 0s 47us/step - loss: 0.0285 - ac
c: 0.9906
Epoch 57/150
426/426 [==============================] - 0s 41us/step - loss: 0.0262 - ac
c: 0.9906
Epoch 58/150
426/426 [==============================] - 0s 49us/step - loss: 0.0294 - ac
c: 0.9906
Epoch 59/150
426/426 [==============================] - 0s 41us/step - loss: 0.0260 - ac
c: 0.9906
Epoch 60/150
426/426 [==============================] - 0s 48us/step - loss: 0.0286 - ac
c: 0.9906
Epoch 61/150
426/426 [==============================] - 0s 48us/step - loss: 0.0308 - ac
c: 0.9906
Epoch 62/150
426/426 [==============================] - ETA: 0s - loss: 0.0086 - acc: 1.
000 - 0s 55us/step - loss: 0.0265 - acc: 0.9906
Epoch 63/150
426/426 [==============================] - 0s 41us/step - loss: 0.0259 - ac
c: 0.9906
Epoch 64/150
426/426 [==============================] - 0s 43us/step - loss: 0.0278 - ac
c: 0.9906
Epoch 65/150
426/426 [==============================] - 0s 48us/step - loss: 0.0265 - ac
c: 0.9906
Epoch 66/150
426/426 [==============================] - 0s 44us/step - loss: 0.0255 - ac
c: 0.9906
Epoch 67/150
426/426 [==============================] - 0s 41us/step - loss: 0.0245 - ac
c: 0.9906
Epoch 68/150
426/426 [==============================] - 0s 43us/step - loss: 0.0267 - ac
c: 0.9906
Epoch 69/150
426/426 [==============================] - 0s 48us/step - loss: 0.0248 - ac
c: 0.9906
Epoch 70/150
426/426 [==============================] - 0s 42us/step - loss: 0.0290 - ac
c: 0.9906
Epoch 71/150
426/426 [==============================] - 0s 41us/step - loss: 0.0269 - ac
c: 0.9906
Epoch 72/150
426/426 [==============================] - 0s 43us/step - loss: 0.0255 - ac
c: 0.9906
Epoch 73/150
426/426 [==============================] - 0s 54us/step - loss: 0.0262 - ac
c: 0.9906
Epoch 74/150
426/426 [==============================] - 0s 45us/step - loss: 0.0252 - ac
c: 0.9906
Epoch 75/150
```

```
426/426 [==============================] - 0s 44us/step - loss: 0.0261 - ac
c: 0.9906
Epoch 76/150
426/426 [==============================] - 0s 45us/step - loss: 0.0241 - ac
c: 0.9906
Epoch 77/150
426/426 [==============================] - 0s 61us/step - loss: 0.0243 - ac
c: 0.9906
Epoch 78/150
426/426 [==============================] - 0s 48us/step - loss: 0.0257 - ac
c: 0.9906
Epoch 79/150
426/426 [==============================] - 0s 40us/step - loss: 0.0267 - ac
c: 0.9906
Epoch 80/150
426/426 [==============================] - 0s 51us/step - loss: 0.0249 - ac
c: 0.9906
Epoch 81/150
426/426 [==============================] - 0s 49us/step - loss: 0.0241 - ac
c: 0.9906
Epoch 82/150
426/426 [==============================] - 0s 38us/step - loss: 0.0228 - ac
c: 0.9906
Epoch 83/150
426/426 [==============================] - 0s 42us/step - loss: 0.0226 - ac
c: 0.9906
Epoch 84/150

426/426 [==============================] - 0s 47us/step - loss: 0.0256 - ac
c: 0.9906
Epoch 85/150
426/426 [==============================] - 0s 51us/step - loss: 0.0251 - ac
c: 0.9906
Epoch 86/150
426/426 [==============================] - 0s 48us/step - loss: 0.0255 - ac
c: 0.9883
Epoch 87/150
426/426 [==============================] - 0s 38us/step - loss: 0.0225 - ac
c: 0.9906
Epoch 88/150
426/426 [==============================] - 0s 40us/step - loss: 0.0234 - ac
c: 0.9906
Epoch 89/150
426/426 [==============================] - 0s 64us/step - loss: 0.0226 - ac
c: 0.9906
Epoch 90/150
426/426 [==============================] - 0s 49us/step - loss: 0.0226 - ac
c: 0.9906
Epoch 91/150
426/426 [==============================] - 0s 38us/step - loss: 0.0218 - ac
c: 0.9906
Epoch 92/150
426/426 [==============================] - 0s 42us/step - loss: 0.0255 - ac
c: 0.9906
Epoch 93/150
426/426 [==============================] - 0s 44us/step - loss: 0.0223 - ac
c: 0.9906
```

```
Epoch 94/150
426/426 [==============================] - 0s 54us/step - loss: 0.0235 - ac
c: 0.9906
Epoch 95/150
426/426 [==============================] - 0s 63us/step - loss: 0.0225 - ac
c: 0.9906
Epoch 96/150
426/426 [==============================] - 0s 49us/step - loss: 0.0229 - ac
c: 0.9930
Epoch 97/150
426/426 [==============================] - 0s 54us/step - loss: 0.0208 - ac
c: 0.9930
Epoch 98/150
426/426 [==============================] - 0s 50us/step - loss: 0.0220 - ac
c: 0.9930
Epoch 99/150
426/426 [==============================] - 0s 50us/step - loss: 0.0227 - ac
c: 0.9906
Epoch 100/150
426/426 [==============================] - 0s 57us/step - loss: 0.0211 - ac
c: 0.9906
Epoch 101/150
426/426 [==============================] - 0s 45us/step - loss: 0.0244 - ac
c: 0.9883
Epoch 102/150
426/426 [==============================] - 0s 62us/step - loss: 0.0235 - ac
c: 0.9906
Epoch 103/150
426/426 [==============================] - 0s 43us/step - loss: 0.0189 - ac
c: 0.9930
Epoch 104/150
426/426 [==============================] - 0s 54us/step - loss: 0.0193 - ac
c: 0.9930
Epoch 105/150
426/426 [==============================] - 0s 43us/step - loss: 0.0235 - ac
c: 0.9906
Epoch 106/150
426/426 [==============================] - 0s 58us/step - loss: 0.0187 - ac
c: 0.9930
Epoch 107/150
426/426 [==============================] - 0s 48us/step - loss: 0.0194 - ac
c: 0.9930
Epoch 108/150
426/426 [==============================] - 0s 49us/step - loss: 0.0189 - ac
c: 0.9930
Epoch 109/150
426/426 [==============================] - 0s 55us/step - loss: 0.0200 - ac
c: 0.9930
Epoch 110/150
426/426 [==============================] - 0s 55us/step - loss: 0.0197 - ac
c: 0.9930
Epoch 111/150
426/426 [==============================] - 0s 48us/step - loss: 0.0193 - ac
c: 0.9930
Epoch 112/150
426/426 [==============================] - 0s 44us/step - loss: 0.0196 - ac
c: 0.9930
```

```
Epoch 113/150
426/426 [==============================] - 0s 49us/step - loss: 0.0183 - ac
c: 0.9930
Epoch 114/150
426/426 [==============================] - 0s 40us/step - loss: 0.0175 - ac
c: 0.9930
Epoch 115/150
426/426 [==============================] - 0s 42us/step - loss: 0.0183 - ac
c: 0.9930
Epoch 116/150
426/426 [==============================] - 0s 56us/step - loss: 0.0160 - ac
c: 0.9953
Epoch 117/150
426/426 [==============================] - 0s 52us/step - loss: 0.0181 - ac
c: 0.9930
Epoch 118/150
426/426 [==============================] - 0s 49us/step - loss: 0.0172 - ac
c: 0.9930
Epoch 119/150
426/426 [==============================] - 0s 45us/step - loss: 0.0186 - ac
c: 0.9930
Epoch 120/150
426/426 [==============================] - 0s 49us/step - loss: 0.0166 - ac
c: 0.9930
Epoch 121/150
426/426 [==============================] - 0s 45us/step - loss: 0.0219 - ac
c: 0.9930
Epoch 122/150
426/426 [==============================] - 0s 42us/step - loss: 0.0179 - ac
c: 0.9953
Epoch 123/150
426/426 [==============================] - 0s 41us/step - loss: 0.0170 - ac
c: 0.9953
Epoch 124/150
426/426 [==============================] - 0s 59us/step - loss: 0.0187 - ac
c: 0.9953
Epoch 125/150
426/426 [==============================] - 0s 48us/step - loss: 0.0169 - ac
c: 0.9953
Epoch 126/150
426/426 [==============================] - 0s 49us/step - loss: 0.0166 - ac
c: 0.9953
Epoch 127/150
426/426 [==============================] - 0s 48us/step - loss: 0.0168 - ac
c: 0.9953
Epoch 128/150
426/426 [==============================] - 0s 58us/step - loss: 0.0158 - ac
c: 0.9953
Epoch 129/150
426/426 [==============================] - 0s 44us/step - loss: 0.0195 - ac
c: 0.9906
Epoch 130/150
426/426 [==============================] - 0s 51us/step - loss: 0.0176 - ac
c: 0.9953
Epoch 131/150
426/426 [==============================] - 0s 48us/step - loss: 0.0155 - ac
c: 0.9953
```

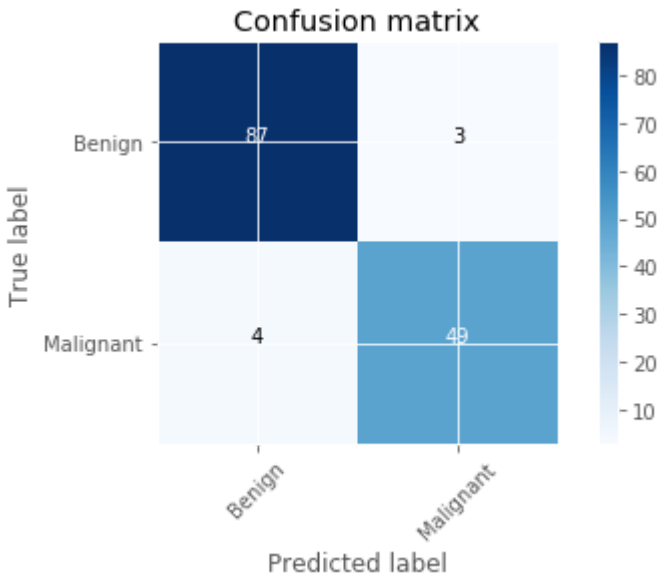
```
Epoch 132/150
426/426 [==============================] - 0s 43us/step - loss: 0.0155 - ac
c: 0.9930
Epoch 133/150
426/426 [==============================] - 0s 54us/step - loss: 0.0154 - ac
c: 0.9953
Epoch 134/150
426/426 [==============================] - 0s 42us/step - loss: 0.0144 - ac
c: 0.9953
Epoch 135/150
426/426 [==============================] - 0s 65us/step - loss: 0.0155 - ac
c: 0.9930
Epoch 136/150
426/426 [==============================] - 0s 51us/step - loss: 0.0164 - ac
c: 0.9953
Epoch 137/150
426/426 [==============================] - 0s 41us/step - loss: 0.0166 - ac
c: 0.9953
Epoch 138/150
426/426 [==============================] - 0s 50us/step - loss: 0.0137 - ac
c: 0.9953
Epoch 139/150
426/426 [==============================] - 0s 48us/step - loss: 0.0144 - ac
c: 0.9953
Epoch 140/150
426/426 [==============================] - 0s 50us/step - loss: 0.0174 - ac
c: 0.9953
Epoch 141/150
426/426 [==============================] - 0s 51us/step - loss: 0.0145 - ac
c: 0.9953
Epoch 142/150
426/426 [==============================] - 0s 44us/step - loss: 0.0163 - ac
c: 0.9906
Epoch 143/150
426/426 [==============================] - 0s 54us/step - loss: 0.0145 - ac
c: 0.9953
Epoch 144/150
426/426 [==============================] - 0s 45us/step - loss: 0.0137 - ac
c: 0.9930
Epoch 145/150
426/426 [==============================] - 0s 38us/step - loss: 0.0135 - ac
c: 0.9953
Epoch 146/150
426/426 [==============================] - 0s 48us/step - loss: 0.0146 - ac
c: 0.9930
Epoch 147/150
426/426 [==============================] - 0s 55us/step - loss: 0.0132 - ac
c: 0.9953
Epoch 148/150
426/426 [==============================] - 0s 45us/step - loss: 0.0124 - ac
c: 0.9977
Epoch 149/150
426/426 [==============================] - 0s 40us/step - loss: 0.0135 - ac
c: 0.9977
Epoch 150/150
426/426 [==============================] - 0s 55us/step - loss: 0.0129 - ac
c: 0.9977
```

Out[98]: <keras.callbacks.History at 0x259a2042320>

In [99]:
```
# Predicting the Test set results
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)
```

In [100]:
```
# Making the Confusion Matrix
cm = confusion_matrix(Y_test, y_pred)

TN = cm[0][0]
TP = cm[1][1]
FN = cm[1][0]
FP = cm[0][1]

print(cm)
print('Our accuracy is {}%'.format(((cm[0][0] + cm[1][1])/(cm[0][0] + cm[1][1
plot_confusion_matrix(cm, classes=class_names, title='Confusion matrix')
plt.show();
```

```
[[87  3]
 [ 4 49]]
Our accuracy is 95.1048951048951%
```



Using ANN model showed an accuracy of 95%. I believe using 150 epochs will help the model perform to the highest level of accuracy.

Conclusion

In closing, the machine learning models with the highest level of accuracy in predicting malignancies was logistic regression at 97%. Random Forest was a close second, coming in at 95%. The deep learning model using Keras showed 95%, as well as SVM.

I was hoping that the neural network would show the highest level of accuracy (since it mimics the human brain), but its performance was on par in comparison to some of the machine learning models. Overall that still not bad though. While using machine learning or deep learning models can help predict which patients have breast cancer, this technique is still fallible. Even using a medical doctor to determine breast cancer is not 100% accurate. However, this technique does show great promise.

I believe using either machine learning or deep learning to predict breast cancer, along with the skillful trained eye of a physician, will be the best course of action for assessing breast cancer malignancies in patients. This methodology can help minimize false negatives (which would subject the patient to unecessary testing) and maximize true positives in breast cancer.