# Chapter 1: Breast Cancer Detection

In [6]:
```
# Let explore the dataset and do a few visualizations
print(df.loc[10])

# Print the shape of the dataset
print(df.shape)
```

```
clump_thickness           1
uniform_cell_size         1
uniform_cell_shape        1
marginal_adhesion         1
single_epithelial_size    1
bare_nuclei               1
bland_chromatin           3
normal_nucleoli           1
mitoses                   1
class                     2
Name: 10, dtype: object
(699, 10)
```

```
In [10]: # Define models to train
models = []
models.append(('KNN', KNeighborsClassifier(n_neighbors = 5)))
models.append(('SVM', SVC()))

# evaluate each model in turn
results = []
names = []

for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state = seed)
    cv_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
KNN: 0.962468 (0.018609)
SVM: 0.958929 (0.029934)
```

Anaconda Prompt

```
(base) C:\Users\test>D:

(base) D:\>cd D:\Tutorial

(base) D:\Tutorial>conda install numpy
```

Anaconda Prompt - jupyter notebook

```
(base) C:\Users\test>D:

(base) D:\>cd D:\Tutorial

(base) D:\Tutorial>jupyter notebook
[I 18:37:09.670 NotebookApp] The port 8888 is already in use, trying another port.
[I 18:37:10.100 NotebookApp] Loading IPython parallel extension
[I 18:37:10.330 NotebookApp] JupyterLab extension loaded from C:\Users\test\Anaconda3\lib\site-packages\jupyterlab
[I 18:37:10.330 NotebookApp] JupyterLab application directory is C:\Users\test\Anaconda3\share\jupyter\lab
[I 18:37:10.360 NotebookApp] Serving notebooks from local directory: D:\Tutorial
[I 18:37:10.360 NotebookApp] The Jupyter Notebook is running at:
[I 18:37:10.360 NotebookApp] http://localhost:8889/?token=b9b0fa73e9ce7368ed96458fa0e133994e987c4befb387dc
[I 18:37:10.360 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 18:37:10.380 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8889/?token=b9b0fa73e9ce7368ed96458fa0e133994e987c4befb387dc
[I 18:37:10.660 NotebookApp] Accepting one-time-token-authenticated connection from ::1
```

In [1]:

```python
import sys
import scipy
import numpy
import matplotlib
import pandas
import sklearn

print('Python: {}'.format(sys.version))
print('scipy: {}'.format(scipy.__version__))
print('numpy: {}'.format(numpy.__version__))
print('matplotlib: {}'.format(matplotlib.__version__))
print('pandas: {}'.format(pandas.__version__))
print('sklearn: {}'.format(sklearn.__version__))
```

```
Python: 3.6.6 |Anaconda, Inc.| (default, Jun 28 2018, 11:27:44) [MSC v.1900 64 bit (AMD64)]
scipy: 1.1.0
numpy: 1.15.0
matplotlib: 2.2.2
pandas: 0.23.3
sklearn: 0.19.1
```

In [2]:
```python
import numpy as np
from sklearn import preprocessing, cross_validation
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn import model_selection
from sklearn.metrics import classification_report, accuracy_score
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
import pandas as pd
```

```
C:\Users\test\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41: Depr
ecationWarning: This module was deprecated in version 0.18 in favor of the mod
el_selection module into which all the refactored classes and functions are mo
ved. Also note that the interface of the new CV iterators are different from t
hat of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [3]:
```python
# Load Dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data"
names = ['id', 'clump_thickness', 'uniform_cell_size', 'uniform_cell_shape',
       'marginal_adhesion', 'single_epithelial_size', 'bare_nuclei',
       'bland_chromatin', 'normal_nucleoli', 'mitoses', 'class']
df = pd.read_csv(url, names=names)
```

In [4]:
```python
# Preprocess the data
df.replace('?',-99999, inplace=True)
print(df.axes)

df.drop(['id'], 1, inplace=True)

# Print the shape of the dataset
print(df.shape)
```

```
[RangeIndex(start=0, stop=699, step=1), Index(['id', 'clump_thickness', 'uniform_cell_size', 'unifor
m_cell_shape',
       'marginal_adhesion', 'single_epithelial_size', 'bare_nuclei',
       'bland_chromatin', 'normal_nucleoli', 'mitoses', 'class'],
      dtype='object')]
(699, 10)
```

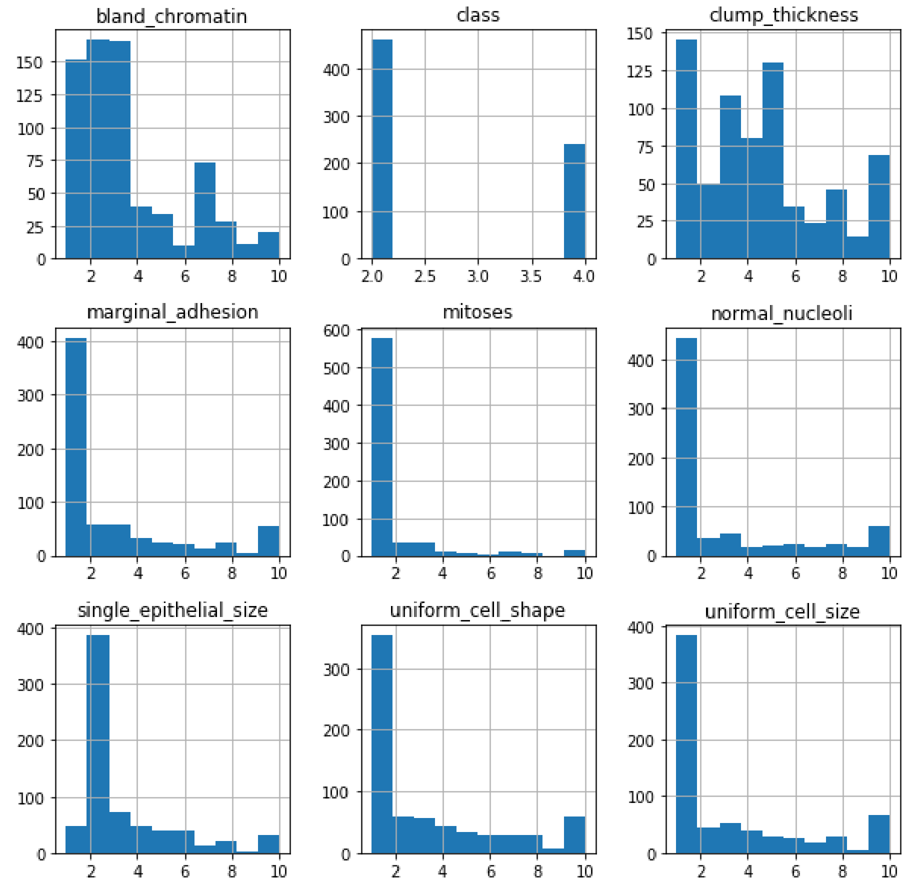
In [5]:
```
# Do dataset visualizations
print(df.loc[6])
```

```
clump_thickness             1
uniform_cell_size           1
uniform_cell_shape          1
marginal_adhesion           1
single_epithelial_size      2
bare_nuclei                10
bland_chromatin             3
normal_nucleoli             1
mitoses                     1
class                       2
Name: 6, dtype: object
```

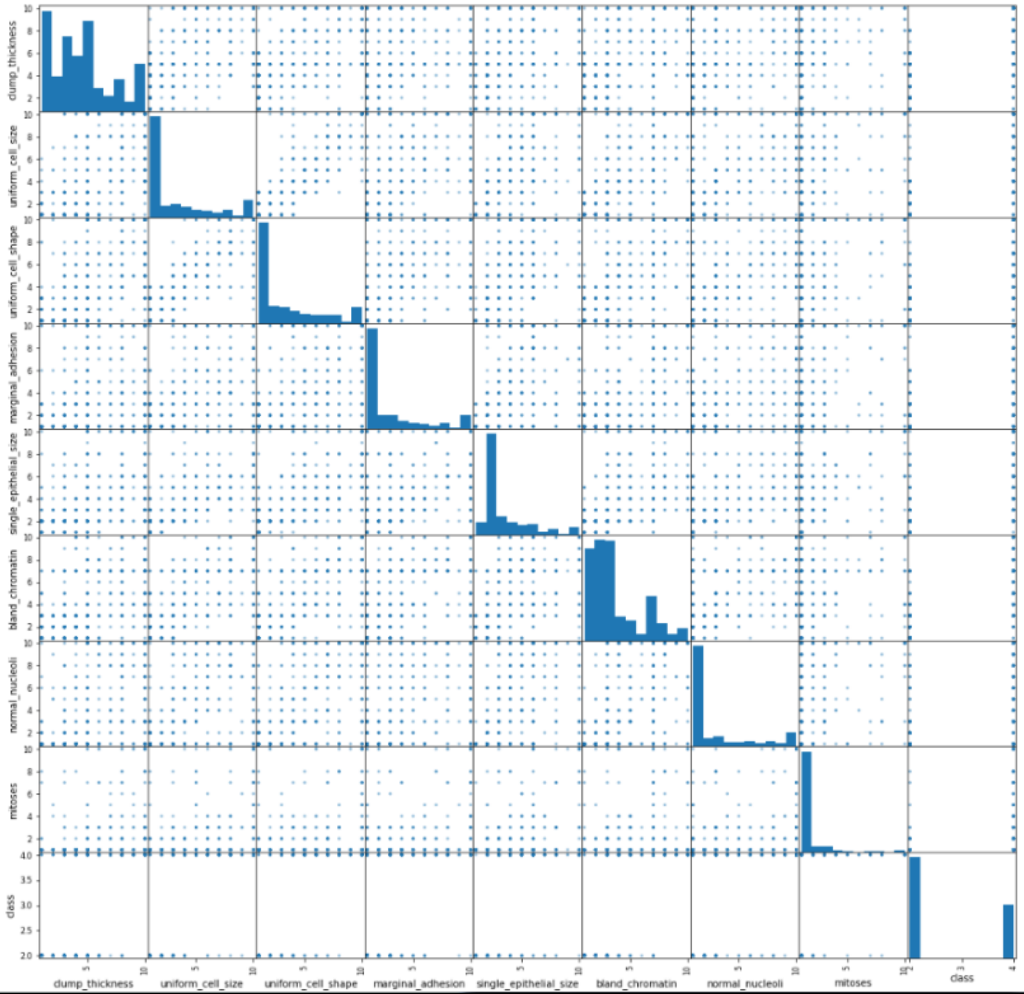
```
In [6]: # Do dataset visualizations
        print(df.loc[6])
        print(df.describe())
```

```
clump_thickness          1
uniform_cell_size        1
uniform_cell_shape       1
marginal_adhesion        1
single_epithelial_size   2
bare_nuclei             10
bland_chromatin          3
normal_nucleoli          1
mitoses                  1
class                    2
Name: 6, dtype: object
       clump_thickness  uniform_cell_size  uniform_cell_shape  \
count       699.000000         699.000000          699.000000
mean          4.417740           3.134478            3.207439
std           2.815741           3.051459            2.971913
min           1.000000           1.000000            1.000000
25%           2.000000           1.000000            1.000000
50%           4.000000           1.000000            1.000000
75%           6.000000           5.000000            5.000000
max          10.000000          10.000000           10.000000

       marginal_adhesion  single_epithelial_size  bland_chromatin  \
count         699.000000              699.000000       699.000000
mean            2.806867                3.216023         3.437768
std             2.855379                2.214300         2.438364
min             1.000000                1.000000         1.000000
25%             1.000000                2.000000         2.000000
50%             1.000000                2.000000         3.000000
75%             4.000000                4.000000         5.000000
max            10.000000               10.000000        10.000000

       normal_nucleoli     mitoses       class
count       699.000000  699.000000  699.000000
mean          2.866953    1.589413    2.689557
std           3.053634    1.715078    0.951273
min           1.000000    1.000000    2.000000
25%           1.000000    1.000000    2.000000
50%           1.000000    1.000000    2.000000
75%           4.000000    1.000000    4.000000
max          10.000000   10.000000    4.000000
```

In [7]:
```
# Plot histograms for each variable
df.hist(figsize = (10, 10))
plt.show()
```



```
In [8]: # Create scatter plot matrix
        scatter_matrix(df, figsize = (18,18))
        plt.show()
```




```
In [9]: # Create X and Y datasets for training
        X = np.array(df.drop(['class'], 1))
        y = np.array(df['class'])

        X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y, test_size=0.2)
```

In [10]:
```python
# Testing Options
seed = 8
scoring = 'accuracy'
```

In [11]:
```python
# Define models to train
models = []
models.append(('KNN', KNeighborsClassifier(n_neighbors = 5)))
models.append(('SVM', SVC()))

# evaluate each model in turn
results = []
names = []

for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state = seed)
    cv_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
KNN: 0.966039 (0.018616)
SVM: 0.955292 (0.021477)
```

In [11]:
```python
# Define models to train
models = []
models.append(('KNN', KNeighborsClassifier(n_neighbors = 5)))
models.append(('SVM', SVC()))

# evaluate each model in turn
results = []
names = []

for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state = seed)
    cv_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
KNN: 0.966039 (0.018616)
SVM: 0.955292 (0.021477)
```

In [11]:

```python
# Make predictions on validation dataset

for name, model in models:
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    print(name)
    print(accuracy_score(y_test, predictions))
    print(classification_report(y_test, predictions))
```

```
KNN
0.9785714285714285
             precision    recall  f1-score   support

          2       0.98      0.99      0.98        95
          4       0.98      0.96      0.97        45

avg / total       0.98      0.98      0.98       140

SVM
0.9571428571428572
             precision    recall  f1-score   support

          2       1.00      0.94      0.97        95
          4       0.88      1.00      0.94        45

avg / total       0.96      0.96      0.96       140
```

```
In [11]: # Make predictions on validation dataset

         for name, model in models:
             model.fit(X_train, y_train)
             predictions = model.predict(X_test)
             print(name)
             print(accuracy_score(y_test, predictions))
             print(classification_report(y_test, predictions))
```

```
KNN
0.9785714285714285
             precision    recall  f1-score   support

          2       0.98      0.99      0.98        95
          4       0.98      0.96      0.97        45

avg / total       0.98      0.98      0.98       140

SVM
0.9571428571428572
             precision    recall  f1-score   support

          2       1.00      0.94      0.97        95
          4       0.88      1.00      0.94        45

avg / total       0.96      0.96      0.96       140
```

```
In [13]: clf = SVC()

         clf.fit(X_train, y_train)
         accuracy = clf.score(X_test, y_test)
         print(accuracy)

         example_measures = np.array([[4,2,1,1,1,2,3,2,1]])
         example_measures = example_measures.reshape(len(example_measures), -1)
         prediction = clf.predict(example_measures)
         print(prediction)
```

```
0.95
[2]
```

```
In [12]: clf = SVC()

         clf.fit(X_train, y_train)
         accuracy = clf.score(X_test, y_test)
         print(accuracy)

         example_measures = np.array([[4,2,1,1,1,2,3,2,1]])
         example_measures = example_measures.reshape(len(example_measures), -1)
         prediction = clf.predict(example_measures)
         print(prediction)
```

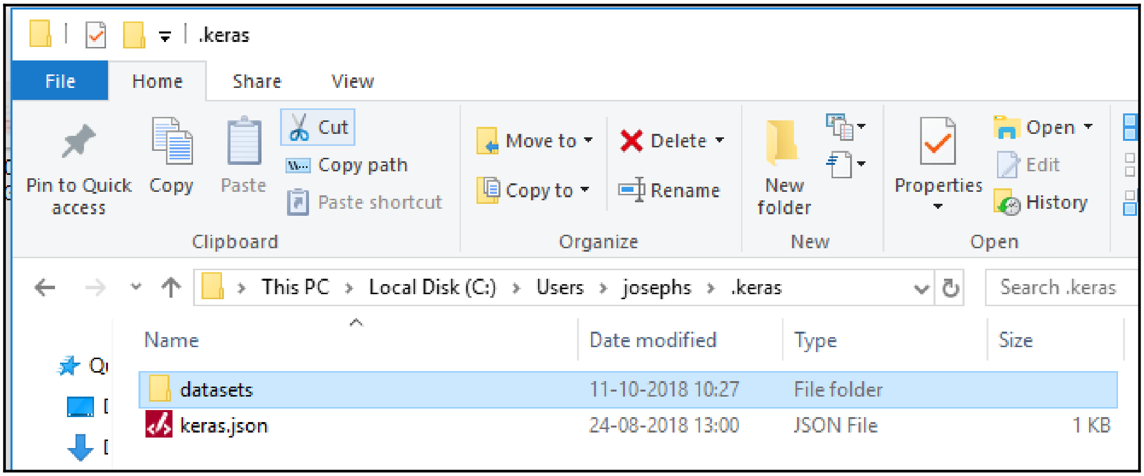
...

```
In [13]: clf = SVC()

         clf.fit(X_train, y_train)
         accuracy = clf.score(X_test, y_test)
         print(accuracy)

         example_measures = np.array([[4,2,1,1,1,2,3,2,1]])
         example_measures = example_measures.reshape(len(example_measures), -1)
         prediction = clf.predict(example_measures)
         print(prediction)

    0.95
    [2]
```

# Chapter 2: Diabetes Onset Detection

```
In [1]: import sys
        import pandas
        import numpy
        import sklearn
        import keras

        print('Python: {}'.format(sys.version))
        print('Pandas: {}'.format(pandas.__version__))
        print('Numpy: {}'.format(numpy.__version__))
        print('Sklearn: {}'.format(sklearn.__version__))
        print('Keras: {}'.format(keras.__version__))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the
p.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
Python: 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32:41) [MSC v.1900 64 bit (AMD64)]
Pandas: 0.23.0
Numpy: 1.14.3
Sklearn: 0.19.1
Keras: 2.2.2
```



keras.json - Notepad

File Edit Format View Help

```
{
    "floatx": "float32",
    "epsilon": 1e-07,
    "backend": "tensorflow",
    "image_data_format": "channels_last"
}
```

In [2]:
```
import pandas as pd
import numpy as np

# import the uci pima indians diabetes dataset
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['n_pregnant', 'glucose_concentration', 'blood_pressure (mm Hg)', 'skin_thickness (mm)', 'serum_insulin (mu U/ml)',
        'BMI', 'pedigree_function', 'age', 'class']
df = pd.read_csv(url, names = names)
```

In [3]:
```
# Describe the dataset
df.describe()
```

Out[3]:

| | n_pregnant | glucose_concentration | blood_pressure (mm Hg) | skin_thickness (mm) | serum_insulin (mu U/ml) | BMI | pedigree_function | age | class |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 |
| **mean** | 3.845052 | 120.894531 | 69.105469 | 20.536458 | 79.799479 | 31.992578 | 0.471876 | 33.240885 | 0.348958 |
| **std** | 3.369578 | 31.972618 | 19.355807 | 15.952218 | 115.244002 | 7.884160 | 0.331329 | 11.760232 | 0.476951 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.078000 | 21.000000 | 0.000000 |
| **25%** | 1.000000 | 99.000000 | 62.000000 | 0.000000 | 0.000000 | 27.300000 | 0.243750 | 24.000000 | 0.000000 |
| **50%** | 3.000000 | 117.000000 | 72.000000 | 23.000000 | 30.500000 | 32.000000 | 0.372500 | 29.000000 | 0.000000 |
| **75%** | 6.000000 | 140.250000 | 80.000000 | 32.000000 | 127.250000 | 36.600000 | 0.626250 | 41.000000 | 1.000000 |
| **max** | 17.000000 | 199.000000 | 122.000000 | 99.000000 | 846.000000 | 67.100000 | 2.420000 | 81.000000 | 1.000000 |

In [4]:
```
df[df['glucose_concentration'] == 0]
```

Out[4]:

| | n_pregnant | glucose_concentration | blood_pressure (mm Hg) | skin_thickness (mm) | serum_insulin (mu U/ml) | BMI | pedigree_function | age | class |
|---|---|---|---|---|---|---|---|---|---|
| **75** | 1 | 0 | 48 | 20 | 0 | 24.7 | 0.140 | 22 | 0 |
| **182** | 1 | 0 | 74 | 20 | 23 | 27.7 | 0.299 | 21 | 0 |
| **342** | 1 | 0 | 68 | 35 | 0 | 32.0 | 0.389 | 22 | 0 |
| **349** | 5 | 0 | 80 | 32 | 0 | 41.0 | 0.346 | 37 | 1 |
| **502** | 6 | 0 | 68 | 41 | 0 | 39.0 | 0.727 | 41 | 1 |

In [5]:
```python
# Preprocess the data, mark zero values as NaN and drop
columns = ['glucose_concentration', 'blood_pressure (mm Hg)', 'skin_thickness (mm)', 'serum_insulin (mu U/ml)', 'BMI']

for col in columns:
    df[col].replace(0, np.NaN, inplace=True)

df.describe()
```

Out[5]:

| | n_pregnant | glucose_concentration | blood_pressure (mm Hg) | skin_thickness (mm) | serum_insulin (mu U/ml) | BMI | pedigree_function | age | class |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 768.000000 | 763.000000 | 733.000000 | 541.000000 | 394.000000 | 757.000000 | 768.000000 | 768.000000 | 768.000000 |
| **mean** | 3.845052 | 121.686763 | 72.405184 | 29.153420 | 155.548223 | 32.457464 | 0.471876 | 33.240885 | 0.348958 |
| **std** | 3.369578 | 30.535641 | 12.382158 | 10.476982 | 118.775855 | 6.924988 | 0.331329 | 11.760232 | 0.476951 |
| **min** | 0.000000 | 44.000000 | 24.000000 | 7.000000 | 14.000000 | 18.200000 | 0.078000 | 21.000000 | 0.000000 |
| **25%** | 1.000000 | 99.000000 | 64.000000 | 22.000000 | 76.250000 | 27.500000 | 0.243750 | 24.000000 | 0.000000 |
| **50%** | 3.000000 | 117.000000 | 72.000000 | 29.000000 | 125.000000 | 32.300000 | 0.372500 | 29.000000 | 0.000000 |
| **75%** | 6.000000 | 141.000000 | 80.000000 | 36.000000 | 190.000000 | 36.600000 | 0.626250 | 41.000000 | 1.000000 |
| **max** | 17.000000 | 199.000000 | 122.000000 | 99.000000 | 846.000000 | 67.100000 | 2.420000 | 81.000000 | 1.000000 |

In [6]:
```python
# Drop rows with missing values
df.dropna(inplace=True)

# summarize the number of rows and columns in df
df.describe()
```

Out[6]:

| | n_pregnant | glucose_concentration | blood_pressure (mm Hg) | skin_thickness (mm) | serum_insulin (mu U/ml) | BMI | pedigree_function | age | class |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 392.000000 | 392.000000 | 392.000000 | 392.000000 | 392.000000 | 392.000000 | 392.000000 | 392.000000 | 392.000000 |
| **mean** | 3.301020 | 122.627551 | 70.663265 | 29.145408 | 156.056122 | 33.086224 | 0.523046 | 30.864796 | 0.331633 |
| **std** | 3.211424 | 30.860781 | 12.496092 | 10.516424 | 118.841690 | 7.027659 | 0.345488 | 10.200777 | 0.471401 |
| **min** | 0.000000 | 56.000000 | 24.000000 | 7.000000 | 14.000000 | 18.200000 | 0.085000 | 21.000000 | 0.000000 |
| **25%** | 1.000000 | 99.000000 | 62.000000 | 21.000000 | 76.750000 | 28.400000 | 0.269750 | 23.000000 | 0.000000 |
| **50%** | 2.000000 | 119.000000 | 70.000000 | 29.000000 | 125.500000 | 33.200000 | 0.449500 | 27.000000 | 0.000000 |
| **75%** | 5.000000 | 143.000000 | 78.000000 | 37.000000 | 190.000000 | 37.100000 | 0.687000 | 36.000000 | 1.000000 |
| **max** | 17.000000 | 198.000000 | 110.000000 | 63.000000 | 846.000000 | 67.100000 | 2.420000 | 81.000000 | 1.000000 |

In [8]:
```python
# Convert dataframe to numpy array
dataset = df.values
print(dataset.shape)
```

```
(392, 9)
```

In [17]:
```
print(X.shape)
print(Y.shape)
print(X[:5])
```

```
(392, 8)
(392,)
[[1.000e+00 8.900e+01 6.600e+01 2.300e+01 9.400e+01 2.810e+01 1.670e-01
  2.100e+01]
 [0.000e+00 1.370e+02 4.000e+01 3.500e+01 1.680e+02 4.310e+01 2.288e+00
  3.300e+01]
 [3.000e+00 7.800e+01 5.000e+01 3.200e+01 8.800e+01 3.100e+01 2.480e-01
  2.600e+01]
 [2.000e+00 1.970e+02 7.000e+01 4.500e+01 5.430e+02 3.050e+01 1.580e-01
  5.300e+01]
 [1.000e+00 1.890e+02 6.000e+01 2.300e+01 8.460e+02 3.010e+01 3.980e-01
  5.900e+01]]
```

In [12]:
```
print(scaler)
```

```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [13]:
```
# Transform and display the training data
X_standardized = scaler.transform(X)

data = pd.DataFrame(X_standardized)
data.describe()
```

Out[13]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **count** | 3.920000e+02 | 3.920000e+02 | 3.920000e+02 | 3.920000e+02 | 3.920000e+02 | 3.920000e+02 | 3.920000e+02 | 3.920000e+02 |
| **mean** | -4.021726e-17 | 3.129583e-17 | -4.641624e-16 | 1.042250e-16 | 6.485742e-17 | 1.543550e-16 | 3.880116e-17 | 1.028089e-16 |
| **std** | 1.001278e+00 | 1.001278e+00 | 1.001278e+00 | 1.001278e+00 | 1.001278e+00 | 1.001278e+00 | 1.001278e+00 | 1.001278e+00 |
| **min** | -1.029213e+00 | -2.161731e+00 | -3.739001e+00 | -2.108484e+00 | -1.196867e+00 | -2.120941e+00 | -1.269525e+00 | -9.682991e-01 |
| **25%** | -7.174265e-01 | -7.665958e-01 | -6.941640e-01 | -7.755315e-01 | -6.681786e-01 | -6.676780e-01 | -7.340909e-01 | -7.719850e-01 |
| **50%** | -4.056403e-01 | -1.176959e-01 | -5.314565e-02 | -1.384444e-02 | -2.574448e-01 | 1.621036e-02 | -2.131475e-01 | -3.793569e-01 |
| **75%** | 5.297185e-01 | 6.609841e-01 | 5.878727e-01 | 7.478426e-01 | 2.859877e-01 | 5.718696e-01 | 4.751644e-01 | 5.040564e-01 |
| **max** | 4.271153e+00 | 2.445459e+00 | 3.151946e+00 | 3.223325e+00 | 5.812990e+00 | 4.846172e+00 | 5.497667e+00 | 4.921123e+00 |

In [15]:

```python
# Start defining the model
def create_model():
    # create model
    model = Sequential()
    model.add(Dense(8, input_dim = 8, kernel_initializer='normal', activation='relu'))
    model.add(Dense(4, input_dim = 8, kernel_initializer='normal', activation='relu'))
    model.add(Dense(1, activation='sigmoid'))

    # compile the model
    adam = Adam(lr = 0.01)
    model.compile(loss = 'binary_crossentropy', optimizer = adam, metrics = ['accuracy'])
    return model

model = create_model()
print(model.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 8)                 72
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 36
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 5
=================================================================
Total params: 113
Trainable params: 113
Non-trainable params: 0
_________________________________________________________________
None
```

In [16]:

```python
def create_model():
    # create model
    model = Sequential()
    model.add(Dense(8, input_dim = 8, kernel_initializer='normal', activation='relu'))
    model.add(Dense(4, input_dim = 8, kernel_initializer='normal', activation='relu'))
    model.add(Dense(1, activation='sigmoid'))

    # compile the model
    adam = Adam(lr = 0.01)
    model.compile(loss = 'binary_crossentropy', optimizer = adam, metrics = ['accuracy'])
    return model
```

In [16]:

```
# Define a random seed
seed = 6
np.random.seed(seed)

# Start defining the model
def create_model():
    # create model
    model = Sequential()
    model.add(Dense(8, input_dim = 8, kernel_initializer='normal', activation='relu'))
    model.add(Dense(4, input_dim = 8, kernel_initializer='normal', activation='relu'))
    model.add(Dense(1, activation='sigmoid'))

    # compile the model
    adam = Adam(lr = 0.01)
    model.compile(loss = 'binary_crossentropy', optimizer = adam, metrics = ['accuracy'])
    return model
```

In [19]:

```
# Define a random seed
seed = 6
np.random.seed(seed)

# Start defining the model
def create_model():
    # create model
    model = Sequential()
    model.add(Dense(8, input_dim = 8, kernel_initializer='normal', activation='relu'))
    model.add(Dense(4, input_dim = 8, kernel_initializer='normal', activation='relu'))
    model.add(Dense(1, activation='sigmoid'))

    # compile the model
    adam = Adam(lr = 0.01)
    model.compile(loss = 'binary_crossentropy', optimizer = adam, metrics = ['accuracy'])
    return model

# create the model
model = KerasClassifier(build_fn = create_model, verbose = 0)

# define the grid search parameters
batch_size = [10, 20, 40]
epochs = [10, 50, 100]

# make a dictionary of the grid search parameters
param_grid = dict(batch_size=batch_size, epochs=epochs)
```

In [21]:

```
# Do a grid search for the optimal batch size and number of epochs
# Define a random seed
seed = 6
np.random.seed(seed)

# Start defining the model
def create_model():
    # create model
    model = Sequential()
    model.add(Dense(8, input_dim = 8, kernel_initializer='normal', activation='relu'))
    model.add(Dense(4, input_dim = 8, kernel_initializer='normal', activation='relu'))
    model.add(Dense(1, activation='sigmoid'))

    # compile the model
    adam = Adam(lr = 0.01)
    model.compile(loss = 'binary_crossentropy', optimizer = adam, metrics = ['accuracy'])
    return model

# create the model
model = KerasClassifier(build_fn = create_model, verbose = 0)

# define the grid search parameters
batch_size = [10, 20, 40]
epochs = [10, 50, 100]

# make a dictionary of the grid search parameters
param_grid = dict(batch_size=batch_size, epochs=epochs)

# build and fit the GridSearchCV
grid = GridSearchCV(estimator = model, param_grid = param_grid, cv = KFold(random_state=seed), verbose = 10)
grid_results = grid.fit(X_standardized, Y)

# summarize the results
print("Best: {0}, using {1}".format(grid_results.best_score_, grid_results.best_params_))
means = grid_results.cv_results_['mean_test_score']
stds = grid_results.cv_results_['std_test_score']
params = grid_results.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('{0} ({1}) with: {2}'.format(mean, stdev, param))
```

```
Fitting 3 folds for each of 9 candidates, totalling 27 fits
[CV] batch_size=10, epochs=10 ........................................
[CV]  batch_size=10, epochs=10, score=0.7480915975934677, total=   9.6s
[CV] batch_size=10, epochs=10 ........................................
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    9.7s remaining:    0.0s
[CV]  batch_size=10, epochs=10, score=0.7633587722559921, total=   7.2s
[CV] batch_size=10, epochs=10 ........................................
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:   17.0s remaining:    0.0s
[CV]  batch_size=10, epochs=10, score=0.8230769175749558, total=   6.6s
[CV] batch_size=10, epochs=50 ........................................
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:   23.7s remaining:    0.0s
[CV]  batch_size=10, epochs=50, score=0.7175572619183372, total=  55.7s
[CV] batch_size=10, epochs=50 ........................................
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:  1.3min remaining:    0.0s
[CV]  batch_size=10, epochs=50, score=0.7328244229309432, total=  14.4s
[CV] batch_size=10, epochs=50 ........................................
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:  1.6min remaining:    0.0s
[CV]  batch_size=10, epochs=50, score=0.8153846126336318, total=  14.5s
[CV] batch_size=10, epochs=100 .......................................
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:  1.8min remaining:    0.0s
[CV]  batch_size=10, epochs=100, score=0.7251908469746131, total=  17.2s
[CV] batch_size=10, epochs=100 .......................................
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:  2.1min remaining:    0.0s
[CV]  batch_size=10, epochs=100, score=0.7328244297559025, total=  15.1s
[CV] batch_size=10, epochs=100 .......................................
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:  2.4min remaining:    0.0s
[CV]  batch_size=10, epochs=100, score=0.8153846080486591, total=  15.4s
[CV] batch_size=20, epochs=10 ........................................
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:  2.6min remaining:    0.0s
[CV]  batch_size=20, epochs=10, score=0.7251908351446836, total=   8.2s
[CV] batch_size=20, epochs=10 ........................................
[CV]  batch_size=20, epochs=10, score=0.7709923695971947, total=   8.7s
[CV] batch_size=20, epochs=10 ........................................
[CV]  batch_size=20, epochs=10, score=0.8000000027509836, total=  11.3s
[CV] batch_size=20, epochs=50 ........................................
[CV]  batch_size=20, epochs=50, score=0.7557251844697326, total=   9.6s
[CV] batch_size=20, epochs=50 ........................................
[CV]  batch_size=20, epochs=50, score=0.7709923796071351, total=   9.7s
[CV] batch_size=20, epochs=50 ........................................
[CV]  batch_size=20, epochs=50, score=0.8000000165059016, total=  10.1s
[CV] batch_size=20, epochs=100 .......................................
[CV]  batch_size=20, epochs=100, score=0.748091594863484, total=  12.7s
[CV] batch_size=20, epochs=100 .......................................
[CV]  batch_size=20, epochs=100, score=0.7862595488096922, total=  14.3s
[CV] batch_size=20, epochs=100 .......................................
[CV]  batch_size=20, epochs=100, score=0.8230769359148465, total=  12.7s
[CV] batch_size=40, epochs=10 ........................................
[CV]  batch_size=40, epochs=10, score=0.7175572364384891, total=   8.0s
[CV] batch_size=40, epochs=10 ........................................
[CV]  batch_size=40, epochs=10, score=0.7709923614072436, total=   7.3s
[CV] batch_size=40, epochs=10 ........................................
[CV]  batch_size=40, epochs=10, score=0.8230769313298739, total=   7.2s
[CV] batch_size=40, epochs=50 ........................................
[CV]  batch_size=40, epochs=50, score=0.7480916048734243, total=   8.1s
[CV] batch_size=40, epochs=50 ........................................
[CV]  batch_size=40, epochs=50, score=0.7938931293159951, total=   8.5s
[CV] batch_size=40, epochs=50 ........................................
[CV]  batch_size=40, epochs=50, score=0.8307692179313073, total=   8.3s
[CV] batch_size=40, epochs=100 .......................................
[CV]  batch_size=40, epochs=100, score=0.7175572546383807, total=   9.8s
[CV] batch_size=40, epochs=100 .......................................
[CV]  batch_size=40, epochs=100, score=0.7709923695971947, total=   9.7s
[CV] batch_size=40, epochs=100 .......................................
[CV]  batch_size=40, epochs=100, score=0.799999998166011, total=   9.6s
[Parallel(n_jobs=1)]: Done  27 out of  27 | elapsed:  5.6min finished
Best: 0.7908163227292956, using {'batch_size': 40, 'epochs': 50}
0.7780612187117947 (0.0323174727132797) with: {'batch_size': 10, 'epochs': 10}
0.7551020417286425 (0.04291935222102097) with: {'batch_size': 10, 'epochs': 50}
0.7576530619847531 (0.0407857977678057) with: {'batch_size': 10, 'epochs': 100}
0.7653061229051376 (0.030785719311948196) with: {'batch_size': 20, 'epochs': 10}
0.7755102118363186 (0.018344839015040526) with: {'batch_size': 20, 'epochs': 50}
0.7857142895156023 (0.030595321909261793) with: {'batch_size': 20, 'epochs': 100}
0.7704081591598841 (0.04305242641100368) with: {'batch_size': 40, 'epochs': 10}
0.7908163227292956 (0.0338015720829859) with: {'batch_size': 40, 'epochs': 50}
0.7627551034092903 (0.03413789315396789) with: {'batch_size': 40, 'epochs': 100}
```

```
In [15]: # Do a grid search for the optimal batch size and number of epochs
from keras.layers import Dropout

# Define a random seed
seed = 6
np.random.seed(seed)

# Start defining the model
def create_model():
    # create model
    model = Sequential()
    model.add(Dense(8, input_dim = 8, kernel_initializer='normal', activation='relu'))
    model.add(Dense(4, input_dim = 8, kernel_initializer='normal', activation='relu'))
    model.add(Dense(1, activation='sigmoid'))

    # compile the model
    adam = Adam(lr = learn_rate)
    model.compile(loss = 'binary_crossentropy', optimizer = adam, metrics = ['accuracy'])
    return model

# create the model
model = KerasClassifier(build_fn = create_model, verbose = 0)

# define the grid search parameters
batch_size = [10, 20, 40]
epochs = [10, 50, 100]

# make a dictionary of the grid search parameters
param_grid = dict(batch_size=batch_size, epochs=epochs)

# build and fit the GridSearchCV
grid = GridSearchCV(estimator = model, param_grid = param_grid, cv = KFold(random_state=seed), verbose = 10)
grid_results = grid.fit(X_standardized, Y)

# summarize the results
print("Best: {0}, using {1}".format(grid_results.best_score_, grid_results.best_params_))
means = grid_results.cv_results_['mean_test_score']
stds = grid_results.cv_results_['std_test_score']
params = grid_results.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('{0} ({1}) with: {2}'.format(mean, stdev, param))
```

```
# Start defining the model
def create_model(learn_rate, dropout_rate):
    # create model
    model = Sequential()
    model.add(Dense(8, input_dim = 8, kernel_initializer='normal', activation='relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(4, input_dim = 8, kernel_initializer='normal', activation='relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(1, activation='sigmoid'))
```

```
# create the model
model = KerasClassifier(build_fn = create_model, epochs = 100, batch_size = 20, verbose = 0)
```

```python
# define the grid search parameters
learn_rate = [0.001, 0.01, 0.1]
dropout_rate = [0.0, 0.1, 0.2]
```

```python
# make a dictionary of the grid search parameters
param_grid = dict(learn_rate=learn_rate, dropout_rate=dropout_rate)
```

```
Fitting 3 folds for each of 9 candidates, totalling 27 fits
[CV] dropout_rate=0.0, learn_rate=0.001 ..............................
[CV]  dropout_rate=0.0, learn_rate=0.001, score=0.74809161397337, total=  26.3s
[CV] dropout_rate=0.0, learn_rate=0.001 ..............................
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   26.4s remaining:    0.0s
[CV]  dropout_rate=0.0, learn_rate=0.001, score=0.778625960113438, total=   5.1s
[CV] dropout_rate=0.0, learn_rate=0.001 ..............................
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:   31.6s remaining:    0.0s
[CV]  dropout_rate=0.0, learn_rate=0.001, score=0.8461538553237915, total=   5.0s
[CV] dropout_rate=0.0, learn_rate=0.01 ...............................
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:   36.7s remaining:    0.0s
[CV]  dropout_rate=0.0, learn_rate=0.01, score=0.740458014357181, total=   5.0s
[CV] dropout_rate=0.0, learn_rate=0.01 ...............................
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:   41.8s remaining:    0.0s
[CV]  dropout_rate=0.0, learn_rate=0.01, score=0.7862595406197409, total=   5.0s
[CV] dropout_rate=0.0, learn_rate=0.01 ...............................
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:   46.9s remaining:    0.0s
[CV]  dropout_rate=0.0, learn_rate=0.01, score=0.7769230741720933, total=   5.1s
[CV] dropout_rate=0.0, learn_rate=0.1 ................................
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:   52.1s remaining:    0.0s
[CV]  dropout_rate=0.0, learn_rate=0.1, score=0.6946564858196346, total=   5.0s
[CV] dropout_rate=0.0, learn_rate=0.1 ................................
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:   57.2s remaining:    0.0s
[CV]  dropout_rate=0.0, learn_rate=0.1, score=0.763358779990946, total=   5.1s
[CV] dropout_rate=0.0, learn_rate=0.1 ................................
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:  1.0min remaining:    0.0s
[CV]  dropout_rate=0.0, learn_rate=0.1, score=0.800000011920929, total=   5.7s
[CV] dropout_rate=0.1, learn_rate=0.001 ..............................
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:  1.1min remaining:    0.0s
[CV]  dropout_rate=0.1, learn_rate=0.001, score=0.725190845154624, total=   5.7s
[CV] dropout_rate=0.1, learn_rate=0.001 ..............................
[CV]  dropout_rate=0.1, learn_rate=0.001, score=0.7633587900008864, total=   5.7s
[CV] dropout_rate=0.1, learn_rate=0.001 ..............................
[CV]  dropout_rate=0.1, learn_rate=0.001, score=0.8384615366275494, total=   5.9s
[CV] dropout_rate=0.1, learn_rate=0.01 ...............................
[CV]  dropout_rate=0.1, learn_rate=0.01, score=0.7404580234571267, total=   6.4s
[CV] dropout_rate=0.1, learn_rate=0.01 ...............................
[CV]  dropout_rate=0.1, learn_rate=0.01, score=0.7480916048734243, total=   6.0s
[CV] dropout_rate=0.1, learn_rate=0.01 ...............................
[CV]  dropout_rate=0.1, learn_rate=0.01, score=0.830769236271198, total=   6.5s
[CV] dropout_rate=0.1, learn_rate=0.1 ................................
[CV]  dropout_rate=0.1, learn_rate=0.1, score=0.7099236600271618, total=   6.4s
[CV] dropout_rate=0.1, learn_rate=0.1 ................................
[CV]  dropout_rate=0.1, learn_rate=0.1, score=0.770992373692170З, total=   6.1s
[CV] dropout_rate=0.1, learn_rate=0.1 ................................
[CV]  dropout_rate=0.1, learn_rate=0.1, score=0.7769230833420386, total=   7.1s
[CV] dropout_rate=0.2, learn_rate=0.001 ..............................
[CV]  dropout_rate=0.2, learn_rate=0.001, score=0.7404580152671756, total=   7.5s
[CV] dropout_rate=0.2, learn_rate=0.001 ..............................
[CV]  dropout_rate=0.2, learn_rate=0.001, score=0.7709923705071894, total=   7.3s
[CV] dropout_rate=0.2, learn_rate=0.001 ..............................
[CV]  dropout_rate=0.2, learn_rate=0.001, score=0.8384615457974948, total=   7.2s
[CV] dropout_rate=0.2, learn_rate=0.01 ...............................
[CV]  dropout_rate=0.2, learn_rate=0.01, score=0.740458014357181, total=   7.3s
[CV] dropout_rate=0.2, learn_rate=0.01 ...............................
[CV]  dropout_rate=0.2, learn_rate=0.01, score=0.770992360497249, total=   6.6s
[CV] dropout_rate=0.2, learn_rate=0.01 ...............................
[CV]  dropout_rate=0.2, learn_rate=0.01, score=0.8153846218035772, total=   6.7s
[CV] dropout_rate=0.2, learn_rate=0.1 ................................
[CV]  dropout_rate=0.2, learn_rate=0.1, score=0.7099236600271618, total=   6.9s
[CV] dropout_rate=0.2, learn_rate=0.1 ................................
[CV]  dropout_rate=0.2, learn_rate=0.1, score=0.748091610788389, total=   6.8s
[CV] dropout_rate=0.2, learn_rate=0.1 ................................
[CV]  dropout_rate=0.2, learn_rate=0.1, score=0.699999988079071, total=   6.9s
[Parallel(n_jobs=1)]: Done  27 out of  27 | elapsed:  3.2min finished
Best: 0.7908163351976142, using {'dropout_rate': 0.0, 'learn_rate': 0.001}
0.7908163351976142 (0.04092945245626919) with: {'dropout_rate': 0.0, 'learn_rate': 0.001}
0.7678571411845635 (0.019781398964752953) with: {'dropout_rate': 0.0, 'learn_rate': 0.01}
0.752551023905З745 (0.0436552908563337) with: {'dropout_rate': 0.0, 'learn_rate': 0.1}
0.7755102090993706 (0.04700773782551412) with: {'dropout_rate': 0.1, 'learn_rate': 0.001}
0.77295918884326 (0.040840932450670844) with: {'dropout_rate': 0.1, 'learn_rate': 0.01}
0.7525510236012692 (0.03029656314273123) with: {'dropout_rate': 0.1, 'learn_rate': 0.1}
0.783163269107439 (0.0409031279769419) with: {'dropout_rate': 0.2, 'learn_rate': 0.001}
0.77551020392958 (0.030736028865651223) with: {'dropout_rate': 0.2, 'learn_rate': 0.01}
0.7193877523650929 (0.020734660668352178) with: {'dropout_rate': 0.2, 'learn_rate': 0.1}
```

```
In [20]: # Do a grid search for learning rate and dropout rate
         # import necessary packages

         # Define a random seed
         seed = 6
         np.random.seed(seed)

         # Start defining the model
         def create_model(activation, init):
             # create model
             model = Sequential()
             model.add(Dense(8, input_dim = 8, kernel_initializer= init, activation= activation))
             model.add(Dense(4, input_dim = 8, kernel_initializer= init, activation= activation))
             model.add(Dense(1, activation='sigmoid'))

             # compile the model
             adam = Adam(lr = 0.001)
             model.compile(loss = 'binary_crossentropy', optimizer = adam, metrics = ['accuracy'])
             return model


         # create the model
         model = KerasClassifier(build_fn = create_model, epochs = 100, batch_size = 20, verbose = 0)

         # define the grid search parameters
         learn_rate = [0.001, 0.01, 0.1]
         dropout_rate = [0.0, 0.1, 0.2]

         # make a dictionary of the grid search parameters
         param_grid = dict(learn_rate=learn_rate, dropout_rate=dropout_rate)

         # build and fit the GridSearchCV
         grid = GridSearchCV(estimator = model, param_grid = param_grid, cv = KFold(random_state=seed), verbose = 10)
         grid_results = grid.fit(X_standardized, Y)
```

```
Fitting 3 folds for each of 12 candidates, totalling 36 fits
[CV] activation=softmax, init=uniform ................................
[CV]  activation=softmax, init=uniform, score=0.7557252035796187, total=   5.2s
[CV] activation=softmax, init=uniform ................................

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    5.2s remaining:    0.0s

[CV]  activation=softmax, init=uniform, score=0.7557252003946378, total=   5.5s
[CV] activation=softmax, init=uniform ................................

[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:   10.9s remaining:    0.0s

[CV]  activation=softmax, init=uniform, score=0.8153846218035772, total=   6.5s
[CV] activation=softmax, init=normal .................................

[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:   17.4s remaining:    0.0s

[CV]  activation=softmax, init=normal, score=0.6106870242657553, total=   6.1s
[CV] activation=softmax, init=normal .................................

[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:   23.6s remaining:    0.0s

[CV]  activation=softmax, init=normal, score=0.7557252003946378, total=   5.7s
[CV] activation=softmax, init=normal .................................

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:   29.4s remaining:    0.0s

[CV]  activation=softmax, init=normal, score=0.8230769267449012, total=   5.1s
[CV] activation=softmax, init=zero ...................................

[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:   34.6s remaining:    0.0s

[CV]  activation=softmax, init=zero, score=0.6106870242657553, total=   5.2s
[CV] activation=softmax, init=zero ...................................

[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:   40.0s remaining:    0.0s

[CV]  activation=softmax, init=zero, score=0.6946564958295749, total=   5.5s
[CV] activation=softmax, init=zero ...................................

[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:   45.5s remaining:    0.0s

[CV]  activation=softmax, init=zero, score=0.699999988079071, total=   5.2s
[CV] activation=relu, init=uniform ...................................

[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:   50.8s remaining:    0.0s

[CV]  activation=relu, init=uniform, score=0.7328244347608727, total=   5.5s
[CV] activation=relu, init=uniform ...................................
[CV]  activation=relu, init=uniform, score=0.748091610788389, total=   5.3s
[CV] activation=relu, init=uniform ...................................
[CV]  activation=relu, init=uniform, score=0.823076921599286, total=   5.6s
[CV] activation=relu, init=normal ....................................
[CV]  activation=relu, init=normal, score=0.7251908351446836, total=   5.4s
[CV] activation=relu, init=normal ....................................
[CV]  activation=relu, init=normal, score=0.7709923705071894, total=   5.5s
[CV] activation=relu, init=normal ....................................
[CV]  activation=relu, init=normal, score=0.8461538599087641, total=   5.8s
[CV] activation=relu, init=zero ......................................
[CV]  activation=relu, init=zero, score=0.6106870242657553, total=   5.7s
[CV] activation=relu, init=zero ......................................
[CV]  activation=relu, init=zero, score=0.6946564958295749, total=   5.8s
[CV] activation=relu, init=zero ......................................
[CV]  activation=relu, init=zero, score=0.699999988079071, total=   5.9s
[CV] activation=tanh, init=uniform ...................................
[CV]  activation=tanh, init=uniform, score=0.755725194479673, total=   6.3s
[CV] activation=tanh, init=uniform ...................................
[CV]  activation=tanh, init=uniform, score=0.7709923796071351, total=   6.1s
[CV] activation=tanh, init=uniform ...................................
[CV]  activation=tanh, init=uniform, score=0.8230769267449012, total=   6.4s
[CV] activation=tanh, init=normal ....................................
[CV]  activation=tanh, init=normal, score=0.7633587840859216, total=   6.1s
[CV] activation=tanh, init=normal ....................................
[CV]  activation=tanh, init=normal, score=0.7786259692133838, total=   6.4s
[CV] activation=tanh, init=normal ....................................
[CV]  activation=tanh, init=normal, score=0.8384615366275494, total=   7.1s
[CV] activation=tanh, init=zero ......................................
[CV]  activation=tanh, init=zero, score=0.6106870242657553, total=   7.3s
[CV] activation=tanh, init=zero ......................................
[CV]  activation=tanh, init=zero, score=0.6946564958295749, total=   6.9s
[CV] activation=tanh, init=zero ......................................
[CV]  activation=tanh, init=zero, score=0.699999988079071, total=   6.8s
[CV] activation=linear, init=uniform .................................
[CV]  activation=linear, init=uniform, score=0.7709923645922245, total=   7.1s
[CV] activation=linear, init=uniform .................................
[CV]  activation=linear, init=uniform, score=0.7633587900008864, total=   6.8s
[CV] activation=linear, init=uniform .................................
[CV]  activation=linear, init=uniform, score=0.8461538553237915, total=   7.0s
[CV] activation=linear, init=normal ..................................
[CV]  activation=linear, init=normal, score=0.7709923554922788, total=   7.3s
[CV] activation=linear, init=normal ..................................
[CV]  activation=linear, init=normal, score=0.7633587900008864, total=   7.2s
[CV] activation=linear, init=normal ..................................
[CV]  activation=linear, init=normal, score=0.8384615457974948, total=   7.5s
[CV] activation=linear, init=zero ....................................
[CV]  activation=linear, init=zero, score=0.6106870242657553, total=   7.3s
[CV] activation=linear, init=zero ....................................
[CV]  activation=linear, init=zero, score=0.6946564958295749, total=   7.5s
[CV] activation=linear, init=zero ....................................
[CV]  activation=linear, init=zero, score=0.699999988079071, total=   7.6s

[Parallel(n_jobs=1)]: Done  36 out of  36 | elapsed:  3.8min finished

Best: 0.7933673531729348, using {'activation': 'tanh', 'init': 'normal'}
0.7755102136609505 (0.028087641954935023) with: {'activation': 'softmax', 'init': 'uniform'}
0.7295918416003792 (0.08860773018104312) with: {'activation': 'softmax', 'init': 'normal'}
0.6683673458744068 (0.040922317011154695) with: {'activation': 'softmax', 'init': 'zero'}
0.7678571475707755 (0.03939442059435707) with: {'activation': 'relu', 'init': 'uniform'}
0.7806122493074865 (0.049819449702508574) with: {'activation': 'relu', 'init': 'normal'}
0.6683673458744068 (0.040922317011154695) with: {'activation': 'relu', 'init': 'zero'}
0.7831632721484924 (0.02879958859349298) with: {'activation': 'tanh', 'init': 'uniform'}
0.7933673531729348 (0.032371719468446684) with: {'activation': 'tanh', 'init': 'normal'}
0.6683673458744068 (0.040922317011154695) with: {'activation': 'tanh', 'init': 'zero'}
0.7933673531729348 (0.037313655933021314) with: {'activation': 'linear', 'init': 'uniform'}
0.7908163291155076 (0.03370616593390663) with: {'activation': 'linear', 'init': 'normal'}
0.6683673458744068 (0.040922317011154695) with: {'activation': 'linear', 'init': 'zero'}
```

```
Fitting 3 folds for each of 9 candidates, totalling 27 fits
[CV] neuron1=4, neuron2=2 ...........................................
[CV] ... neuron1=4, neuron2=2, score=0.7709923645922245, total=  14.9s
[CV] neuron1=4, neuron2=2 ...........................................
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   15.0s remaining:    0.0s
[CV] ... neuron1=4, neuron2=2, score=0.7633587900008864, total=  13.3s
[CV] neuron1=4, neuron2=2 ...........................................
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:   28.4s remaining:    0.0s
[CV] ... neuron1=4, neuron2=2, score=0.8230769267449012, total=  17.4s
[CV] neuron1=4, neuron2=4 ...........................................
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:   46.0s remaining:    0.0s
[CV] ... neuron1=4, neuron2=4, score=0.7709923645922245, total=  11.5s
[CV] neuron1=4, neuron2=4 ...........................................
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:   57.6s remaining:    0.0s
[CV] ... neuron1=4, neuron2=4, score=0.7786259692133838, total=  10.7s
[CV] neuron1=4, neuron2=4 ...........................................
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:  1.1min remaining:    0.0s
[CV] ... neuron1=4, neuron2=4, score=0.8153846218035772, total=  11.3s
[CV] neuron1=4, neuron2=8 ...........................................
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:  1.3min remaining:    0.0s
[CV] ... neuron1=4, neuron2=8, score=0.7633587749859759, total=  17.1s
[CV] neuron1=4, neuron2=8 ...........................................
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:  1.6min remaining:    0.0s
[CV] ... neuron1=4, neuron2=8, score=0.7633587900008864, total=  13.7s
[CV] neuron1=4, neuron2=8 ...........................................
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:  1.8min remaining:    0.0s
[CV] .... neuron1=4, neuron2=8, score=0.830769236271198, total=  12.5s
[CV] neuron1=8, neuron2=2 ...........................................
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:  2.1min remaining:    0.0s
[CV] ... neuron1=8, neuron2=2, score=0.7633587840859216, total=  17.1s
[CV] neuron1=8, neuron2=2 ...........................................
[CV] ... neuron1=8, neuron2=2, score=0.7633587900008864, total=  13.9s
[CV] neuron1=8, neuron2=2 ...........................................
[CV] ... neuron1=8, neuron2=2, score=0.8384615457974948, total=  18.5s
[CV] neuron1=8, neuron2=4 ...........................................
[CV] ... neuron1=8, neuron2=4, score=0.7633587749859759, total=  17.8s
[CV] neuron1=8, neuron2=4 ...........................................
[CV] ... neuron1=8, neuron2=4, score=0.7633587900008864, total=  14.4s
[CV] neuron1=8, neuron2=4 ...........................................
[CV] ... neuron1=8, neuron2=4, score=0.8384615457974948, total=  12.0s
[CV] neuron1=8, neuron2=8 ...........................................
[CV] ... neuron1=8, neuron2=8, score=0.7633587840859216, total=  11.5s
[CV] neuron1=8, neuron2=8 ...........................................
[CV] ... neuron1=8, neuron2=8, score=0.7633587900008864, total=  11.7s
[CV] neuron1=8, neuron2=8 ...........................................
[CV] .... neuron1=8, neuron2=8, score=0.830769236271198, total=  12.3s
[CV] neuron1=16, neuron2=2 ..........................................
[CV] .. neuron1=16, neuron2=2, score=0.7633587840859216, total=  13.5s
[CV] neuron1=16, neuron2=2 ..........................................
[CV] .. neuron1=16, neuron2=2, score=0.7633587900008864, total=  14.7s
[CV] neuron1=16, neuron2=2 ..........................................
[CV] .. neuron1=16, neuron2=2, score=0.8461538553237915, total=  13.2s
[CV] neuron1=16, neuron2=4 ..........................................
[CV] .. neuron1=16, neuron2=4, score=0.7709923645922245, total=  12.7s
[CV] neuron1=16, neuron2=4 ..........................................
[CV] .. neuron1=16, neuron2=4, score=0.7633587900008864, total=  13.9s
[CV] neuron1=16, neuron2=4 ..........................................
[CV] ... neuron1=16, neuron2=4, score=0.838461541212522, total=  12.4s
[CV] neuron1=16, neuron2=8 ..........................................
[CV] .. neuron1=16, neuron2=8, score=0.7633587840859216, total=  12.7s
[CV] neuron1=16, neuron2=8 ..........................................
[CV] .. neuron1=16, neuron2=8, score=0.7633587900008864, total=  12.3s
[CV] neuron1=16, neuron2=8 ..........................................
[CV] ... neuron1=16, neuron2=8, score=0.830769236271198, total= 2.9min
[Parallel(n_jobs=1)]: Done  27 out of  27 | elapsed:  8.9min finished
Best: 0.7908163351976142, using {'neuron1': 16, 'neuron2': 2}
0.785714290123813 (0.02650267677503863) with: {'neuron1': 4, 'neuron2': 2}
0.7882653126607135 (0.019356087898682556) with: {'neuron1': 4, 'neuron2': 4}
0.785714290123813 (0.03173682706863349) with: {'neuron1': 4, 'neuron2': 8}
0.7882653141812402 (0.035358362588889225) with: {'neuron1': 8, 'neuron2': 2}
0.7882653111401869 (0.03535836473101593) with: {'neuron1': 8, 'neuron2': 4}
0.7857142931648663 (0.031736824926506764) with: {'neuron1': 8, 'neuron2': 8}
0.7908163351976142 (0.03897990025127174) with: {'neuron1': 16, 'neuron2': 2}
0.7908163306360342 (0.03370616199600475) with: {'neuron1': 16, 'neuron2': 4}
0.7857142931648663 (0.031736824926506764) with: {'neuron1': 16, 'neuron2': 8}
```

```
In [23]: print(y_pred.shape)

(392, 1)
```

```
In [24]: print(y_pred[:5])

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

```
In [25]: # Generate a classification report
         from sklearn.metrics import classification_report, accuracy_score

         print(accuracy_score(Y, y_pred))
         print(classification_report(Y, y_pred))
```

```
0.7806122448979592
             precision    recall  f1-score   support

          0       0.81      0.89      0.84       262
          1       0.71      0.57      0.63       130

avg / total       0.77      0.78      0.77       392
```

```
In [23]: example = df.iloc[1]
         print(example)
```

```
n_pregnant                   0.000
glucose_concentration      137.000
blood_pressure (mm Hg)      40.000
skin_thickness (mm)         35.000
serum_insulin (mu U/ml)    168.000
BMI                         43.100
pedigree_function            2.288
age                         33.000
class                        1.000
Name: 4, dtype: float64
```

```
In [27]: prediction = grid.predict(X_standardized[1].reshape(1, -1))
         print(prediction)
```

```
[[1]]
```

# Chapter 3: DNA Classification

```
Python: 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32:41) [MSC v.1900 64 bit (AMD64)]
Numpy: 1.14.3
Sklearn: 0.19.1
Pandas: 0.23.0
```

```
In [3]: print(data.iloc[0])

Class                                                           +
id                                                            S10
Sequence    \t\ttactagcaatacgcttgcgttcggtggttaagtatgtataat...
Name: 0, dtype: object
```

```
0    +
1    +
2    +
3    +
4    +
Name: Class, dtype: object
```

```
['t', 'a', 'c', 't', 'a', 'g', 'c', 'a', 'a', 't', 'a', 'c', 'g', 'c', 't', 't', 'g', 'c',
'g', 't', 't', 'c', 'g', 'g', 't', 'g', 'g', 't', 't', 'a', 'a', 'g', 't', 'a', 't', 'g',
't', 'a', 't', 'a', 'a', 't', 'g', 'c', 'g', 'c', 'g', 'g', 'g', 'c', 't', 't', 'g', 't',
'c', 'g', 't', '+']
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | t | t | g | a | t | a | c | t | c | t | ... | c | c | t | a | g | c | g | c | c | t |
| 1 | a | g | t | a | c | g | a | t | g | t | ... | c | g | a | g | a | c | t | g | t | a |
| 2 | c | c | a | t | g | g | g | t | a | t | ... | g | c | t | a | g | t | a | c | c | a |
| 3 | t | t | c | t | a | g | g | c | c | t | ... | a | t | g | g | a | c | t | g | g | c |
| 4 | a | a | t | g | t | g | g | t | t | a | ... | g | a | a | g | g | a | t | a | t | a |
| 5 | g | t | a | t | a | c | g | a | t | a | ... | t | g | c | g | c | a | c | c | c | t |
| 6 | c | c | g | g | a | a | g | c | a | a | ... | a | g | c | t | a | t | t | t | c | t |
| 7 | a | c | a | a | t | a | t | a | a | t | ... | g | a | g | g | t | g | c | a | t | a |
| 8 | a | t | g | t | t | g | g | a | t | t | ... | a | c | a | t | g | g | a | c | c | a |
| 9 | t | g | a | g | a | g | g | a | a | t | ... | c | t | a | a | t | c | a | g | a | t |
| 10 | a | a | a | t | a | a | a | a | t | c | ... | c | t | c | c | c | c | c | a | a | a |
| 11 | c | c | c | g | c | g | g | c | a | c | ... | c | t | g | t | a | t | a | t | t | a |
| 12 | g | a | t | t | t | g | g | a | c | t | ... | t | c | a | c | g | c | a | g | g | a |
| 13 | c | g | a | a | a | a | a | c | t | c | ... | t | t | g | c | c | t | g | a | g | t |
| 14 | t | t | g | t | t | t | t | t | g | t | ... | a | t | t | a | c | a | a | g | c | a |
| 15 | t | t | t | c | t | g | t | t | c | t | ... | g | g | c | a | t | a | t | a | c | a |
| 16 | g | g | g | g | g | g | t | g | g | g | ... | a | t | a | g | c | a | t | t | t | g |
| 17 | c | t | c | a | a | a | a | a | a | t | ... | g | t | a | a | g | c | a | g | c | g |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | t | a | c | t | a | g | c | a | a | t | ... | g | c | t | t | g | t | c | g | t | + |
| 1 | t | g | c | t | a | t | c | c | t | g | ... | c | a | t | c | g | c | c | a | a | + |
| 2 | g | t | a | c | t | a | g | a | g | a | ... | c | a | c | c | c | g | g | c | g | + |
| 3 | a | a | t | t | g | t | g | a | t | g | ... | a | a | c | a | a | a | c | t | c | + |
| 4 | t | c | g | a | t | a | a | t | t | a | ... | c | c | g | t | g | g | t | a | g | + |

[5 rows x 58 columns]

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | Class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | t | a | c | t | a | g | c | a | a | t | ... | g | c | t | t | g | t | c | g | t | + |
| 1 | t | g | c | t | a | t | c | c | t | g | ... | c | a | t | c | g | c | c | a | a | + |
| 2 | g | t | a | c | t | a | g | a | g | a | ... | c | a | c | c | c | g | g | c | g | + |
| 3 | a | a | t | t | g | t | g | a | t | g | ... | a | a | c | a | a | a | c | t | c | + |
| 4 | t | c | g | a | t | a | a | t | t | a | ... | c | c | g | t | g | g | t | a | g | + |

[5 rows x 58 columns]

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | Class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 106 | 106 | 106 | 106 | 106 | 106 | 106 | 106 | 106 | 106 | ... | 106 | 106 | 106 | 106 | 106 | 106 | 106 | 106 | 106 | 106 |
| **unique** | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | ... | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 2 |
| **top** | t | a | a | c | a | a | a | a | a | a | ... | c | c | c | t | t | c | c | c | t | - |
| **freq** | 38 | 34 | 30 | 30 | 36 | 42 | 38 | 34 | 33 | 36 | ... | 36 | 42 | 31 | 33 | 35 | 32 | 29 | 29 | 34 | 53 |

4 rows × 58 columns

```
      0     1     2     3     4     5     6     7     8     9   ...    48  \
t  38.0  26.0  27.0  26.0  22.0  24.0  30.0  32.0  32.0  28.0  ...  21.0
c  27.0  22.0  21.0  30.0  19.0  18.0  21.0  20.0  22.0  22.0  ...  36.0
a  26.0  34.0  30.0  22.0  36.0  42.0  38.0  34.0  33.0  36.0  ...  23.0
g  15.0  24.0  28.0  28.0  29.0  22.0  17.0  20.0  19.0  20.0  ...  26.0
-   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN  ...   NaN
+   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN  ...   NaN

     49    50    51    52    53    54    55    56  Class
t  22.0  23.0  33.0  35.0  30.0  23.0  29.0  34.0    NaN
c  42.0  31.0  32.0  21.0  32.0  29.0  29.0  17.0    NaN
a  24.0  28.0  27.0  25.0  22.0  26.0  24.0  27.0    NaN
g  18.0  24.0  14.0  25.0  22.0  28.0  24.0  28.0    NaN
-   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   53.0
+   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   53.0

[6 rows x 58 columns]
```

| | 0_a | 0_c | 0_g | 0_t | 1_a | 1_c | 1_g | 1_t | 2_a | 2_c | ... | 55_a | 55_c | 55_g | 55_t | 56_a | 56_c | 56_g | 56_t | Class_+ | Class_- |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | ... | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| **1** | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | ... | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **2** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | ... | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| **3** | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| **4** | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | ... | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

5 rows × 230 columns

```
   0_a  0_c  0_g  0_t  1_a  1_c  1_g  1_t  2_a  2_c  ...   54_t  55_a  55_c  \
0    0    0    0    1    1    0    0    0    0    1  ...      0     0     0
1    0    0    0    1    0    0    1    0    0    1  ...      0     1     0
2    0    0    1    0    0    0    0    1    1    0  ...      0     0     1
3    1    0    0    0    1    0    0    0    0    0  ...      0     0     0
4    0    0    0    1    0    1    0    0    0    0  ...      1     1     0

   55_g  55_t  56_a  56_c  56_g  56_t  Class
0     1     0     0     0     0     1      1
1     0     0     1     0     0     0      1
2     0     0     0     0     1     0      1
3     0     1     0     1     0     0      1
4     0     0     0     0     1     0      1

[5 rows x 229 columns]
```

```
0_a      0
0_c      0
0_g      1
0_t      0
1_a      1
1_c      0
1_g      0
1_t      0
2_a      0
2_c      0
2_g      1
2_t      0
3_a      0
3_c      0
3_g      1
3_t      0
4_a      0
4_c      0
4_g      0
4_t      1
5_a      0
5_c      0
5_g      1
5_t      0
6_a      0
6_c      0
6_g      1
6_t      0
7_a      0
7_c      1
        ..
49_t     1
50_a     0
50_c     0
50_g     1
50_t     0
51_a     0
51_c     0
51_g     1
51_t     0
52_a     0
52_c     0
52_g     0
52_t     1
53_a     1
53_c     0
53_g     0
53_t     0
54_a     0
54_c     0
54_g     0
54_t     1
55_a     0
55_c     0
55_g     0
55_t     1
56_a     1
56_c     0
56_g     0
56_t     0
Class    0
Name: 60, Length: 229, dtype: uint8
```

```
Nearest Neighbors: 0.823214 (0.113908)
Gaussian Process: 0.873214 (0.056158)
Decision Tree: 0.698214 (0.201628)
Random Forest: 0.607143 (0.162882)

Neural Net: 0.875000 (0.096825)
AdaBoost: 0.925000 (0.114564)
Naive Bayes: 0.837500 (0.137500)
SVM Linear: 0.850000 (0.108972)
SVM RBF: 0.737500 (0.117925)
SVM Sigmoid: 0.569643 (0.159209)
```

```
Nearest Neighbors
0.7777777777777778
             precision    recall  f1-score   support

          0       1.00      0.65      0.79        17
          1       0.62      1.00      0.77        10

avg / total       0.86      0.78      0.78        27
```

```
Gaussian Process
0.8888888888888888
             precision    recall  f1-score   support

          0       1.00      0.82      0.90        17
          1       0.77      1.00      0.87        10

avg / total       0.91      0.89      0.89        27
```

```
Decision Tree
0.7777777777777778
             precision    recall  f1-score   support

          0       1.00      0.65      0.79        17
          1       0.62      1.00      0.77        10

avg / total       0.86      0.78      0.78        27
```

```
Random Forest
0.5925925925925926
             precision    recall  f1-score   support

          0       0.88      0.41      0.56        17
          1       0.47      0.90      0.62        10

avg / total       0.73      0.59      0.58        27
```

```
Neural Net
0.9259259259259259
             precision    recall  f1-score   support

          0       1.00      0.88      0.94        17
          1       0.83      1.00      0.91        10

avg / total       0.94      0.93      0.93        27
```

```
AdaBoost
0.8518518518518519
             precision    recall  f1-score   support

          0       1.00      0.76      0.87        17
          1       0.71      1.00      0.83        10

avg / total       0.89      0.85      0.85        27
```

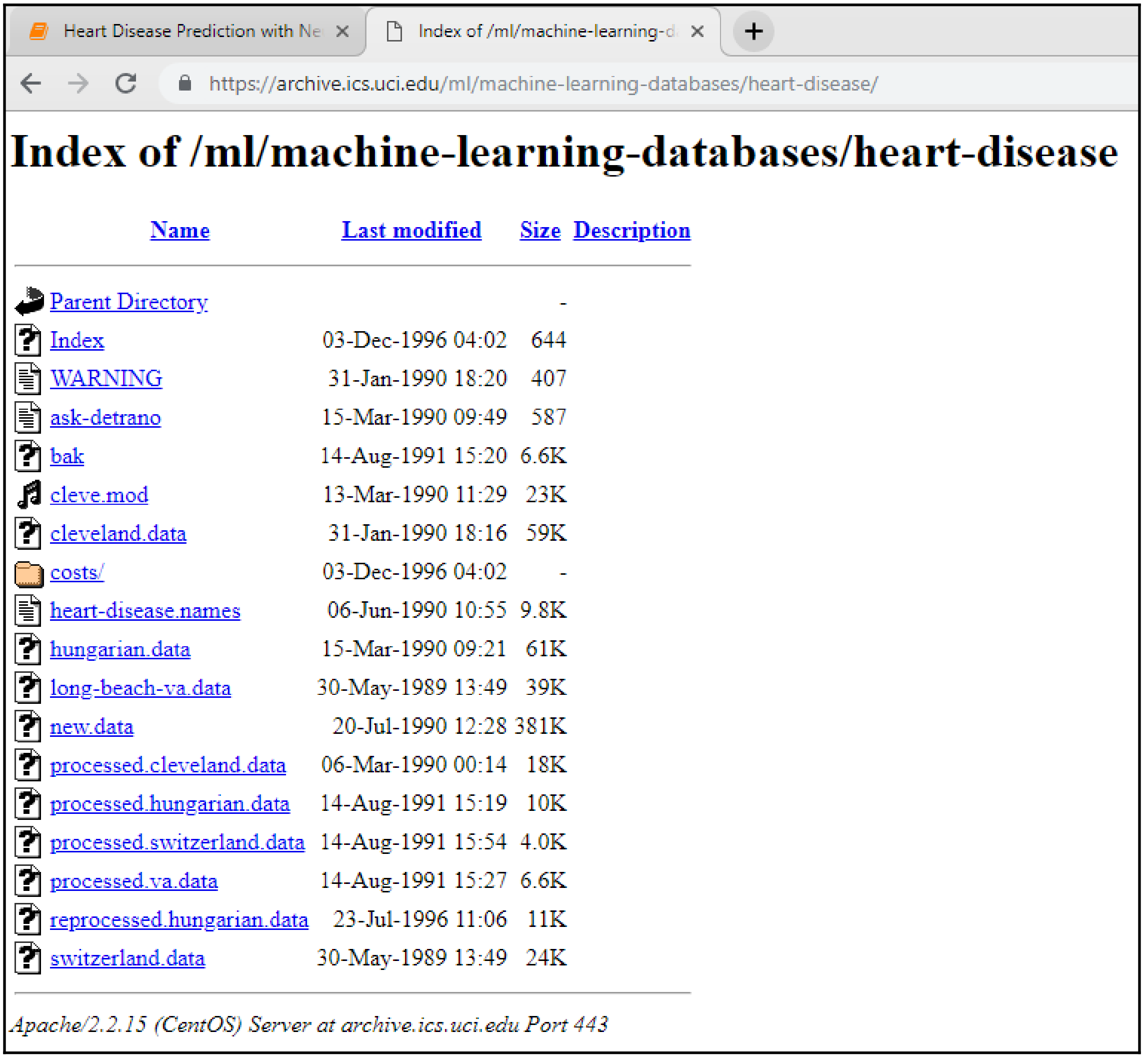
```
Naive Bayes
0.9259259259259259
             precision    recall  f1-score   support

          0       1.00      0.88      0.94        17
          1       0.83      1.00      0.91        10

avg / total       0.94      0.93      0.93        27
```

```
SVM Linear
0.9629629629629629
             precision    recall  f1-score   support

          0       1.00      0.94      0.97        17
          1       0.91      1.00      0.95        10

avg / total       0.97      0.96      0.96        27
```

```
SVM RBF
0.7777777777777778
             precision    recall  f1-score   support

          0       1.00      0.65      0.79        17
          1       0.62      1.00      0.77        10

avg / total       0.86      0.78      0.78        27
```

```
SVM Sigmoid
0.4444444444444444
             precision    recall  f1-score   support

          0       1.00      0.12      0.21        17
          1       0.40      1.00      0.57        10

avg / total       0.78      0.44      0.34        27
```

# Chapter 4: Diagnosing Coronary Artery Disease



```
Python: 2.7.13 |Continuum Analytics, Inc.| (default, May 11 2017, 13:17:26) [MSC v.1500 64 bit (AMD64)]
Pandas: 0.21.0
Numpy: 1.14.3
Sklearn: 0.19.1
Matplotlib: 2.1.0
Keras: 2.1.4
```

```
Shape of DataFrame: (303, 14)
age          67
sex           1
cp            4
trestbps    160
chol        286
fbs           0
restecg       2
thalach     108
exang         1
oldpeak     1.5
slope         2
ca          3.0
thal        3.0
class         2
Name: 1, dtype: object
```

In [25]:
```
# print the last twenty or so data points
cleveland.loc[280:]
```

Out[25]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **280** | 57.0 | 1.0 | 4.0 | 110.0 | 335.0 | 0.0 | 0.0 | 143.0 | 1.0 | 3.0 | 2.0 | 1.0 | 7.0 | 2 |
| **281** | 47.0 | 1.0 | 3.0 | 130.0 | 253.0 | 0.0 | 0.0 | 179.0 | 0.0 | 0.0 | 1.0 | 0.0 | 3.0 | 0 |
| **282** | 55.0 | 0.0 | 4.0 | 128.0 | 205.0 | 0.0 | 1.0 | 130.0 | 1.0 | 2.0 | 2.0 | 1.0 | 7.0 | 3 |
| **283** | 35.0 | 1.0 | 2.0 | 122.0 | 192.0 | 0.0 | 0.0 | 174.0 | 0.0 | 0.0 | 1.0 | 0.0 | 3.0 | 0 |
| **284** | 61.0 | 1.0 | 4.0 | 148.0 | 203.0 | 0.0 | 0.0 | 161.0 | 0.0 | 0.0 | 1.0 | 1.0 | 7.0 | 2 |
| **285** | 58.0 | 1.0 | 4.0 | 114.0 | 318.0 | 0.0 | 1.0 | 140.0 | 0.0 | 4.4 | 3.0 | 3.0 | 6.0 | 4 |
| **286** | 58.0 | 0.0 | 4.0 | 170.0 | 225.0 | 1.0 | 2.0 | 146.0 | 1.0 | 2.8 | 2.0 | 2.0 | 6.0 | 2 |
| **287** | 58.0 | 1.0 | 2.0 | 125.0 | 220.0 | 0.0 | 0.0 | 144.0 | 0.0 | 0.4 | 2.0 | ? | 7.0 | 0 |
| **288** | 56.0 | 1.0 | 2.0 | 130.0 | 221.0 | 0.0 | 2.0 | 163.0 | 0.0 | 0.0 | 1.0 | 0.0 | 7.0 | 0 |
| **289** | 56.0 | 1.0 | 2.0 | 120.0 | 240.0 | 0.0 | 0.0 | 169.0 | 0.0 | 0.0 | 3.0 | 0.0 | 3.0 | 0 |
| **290** | 67.0 | 1.0 | 3.0 | 152.0 | 212.0 | 0.0 | 2.0 | 150.0 | 0.0 | 0.8 | 2.0 | 0.0 | 7.0 | 1 |
| **291** | 55.0 | 0.0 | 2.0 | 132.0 | 342.0 | 0.0 | 0.0 | 166.0 | 0.0 | 1.2 | 1.0 | 0.0 | 3.0 | 0 |
| **292** | 44.0 | 1.0 | 4.0 | 120.0 | 169.0 | 0.0 | 0.0 | 144.0 | 1.0 | 2.8 | 3.0 | 0.0 | 6.0 | 2 |
| **293** | 63.0 | 1.0 | 4.0 | 140.0 | 187.0 | 0.0 | 2.0 | 144.0 | 1.0 | 4.0 | 1.0 | 2.0 | 7.0 | 2 |
| **294** | 63.0 | 0.0 | 4.0 | 124.0 | 197.0 | 0.0 | 0.0 | 136.0 | 1.0 | 0.0 | 2.0 | 0.0 | 3.0 | 1 |
| **295** | 41.0 | 1.0 | 2.0 | 120.0 | 157.0 | 0.0 | 0.0 | 182.0 | 0.0 | 0.0 | 1.0 | 0.0 | 3.0 | 0 |
| **296** | 59.0 | 1.0 | 4.0 | 164.0 | 176.0 | 1.0 | 2.0 | 90.0 | 0.0 | 1.0 | 2.0 | 2.0 | 6.0 | 3 |
| **297** | 57.0 | 0.0 | 4.0 | 140.0 | 241.0 | 0.0 | 0.0 | 123.0 | 1.0 | 0.2 | 2.0 | 0.0 | 7.0 | 1 |
| **298** | 45.0 | 1.0 | 1.0 | 110.0 | 264.0 | 0.0 | 0.0 | 132.0 | 0.0 | 1.2 | 2.0 | 0.0 | 7.0 | 1 |
| **299** | 68.0 | 1.0 | 4.0 | 144.0 | 193.0 | 1.0 | 0.0 | 141.0 | 0.0 | 3.4 | 2.0 | 2.0 | 7.0 | 2 |
| **300** | 57.0 | 1.0 | 4.0 | 130.0 | 131.0 | 0.0 | 0.0 | 115.0 | 1.0 | 1.2 | 2.0 | 1.0 | 7.0 | 3 |
| **301** | 57.0 | 0.0 | 2.0 | 130.0 | 236.0 | 0.0 | 2.0 | 174.0 | 0.0 | 0.0 | 2.0 | 1.0 | 3.0 | 1 |
| **302** | 38.0 | 1.0 | 3.0 | 138.0 | 175.0 | 0.0 | 0.0 | 173.0 | 0.0 | 0.0 | 1.0 | ? | 3.0 | 0 |

In [26]:
```
# remove missing data (indicated with a "?")
data = cleveland[~cleveland.isin(['?'])]
data.loc[280:]
```

Out[26]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **280** | 57.0 | 1.0 | 4.0 | 110.0 | 335.0 | 0.0 | 0.0 | 143.0 | 1.0 | 3.0 | 2.0 | 1.0 | 7.0 | 2 |
| **281** | 47.0 | 1.0 | 3.0 | 130.0 | 253.0 | 0.0 | 0.0 | 179.0 | 0.0 | 0.0 | 1.0 | 0.0 | 3.0 | 0 |
| **282** | 55.0 | 0.0 | 4.0 | 128.0 | 205.0 | 0.0 | 1.0 | 130.0 | 1.0 | 2.0 | 2.0 | 1.0 | 7.0 | 3 |
| **283** | 35.0 | 1.0 | 2.0 | 122.0 | 192.0 | 0.0 | 0.0 | 174.0 | 0.0 | 0.0 | 1.0 | 0.0 | 3.0 | 0 |
| **284** | 61.0 | 1.0 | 4.0 | 148.0 | 203.0 | 0.0 | 0.0 | 161.0 | 0.0 | 0.0 | 1.0 | 1.0 | 7.0 | 2 |
| **285** | 58.0 | 1.0 | 4.0 | 114.0 | 318.0 | 0.0 | 1.0 | 140.0 | 0.0 | 4.4 | 3.0 | 3.0 | 6.0 | 4 |
| **286** | 58.0 | 0.0 | 4.0 | 170.0 | 225.0 | 1.0 | 2.0 | 146.0 | 1.0 | 2.8 | 2.0 | 2.0 | 6.0 | 2 |
| **287** | 58.0 | 1.0 | 2.0 | 125.0 | 220.0 | 0.0 | 0.0 | 144.0 | 0.0 | 0.4 | 2.0 | NaN | 7.0 | 0 |
| **288** | 56.0 | 1.0 | 2.0 | 130.0 | 221.0 | 0.0 | 2.0 | 163.0 | 0.0 | 0.0 | 1.0 | 0.0 | 7.0 | 0 |
| **289** | 56.0 | 1.0 | 2.0 | 120.0 | 240.0 | 0.0 | 0.0 | 169.0 | 0.0 | 0.0 | 3.0 | 0.0 | 3.0 | 0 |
| **290** | 67.0 | 1.0 | 3.0 | 152.0 | 212.0 | 0.0 | 2.0 | 150.0 | 0.0 | 0.8 | 2.0 | 0.0 | 7.0 | 1 |
| **291** | 55.0 | 0.0 | 2.0 | 132.0 | 342.0 | 0.0 | 0.0 | 166.0 | 0.0 | 1.2 | 1.0 | 0.0 | 3.0 | 0 |
| **292** | 44.0 | 1.0 | 4.0 | 120.0 | 169.0 | 0.0 | 0.0 | 144.0 | 1.0 | 2.8 | 3.0 | 0.0 | 6.0 | 2 |
| **293** | 63.0 | 1.0 | 4.0 | 140.0 | 187.0 | 0.0 | 2.0 | 144.0 | 1.0 | 4.0 | 1.0 | 2.0 | 7.0 | 2 |
| **294** | 63.0 | 0.0 | 4.0 | 124.0 | 197.0 | 0.0 | 0.0 | 136.0 | 1.0 | 0.0 | 2.0 | 0.0 | 3.0 | 1 |
| **295** | 41.0 | 1.0 | 2.0 | 120.0 | 157.0 | 0.0 | 0.0 | 182.0 | 0.0 | 0.0 | 1.0 | 0.0 | 3.0 | 0 |
| **296** | 59.0 | 1.0 | 4.0 | 164.0 | 176.0 | 1.0 | 2.0 | 90.0 | 0.0 | 1.0 | 2.0 | 2.0 | 6.0 | 3 |
| **297** | 57.0 | 0.0 | 4.0 | 140.0 | 241.0 | 0.0 | 0.0 | 123.0 | 1.0 | 0.2 | 2.0 | 0.0 | 7.0 | 1 |
| **298** | 45.0 | 1.0 | 1.0 | 110.0 | 264.0 | 0.0 | 0.0 | 132.0 | 0.0 | 1.2 | 2.0 | 0.0 | 7.0 | 1 |
| **299** | 68.0 | 1.0 | 4.0 | 144.0 | 193.0 | 1.0 | 0.0 | 141.0 | 0.0 | 3.4 | 2.0 | 2.0 | 7.0 | 2 |
| **300** | 57.0 | 1.0 | 4.0 | 130.0 | 131.0 | 0.0 | 0.0 | 115.0 | 1.0 | 1.2 | 2.0 | 1.0 | 7.0 | 3 |
| **301** | 57.0 | 0.0 | 2.0 | 130.0 | 236.0 | 0.0 | 2.0 | 174.0 | 0.0 | 0.0 | 2.0 | 1.0 | 3.0 | 1 |
| **302** | 38.0 | 1.0 | 3.0 | 138.0 | 175.0 | 0.0 | 0.0 | 173.0 | 0.0 | 0.0 | 1.0 | NaN | 3.0 | 0 |

In [27]:
```
# drop rows with NaN values from DataFrame
data = data.dropna(axis=0)
data.loc[280:]
```

Out[27]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **280** | 57.0 | 1.0 | 4.0 | 110.0 | 335.0 | 0.0 | 0.0 | 143.0 | 1.0 | 3.0 | 2.0 | 1.0 | 7.0 | 2 |
| **281** | 47.0 | 1.0 | 3.0 | 130.0 | 253.0 | 0.0 | 0.0 | 179.0 | 0.0 | 0.0 | 1.0 | 0.0 | 3.0 | 0 |
| **282** | 55.0 | 0.0 | 4.0 | 128.0 | 205.0 | 0.0 | 1.0 | 130.0 | 1.0 | 2.0 | 2.0 | 1.0 | 7.0 | 3 |
| **283** | 35.0 | 1.0 | 2.0 | 122.0 | 192.0 | 0.0 | 0.0 | 174.0 | 0.0 | 0.0 | 1.0 | 0.0 | 3.0 | 0 |
| **284** | 61.0 | 1.0 | 4.0 | 148.0 | 203.0 | 0.0 | 0.0 | 161.0 | 0.0 | 0.0 | 1.0 | 1.0 | 7.0 | 2 |
| **285** | 58.0 | 1.0 | 4.0 | 114.0 | 318.0 | 0.0 | 1.0 | 140.0 | 0.0 | 4.4 | 3.0 | 3.0 | 6.0 | 4 |
| **286** | 58.0 | 0.0 | 4.0 | 170.0 | 225.0 | 1.0 | 2.0 | 146.0 | 1.0 | 2.8 | 2.0 | 2.0 | 6.0 | 2 |
| **288** | 56.0 | 1.0 | 2.0 | 130.0 | 221.0 | 0.0 | 2.0 | 163.0 | 0.0 | 0.0 | 1.0 | 0.0 | 7.0 | 0 |
| **289** | 56.0 | 1.0 | 2.0 | 120.0 | 240.0 | 0.0 | 0.0 | 169.0 | 0.0 | 0.0 | 3.0 | 0.0 | 3.0 | 0 |
| **290** | 67.0 | 1.0 | 3.0 | 152.0 | 212.0 | 0.0 | 2.0 | 150.0 | 0.0 | 0.8 | 2.0 | 0.0 | 7.0 | 1 |
| **291** | 55.0 | 0.0 | 2.0 | 132.0 | 342.0 | 0.0 | 0.0 | 166.0 | 0.0 | 1.2 | 1.0 | 0.0 | 3.0 | 0 |
| **292** | 44.0 | 1.0 | 4.0 | 120.0 | 169.0 | 0.0 | 0.0 | 144.0 | 1.0 | 2.8 | 3.0 | 0.0 | 6.0 | 2 |
| **293** | 63.0 | 1.0 | 4.0 | 140.0 | 187.0 | 0.0 | 2.0 | 144.0 | 1.0 | 4.0 | 1.0 | 2.0 | 7.0 | 2 |
| **294** | 63.0 | 0.0 | 4.0 | 124.0 | 197.0 | 0.0 | 0.0 | 136.0 | 1.0 | 0.0 | 2.0 | 0.0 | 3.0 | 1 |
| **295** | 41.0 | 1.0 | 2.0 | 120.0 | 157.0 | 0.0 | 0.0 | 182.0 | 0.0 | 0.0 | 1.0 | 0.0 | 3.0 | 0 |
| **296** | 59.0 | 1.0 | 4.0 | 164.0 | 176.0 | 1.0 | 2.0 | 90.0 | 0.0 | 1.0 | 2.0 | 2.0 | 6.0 | 3 |
| **297** | 57.0 | 0.0 | 4.0 | 140.0 | 241.0 | 0.0 | 0.0 | 123.0 | 1.0 | 0.2 | 2.0 | 0.0 | 7.0 | 1 |
| **298** | 45.0 | 1.0 | 1.0 | 110.0 | 264.0 | 0.0 | 0.0 | 132.0 | 0.0 | 1.2 | 2.0 | 0.0 | 7.0 | 1 |
| **299** | 68.0 | 1.0 | 4.0 | 144.0 | 193.0 | 1.0 | 0.0 | 141.0 | 0.0 | 3.4 | 2.0 | 2.0 | 7.0 | 2 |
| **300** | 57.0 | 1.0 | 4.0 | 130.0 | 131.0 | 0.0 | 0.0 | 115.0 | 1.0 | 1.2 | 2.0 | 1.0 | 7.0 | 3 |
| **301** | 57.0 | 0.0 | 2.0 | 130.0 | 236.0 | 0.0 | 2.0 | 174.0 | 0.0 | 0.0 | 2.0 | 1.0 | 3.0 | 1 |

```
In [28]: # print the shape and data type of the dataframe
         print data.shape
         print data.dtypes

(297, 14)
age           float64
sex           float64
cp            float64
trestbps      float64
chol          float64
fbs           float64
restecg       float64
thalach       float64
exang         float64
oldpeak       float64
slope         float64
ca             object
thal           object
class           int64
dtype: object
```

```
In [29]: # transform data to numeric to enable further analysis
         data = data.apply(pd.to_numeric)
         data.dtypes

Out[29]: age           float64
         sex           float64
         cp            float64
         trestbps      float64
         chol          float64
         fbs           float64
         restecg       float64
         thalach       float64
         exang         float64
         oldpeak       float64
         slope         float64
         ca            float64
         thal          float64
         class           int64
         dtype: object
```

In [30]:

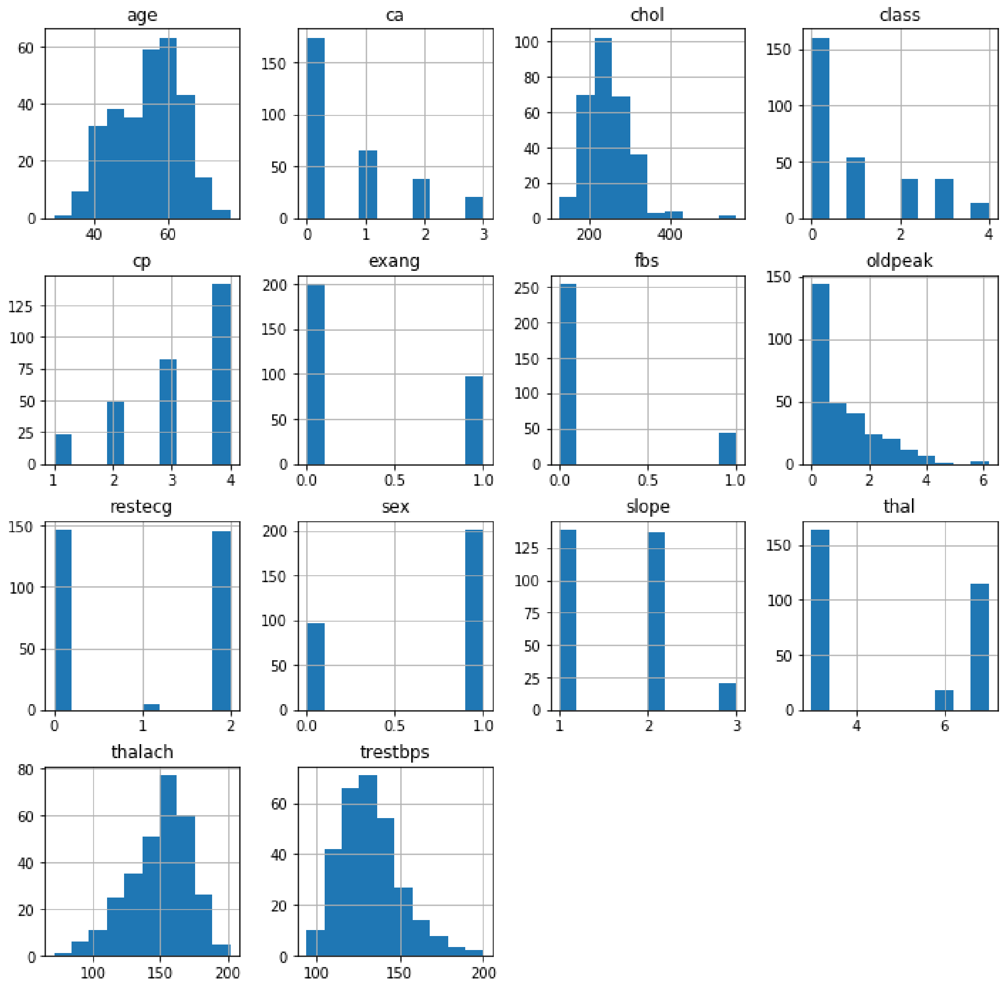
```
# print data characteristics, usings pandas built-in describe() function
data.describe()
```

Out[30]:

| age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 297.000000 | 297.000000 | 297.000000 | 297.000000 | 297.000000 | 297.000000 | 297.000000 | 297.000000 | 297.000000 | 297.000000 | 297.000000 | 297.000000 | 297.000000 |
| 54.542088 | 0.676768 | 3.158249 | 131.693603 | 247.350168 | 0.144781 | 0.996633 | 149.599327 | 0.326599 | 1.055556 | 1.602694 | 0.676768 | 4.730640 |
| 9.049736 | 0.468500 | 0.964859 | 17.762806 | 51.997583 | 0.352474 | 0.994914 | 22.941562 | 0.469761 | 1.166123 | 0.618187 | 0.938965 | 1.938629 |
| 29.000000 | 0.000000 | 1.000000 | 94.000000 | 126.000000 | 0.000000 | 0.000000 | 71.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 3.000000 |
| 48.000000 | 0.000000 | 3.000000 | 120.000000 | 211.000000 | 0.000000 | 0.000000 | 133.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 3.000000 |
| 56.000000 | 1.000000 | 3.000000 | 130.000000 | 243.000000 | 0.000000 | 1.000000 | 153.000000 | 0.000000 | 0.800000 | 2.000000 | 0.000000 | 3.000000 |
| 61.000000 | 1.000000 | 4.000000 | 140.000000 | 276.000000 | 0.000000 | 2.000000 | 166.000000 | 1.000000 | 1.600000 | 2.000000 | 1.000000 | 7.000000 |
| 77.000000 | 1.000000 | 4.000000 | 200.000000 | 564.000000 | 1.000000 | 2.000000 | 202.000000 | 1.000000 | 6.200000 | 3.000000 | 3.000000 | 7.000000 |

In [31]:

```
# plot histograms for each variable
data.hist(figsize = (12, 12))
plt.show()
```



In [33]:

```
# convert the data to categorical labels
from keras.utils.np_utils import to_categorical

Y_train = to_categorical(y_train, num_classes=None)
Y_test = to_categorical(y_test, num_classes=None)
print Y_train.shape
print Y_train[:10]
```

```
(237L, 5L)
[[0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 1. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [1. 0. 0. 0. 0.]]
```

In [34]:

```python
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam

# define a function to build the keras model
def create_model():
    # create model
    model = Sequential()
    model.add(Dense(8, input_dim=13, kernel_initializer='normal', activation='relu'))
    model.add(Dense(4, kernel_initializer='normal', activation='relu'))
    model.add(Dense(5, activation='softmax'))

    # compile model
    adam = Adam(lr=0.001)
    model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
    return model

model = create_model()

print(model.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 8)                 112
_________________________________________________________________
dense_8 (Dense)              (None, 4)                 36
_________________________________________________________________
dense_9 (Dense)              (None, 5)                 25
=================================================================
Total params: 173
Trainable params: 173
Non-trainable params: 0
_________________________________________________________________
None
```

In [35]:

```python
# fit the model to the training data
model.fit(X_train, Y_train, epochs=100, batch_size=10, verbose = 1)
```

```
Epoch 1/100
237/237 [==============================] - 0s 207us/step - loss: 1.6193 - acc: 0.2911
Epoch 2/100
237/237 [==============================] - 0s 194us/step - loss: 1.5907 - acc: 0.5190
Epoch 3/100
237/237 [==============================] - 0s 287us/step - loss: 1.5769 - acc: 0.5190
Epoch 4/100
237/237 [==============================] - 0s 148us/step - loss: 1.5636 - acc: 0.5232
Epoch 5/100
237/237 [==============================] - 0s 152us/step - loss: 1.5498 - acc: 0.5232
Epoch 6/100
237/237 [==============================] - 0s 148us/step - loss: 1.5360 - acc: 0.5232
Epoch 7/100
237/237 [==============================] - 0s 122us/step - loss: 1.5216 - acc: 0.5232
Epoch 8/100
237/237 [==============================] - 0s 169us/step - loss: 1.5075 - acc: 0.5232
Epoch 9/100
237/237 [==============================] - 0s 135us/step - loss: 1.4930 - acc: 0.5232
Epoch 10/100
237/237 [==============================] - 0s 122us/step - loss: 1.4795 - acc: 0.5232
Epoch 11/100
237/237 [==============================] - 0s 143us/step - loss: 1.4659 - acc: 0.5232
Epoch 12/100
237/237 [==============================] - 0s 143us/step - loss: 1.4525 - acc: 0.5232
Epoch 13/100
237/237 [==============================] - 0s 169us/step - loss: 1.4401 - acc: 0.5232
Epoch 14/100
237/237 [==============================] - 0s 127us/step - loss: 1.4273 - acc: 0.5232
Epoch 15/100
237/237 [==============================] - 0s 135us/step - loss: 1.4158 - acc: 0.5232
Epoch 16/100
237/237 [==============================] - 0s 131us/step - loss: 1.4042 - acc: 0.5232
Epoch 17/100
237/237 [==============================] - 0s 131us/step - loss: 1.3744 - acc: 0.5232
Epoch 18/100
237/237 [==============================] - 0s 131us/step - loss: 1.3500 - acc: 0.5232
Epoch 19/100
237/237 [==============================] - 0s 131us/step - loss: 1.3297 - acc: 0.5232
Epoch 20/100
237/237 [==============================] - 0s 118us/step - loss: 1.3110 - acc: 0.5232
Epoch 21/100
237/237 [==============================] - 0s 127us/step - loss: 1.2928 - acc: 0.5232
Epoch 22/100
237/237 [==============================] - 0s 139us/step - loss: 1.2766 - acc: 0.5232
Epoch 23/100
237/237 [==============================] - 0s 143us/step - loss: 1.2575 - acc: 0.5232
Epoch 24/100
237/237 [==============================] - 0s 127us/step - loss: 1.2679 - acc: 0.5232
Epoch 25/100
237/237 [==============================] - 0s 122us/step - loss: 1.2401 - acc: 0.5232
Epoch 26/100
237/237 [==============================] - 0s 127us/step - loss: 1.2307 - acc: 0.5232
Epoch 27/100
237/237 [==============================] - 0s 152us/step - loss: 1.2155 - acc: 0.5232
Epoch 28/100
237/237 [==============================] - 0s 177us/step - loss: 1.2265 - acc: 0.5232
Epoch 29/100
237/237 [==============================] - 0s 148us/step - loss: 1.2047 - acc: 0.5232
Epoch 30/100
237/237 [==============================] - 0s 148us/step - loss: 1.1927 - acc: 0.5232
Epoch 31/100
237/237 [==============================] - 0s 156us/step - loss: 1.1792 - acc: 0.5232
Epoch 32/100
237/237 [==============================] - 0s 156us/step - loss: 1.1733 - acc: 0.5232
Epoch 33/100
237/237 [==============================] - 0s 139us/step - loss: 1.1600 - acc: 0.5232
Epoch 34/100
237/237 [==============================] - 0s 143us/step - loss: 1.1714 - acc: 0.5232
Epoch 35/100
237/237 [==============================] - 0s 127us/step - loss: 1.1578 - acc: 0.5232
Epoch 36/100
237/237 [==============================] - 0s 135us/step - loss: 1.1525 - acc: 0.5232
Epoch 37/100
237/237 [==============================] - 0s 131us/step - loss: 1.1510 - acc: 0.5232
Epoch 38/100
237/237 [==============================] - 0s 148us/step - loss: 1.1371 - acc: 0.5232
Epoch 39/100
237/237 [==============================] - 0s 143us/step - loss: 1.1405 - acc: 0.5232
Epoch 40/100
237/237 [==============================] - 0s 135us/step - loss: 1.1256 - acc: 0.5232
Epoch 41/100
237/237 [==============================] - 0s 148us/step - loss: 1.1302 - acc: 0.5232
Epoch 42/100
237/237 [==============================] - 0s 135us/step - loss: 1.1241 - acc: 0.5232
Epoch 43/100
237/237 [==============================] - 0s 127us/step - loss: 1.1065 - acc: 0.5232
Epoch 44/100
237/237 [==============================] - 0s 148us/step - loss: 1.1146 - acc: 0.5232
Epoch 45/100
237/237 [==============================] - 0s 148us/step - loss: 1.1051 - acc: 0.5232
Epoch 46/100
237/237 [==============================] - 0s 135us/step - loss: 1.0943 - acc: 0.5232
Epoch 47/100
237/237 [==============================] - 0s 118us/step - loss: 1.1030 - acc: 0.5232
Epoch 48/100
237/237 [==============================] - 0s 127us/step - loss: 1.0912 - acc: 0.5232
Epoch 49/100
237/237 [==============================] - 0s 131us/step - loss: 1.0771 - acc: 0.5232
Epoch 50/100
237/237 [==============================] - 0s 127us/step - loss: 1.0775 - acc: 0.5232
Epoch 51/100
237/237 [==============================] - 0s 110us/step - loss: 1.0839 - acc: 0.5232
Epoch 52/100
237/237 [==============================] - 0s 135us/step - loss: 1.0798 - acc: 0.5232
Epoch 53/100
237/237 [==============================] - 0s 127us/step - loss: 1.0951 - acc: 0.5232
Epoch 54/100
237/237 [==============================] - 0s 139us/step - loss: 1.0738 - acc: 0.5443
Epoch 55/100
237/237 [==============================] - 0s 118us/step - loss: 1.0589 - acc: 0.5612
Epoch 56/100
237/237 [==============================] - 0s 131us/step - loss: 1.0499 - acc: 0.5443
Epoch 57/100
237/237 [==============================] - 0s 127us/step - loss: 1.0418 - acc: 0.5485
Epoch 58/100
237/237 [==============================] - 0s 131us/step - loss: 1.0461 - acc: 0.5485
Epoch 59/100
237/237 [==============================] - 0s 122us/step - loss: 1.0387 - acc: 0.5654
Epoch 60/100
237/237 [==============================] - 0s 148us/step - loss: 1.0329 - acc: 0.5654
Epoch 61/100
237/237 [==============================] - 0s 131us/step - loss: 1.0409 - acc: 0.5570
Epoch 62/100
237/237 [==============================] - 0s 131us/step - loss: 1.0312 - acc: 0.5654
Epoch 63/100
237/237 [==============================] - 0s 131us/step - loss: 1.0231 - acc: 0.5654
Epoch 64/100
237/237 [==============================] - 0s 143us/step - loss: 1.0203 - acc: 0.5612
Epoch 65/100
237/237 [==============================] - 0s 122us/step - loss: 1.0137 - acc: 0.5612
Epoch 66/100
237/237 [==============================] - 0s 118us/step - loss: 1.0165 - acc: 0.5527
Epoch 67/100
237/237 [==============================] - 0s 131us/step - loss: 1.0076 - acc: 0.5612
Epoch 68/100
237/237 [==============================] - 0s 114us/step - loss: 1.0124 - acc: 0.5612
Epoch 69/100
237/237 [==============================] - 0s 127us/step - loss: 1.0116 - acc: 0.5696
Epoch 70/100
237/237 [==============================] - 0s 122us/step - loss: 1.0068 - acc: 0.5570
Epoch 71/100
237/237 [==============================] - 0s 143us/step - loss: 1.0017 - acc: 0.5696
Epoch 72/100
237/237 [==============================] - 0s 127us/step - loss: 0.9954 - acc: 0.5696
Epoch 73/100
237/237 [==============================] - 0s 110us/step - loss: 1.0066 - acc: 0.5612
Epoch 74/100
237/237 [==============================] - 0s 122us/step - loss: 0.9907 - acc: 0.5654
Epoch 75/100
237/237 [==============================] - 0s 122us/step - loss: 0.9897 - acc: 0.5612
Epoch 76/100
237/237 [==============================] - 0s 118us/step - loss: 0.9926 - acc: 0.5612
Epoch 77/100
237/237 [==============================] - 0s 118us/step - loss: 0.9854 - acc: 0.5654
Epoch 78/100
237/237 [==============================] - 0s 131us/step - loss: 0.9770 - acc: 0.5738
Epoch 79/100
237/237 [==============================] - 0s 127us/step - loss: 0.9727 - acc: 0.5696
Epoch 80/100
237/237 [==============================] - 0s 135us/step - loss: 0.9837 - acc: 0.5781
Epoch 81/100
237/237 [==============================] - 0s 122us/step - loss: 0.9762 - acc: 0.5696
Epoch 82/100
237/237 [==============================] - 0s 135us/step - loss: 0.9671 - acc: 0.5654
Epoch 83/100
237/237 [==============================] - 0s 118us/step - loss: 0.9734 - acc: 0.5612
Epoch 84/100
237/237 [==============================] - 0s 122us/step - loss: 0.9641 - acc: 0.5696
Epoch 85/100
237/237 [==============================] - 0s 105us/step - loss: 0.9616 - acc: 0.5612
Epoch 86/100
237/237 [==============================] - 0s 122us/step - loss: 0.9616 - acc: 0.5781
Epoch 87/100
237/237 [==============================] - 0s 135us/step - loss: 0.9582 - acc: 0.5696
Epoch 88/100
237/237 [==============================] - 0s 122us/step - loss: 0.9552 - acc: 0.5696
Epoch 89/100
237/237 [==============================] - 0s 118us/step - loss: 0.9630 - acc: 0.5907
Epoch 90/100
237/237 [==============================] - 0s 122us/step - loss: 0.9506 - acc: 0.6076
Epoch 91/100
237/237 [==============================] - 0s 118us/step - loss: 0.9559 - acc: 0.6287
Epoch 92/100
237/237 [==============================] - 0s 127us/step - loss: 0.9597 - acc: 0.6203
Epoch 93/100
237/237 [==============================] - 0s 127us/step - loss: 0.9535 - acc: 0.6245
Epoch 94/100
237/237 [==============================] - 0s 135us/step - loss: 0.9664 - acc: 0.5992
Epoch 95/100
237/237 [==============================] - 0s 139us/step - loss: 0.9460 - acc: 0.6076
Epoch 96/100
237/237 [==============================] - 0s 148us/step - loss: 0.9559 - acc: 0.6287
Epoch 97/100
237/237 [==============================] - 0s 135us/step - loss: 0.9387 - acc: 0.6160
Epoch 98/100
237/237 [==============================] - 0s 114us/step - loss: 0.9416 - acc: 0.6245
Epoch 99/100
237/237 [==============================] - 0s 135us/step - loss: 0.9375 - acc: 0.6160
Epoch 100/100
237/237 [==============================] - 0s 122us/step - loss: 0.9383 - acc: 0.6160

<keras.callbacks.History at 0x17ac65c0>
```

In [36]:
```python
# convert into binary classification problem - heart disease or no heart disease
Y_train_binary = y_train.copy()
Y_test_binary = y_test.copy()

Y_train_binary[Y_train_binary > 0] = 1
Y_test_binary[Y_test_binary > 0] = 1

print Y_train_binary[:20]
```

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

In [37]:
```python
# define a new keras model for binary classification
def create_binary_model():
    # create model
    model = Sequential()
    model.add(Dense(8, input_dim=13, kernel_initializer='normal', activation='relu'))
    model.add(Dense(4, kernel_initializer='normal', activation='relu'))
    model.add(Dense(1, activation='sigmoid'))

    # Compile model
    adam = Adam(lr=0.001)
    model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'])
    return model

binary_model = create_binary_model()

print(binary_model.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_10 (Dense)             (None, 8)                 112
_________________________________________________________________
dense_11 (Dense)             (None, 4)                 36
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 5
=================================================================
Total params: 153
Trainable params: 153
Non-trainable params: 0
_________________________________________________________________
None
```

In [38]:
```
# fit the binary model on the training data
binary_model.fit(X_train, Y_train_binary, epochs=100, batch_size=10, verbose = 1)
```

```
Epoch 1/100
237/237 [==============================] - 0s 460us/step - loss: 0.7973 - acc: 0.4979
Epoch 2/100
237/237 [==============================] - 0s 557us/step - loss: 0.6648 - acc: 0.6203
Epoch 3/100
237/237 [==============================] - 0s 549us/step - loss: 0.6543 - acc: 0.6118
Epoch 4/100
237/237 [==============================] - 0s 599us/step - loss: 0.6367 - acc: 0.6878
Epoch 5/100
237/237 [==============================] - 0s 675us/step - loss: 0.6313 - acc: 0.6624
Epoch 6/100
237/237 [==============================] - 0s 633us/step - loss: 0.6231 - acc: 0.6835
Epoch 7/100
237/237 [==============================] - 0s 443us/step - loss: 0.6170 - acc: 0.6540
Epoch 8/100
237/237 [==============================] - 0s 549us/step - loss: 0.6207 - acc: 0.6667
Epoch 9/100
237/237 [==============================] - 0s 684us/step - loss: 0.5848 - acc: 0.7257
Epoch 10/100
237/237 [==============================] - 0s 612us/step - loss: 0.5958 - acc: 0.6835
Epoch 11/100
237/237 [==============================] - 0s 700us/step - loss: 0.5761 - acc: 0.7089
Epoch 12/100
237/237 [==============================] - 0s 570us/step - loss: 0.5664 - acc: 0.7215
Epoch 13/100
237/237 [==============================] - 0s 616us/step - loss: 0.5537 - acc: 0.7426
Epoch 14/100
237/237 [==============================] - 0s 654us/step - loss: 0.5439 - acc: 0.7342
Epoch 15/100
237/237 [==============================] - 0s 553us/step - loss: 0.5492 - acc: 0.7426
Epoch 16/100
237/237 [==============================] - 0s 549us/step - loss: 0.5340 - acc: 0.7553
Epoch 17/100
237/237 [==============================] - 0s 456us/step - loss: 0.5244 - acc: 0.7426
Epoch 18/100
237/237 [==============================] - 0s 586us/step - loss: 0.5155 - acc: 0.7468
Epoch 19/100
237/237 [==============================] - 0s 578us/step - loss: 0.5069 - acc: 0.7764
Epoch 20/100
237/237 [==============================] - 0s 574us/step - loss: 0.5043 - acc: 0.7384
Epoch 21/100
237/237 [==============================] - 0s 574us/step - loss: 0.4952 - acc: 0.7848
Epoch 22/100
237/237 [==============================] - 0s 582us/step - loss: 0.4962 - acc: 0.7511
Epoch 23/100
237/237 [==============================] - 0s 540us/step - loss: 0.4840 - acc: 0.7595
Epoch 24/100
237/237 [==============================] - 0s 679us/step - loss: 0.5205 - acc: 0.7300
Epoch 25/100
237/237 [==============================] - 0s 418us/step - loss: 0.4855 - acc: 0.7722
Epoch 26/100
237/237 [==============================] - 0s 384us/step - loss: 0.4735 - acc: 0.7764
Epoch 27/100
237/237 [==============================] - 0s 494us/step - loss: 0.4619 - acc: 0.7975
Epoch 28/100
237/237 [==============================] - 0s 367us/step - loss: 0.4504 - acc: 0.8017
Epoch 29/100
237/237 [==============================] - 0s 316us/step - loss: 0.4520 - acc: 0.7975
Epoch 30/100
237/237 [==============================] - 0s 338us/step - loss: 0.4446 - acc: 0.8228
Epoch 31/100
237/237 [==============================] - 0s 312us/step - loss: 0.4422 - acc: 0.8059
Epoch 32/100
237/237 [==============================] - 0s 333us/step - loss: 0.4353 - acc: 0.8101
Epoch 33/100
237/237 [==============================] - 0s 354us/step - loss: 0.4240 - acc: 0.8059
Epoch 34/100
237/237 [==============================] - 0s 338us/step - loss: 0.4628 - acc: 0.7806
Epoch 35/100
237/237 [==============================] - 0s 376us/step - loss: 0.4130 - acc: 0.8143
Epoch 36/100
237/237 [==============================] - 0s 312us/step - loss: 0.4077 - acc: 0.8186
Epoch 37/100
237/237 [==============================] - 0s 304us/step - loss: 0.4256 - acc: 0.8101
Epoch 38/100
237/237 [==============================] - 0s 333us/step - loss: 0.4041 - acc: 0.8270
Epoch 39/100
237/237 [==============================] - 0s 312us/step - loss: 0.4030 - acc: 0.8439
Epoch 40/100
237/237 [==============================] - 0s 295us/step - loss: 0.3976 - acc: 0.8397
Epoch 41/100
237/237 [==============================] - 0s 304us/step - loss: 0.3996 - acc: 0.8270
Epoch 42/100
237/237 [==============================] - 0s 300us/step - loss: 0.4314 - acc: 0.7932
Epoch 43/100
237/237 [==============================] - 0s 291us/step - loss: 0.3902 - acc: 0.8439
Epoch 44/100
237/237 [==============================] - 0s 287us/step - loss: 0.3980 - acc: 0.8481
Epoch 45/100
237/237 [==============================] - 0s 346us/step - loss: 0.3850 - acc: 0.8312
Epoch 46/100
237/237 [==============================] - 0s 291us/step - loss: 0.3873 - acc: 0.8565
Epoch 47/100
237/237 [==============================] - 0s 228us/step - loss: 0.3728 - acc: 0.8565
Epoch 48/100
237/237 [==============================] - 0s 232us/step - loss: 0.4108 - acc: 0.8312
Epoch 49/100
237/237 [==============================] - 0s 295us/step - loss: 0.3728 - acc: 0.8650
Epoch 50/100
237/237 [==============================] - 0s 232us/step - loss: 0.3721 - acc: 0.8565
Epoch 51/100
237/237 [==============================] - 0s 228us/step - loss: 0.3812 - acc: 0.8439
Epoch 52/100
237/237 [==============================] - 0s 228us/step - loss: 0.3880 - acc: 0.8312
Epoch 53/100
237/237 [==============================] - 0s 232us/step - loss: 0.3676 - acc: 0.8692
Epoch 54/100
237/237 [==============================] - 0s 207us/step - loss: 0.3716 - acc: 0.8565
Epoch 55/100
237/237 [==============================] - 0s 232us/step - loss: 0.3591 - acc: 0.8734
Epoch 56/100
237/237 [==============================] - 0s 215us/step - loss: 0.3625 - acc: 0.8565
Epoch 57/100
237/237 [==============================] - 0s 232us/step - loss: 0.3557 - acc: 0.8692
Epoch 58/100
237/237 [==============================] - 0s 236us/step - loss: 0.3604 - acc: 0.8861
Epoch 59/100
237/237 [==============================] - 0s 236us/step - loss: 0.3599 - acc: 0.8692
Epoch 60/100
237/237 [==============================] - 0s 287us/step - loss: 0.3513 - acc: 0.8776
Epoch 61/100
237/237 [==============================] - 0s 367us/step - loss: 0.3853 - acc: 0.8650
Epoch 62/100
237/237 [==============================] - 0s 283us/step - loss: 0.3820 - acc: 0.8439
Epoch 63/100
237/237 [==============================] - 0s 215us/step - loss: 0.4204 - acc: 0.8397
Epoch 64/100
237/237 [==============================] - 0s 312us/step - loss: 0.3694 - acc: 0.8523
Epoch 65/100
237/237 [==============================] - 0s 236us/step - loss: 0.3592 - acc: 0.8692
Epoch 66/100
237/237 [==============================] - 0s 219us/step - loss: 0.3523 - acc: 0.8692
Epoch 67/100
237/237 [==============================] - 0s 291us/step - loss: 0.3566 - acc: 0.8692
Epoch 68/100
237/237 [==============================] - 0s 211us/step - loss: 0.3705 - acc: 0.8270
Epoch 69/100
237/237 [==============================] - 0s 241us/step - loss: 0.3562 - acc: 0.8608
Epoch 70/100
237/237 [==============================] - 0s 203us/step - loss: 0.3765 - acc: 0.8692
Epoch 71/100
237/237 [==============================] - 0s 219us/step - loss: 0.3564 - acc: 0.8650
Epoch 72/100
237/237 [==============================] - 0s 215us/step - loss: 0.3719 - acc: 0.8650
Epoch 73/100
237/237 [==============================] - 0s 198us/step - loss: 0.3559 - acc: 0.8565
Epoch 74/100
237/237 [==============================] - 0s 224us/step - loss: 0.3681 - acc: 0.8523
Epoch 75/100
237/237 [==============================] - 0s 190us/step - loss: 0.3533 - acc: 0.8608
Epoch 76/100
237/237 [==============================] - 0s 207us/step - loss: 0.3598 - acc: 0.8734
Epoch 77/100
237/237 [==============================] - 0s 194us/step - loss: 0.3506 - acc: 0.8650
Epoch 78/100
237/237 [==============================] - 0s 186us/step - loss: 0.4006 - acc: 0.8734
Epoch 79/100
237/237 [==============================] - 0s 203us/step - loss: 0.3521 - acc: 0.8565
Epoch 80/100
237/237 [==============================] - 0s 215us/step - loss: 0.3677 - acc: 0.8734
Epoch 81/100
237/237 [==============================] - 0s 274us/step - loss: 0.3466 - acc: 0.8565
Epoch 82/100
237/237 [==============================] - 0s 203us/step - loss: 0.4005 - acc: 0.8228
Epoch 83/100
237/237 [==============================] - 0s 219us/step - loss: 0.3642 - acc: 0.8692
Epoch 84/100
237/237 [==============================] - 0s 215us/step - loss: 0.3394 - acc: 0.8692
Epoch 85/100
237/237 [==============================] - 0s 203us/step - loss: 0.3484 - acc: 0.8734
Epoch 86/100
237/237 [==============================] - 0s 224us/step - loss: 0.3529 - acc: 0.8650
Epoch 87/100
237/237 [==============================] - 0s 207us/step - loss: 0.3528 - acc: 0.8650
Epoch 88/100
237/237 [==============================] - 0s 203us/step - loss: 0.3473 - acc: 0.8734
Epoch 89/100
237/237 [==============================] - 0s 190us/step - loss: 0.3509 - acc: 0.8565
Epoch 90/100
237/237 [==============================] - 0s 194us/step - loss: 0.3390 - acc: 0.8734
Epoch 91/100
237/237 [==============================] - 0s 207us/step - loss: 0.3598 - acc: 0.8481
Epoch 92/100
237/237 [==============================] - 0s 219us/step - loss: 0.3503 - acc: 0.8608
Epoch 93/100
237/237 [==============================] - 0s 190us/step - loss: 0.3492 - acc: 0.8608
Epoch 94/100
237/237 [==============================] - 0s 211us/step - loss: 0.3719 - acc: 0.8565
Epoch 95/100
237/237 [==============================] - 0s 207us/step - loss: 0.3495 - acc: 0.8650
Epoch 96/100
237/237 [==============================] - 0s 181us/step - loss: 0.3465 - acc: 0.8734
Epoch 97/100
237/237 [==============================] - 0s 203us/step - loss: 0.3582 - acc: 0.8608
Epoch 98/100
237/237 [==============================] - 0s 198us/step - loss: 0.3476 - acc: 0.8734
Epoch 99/100
237/237 [==============================] - 0s 295us/step - loss: 0.3432 - acc: 0.8523
Epoch 100/100
237/237 [==============================] - 0s 215us/step - loss: 0.3385 - acc: 0.8776

<keras.callbacks.History at 0x17200e48>
```

```
In [26]: # generate classification report using predictions for categorical model
         from sklearn.metrics import classification_report, accuracy_score

         categorical_pred = model.predict(X_test)
```

```
In [27]: categorical_pred
```

```
Out[27]: array([[1.01122111e-01, 2.55058527e-01, 2.35597149e-01, 3.12478006e-01,
                 9.57441553e-02],
                [5.33457756e-01, 2.98719645e-01, 9.35494676e-02, 5.82468845e-02,
                 1.60261374e-02],
                [5.80684125e-01, 2.97834665e-01, 7.01721087e-02, 4.20845188e-02,
                 9.22461320e-03],
                [1.16159856e-01, 2.39220947e-01, 2.43541703e-01, 2.89612323e-01,
                 1.11465104e-01],
                [1.20953463e-01, 2.34275043e-01, 2.45628655e-01, 2.82606691e-01,
                 1.16536178e-01],
                [1.21686250e-01, 2.33523130e-01, 2.45929852e-01, 2.81546861e-01,
                 1.17313892e-01],
                [6.76299691e-01, 2.52068818e-01, 4.43583280e-02, 2.27026902e-02,
                 4.57041990e-03],
                [1.95704728e-01, 3.10539395e-01, 2.08661154e-01, 2.19161719e-01,
                 6.59330785e-02],
                [3.60710382e-01, 3.46726894e-01, 1.44093186e-01, 1.18074283e-01,
                 3.03952657e-02],
                [1.15559876e-01, 2.39843398e-01, 2.43266031e-01, 2.90498316e-01,
                 1.10832416e-01],
                [8.28893661e-01, 1.49000451e-01, 1.55386953e-02, 5.51294256e-03,
                 1.05435983e-03],
                [1.06301658e-01, 2.49546438e-01, 2.38584325e-01, 3.04440856e-01,
                 1.01126775e-01],
                [4.37164307e-01, 3.19904685e-01, 1.26573503e-01, 9.00377557e-02,
                 2.63197385e-02],
                [6.83460057e-01, 2.45263875e-01, 4.43375707e-02, 2.22279448e-02,
                 4.71051177e-03],
                [6.97706223e-01, 2.37085029e-01, 4.10035960e-02, 2.00037956e-02,
                 4.20142151e-03],
                [6.39340281e-01, 2.71579951e-01, 5.36735281e-02, 2.93161590e-02,
                 6.09008828e-03],
                [3.31699371e-01, 3.50276917e-01, 1.53014556e-01, 1.31520733e-01,
                 3.34884711e-02],
                [1.06818460e-01, 2.48999819e-01, 2.38867462e-01, 3.03648621e-01,
                 1.01665713e-01],
                [6.94831192e-01, 2.40902245e-01, 4.03533690e-02, 1.99141204e-02,
                 3.99902463e-03],
                [8.40716958e-01, 1.41284823e-01, 1.28218718e-02, 4.40968201e-03,
                 7.66728190e-04],
                [9.30293977e-01, 6.61148280e-02, 2.85831373e-03, 6.40276470e-04,
                 9.25873755e-05],
                [8.31946492e-01, 1.46950290e-01, 1.48829427e-02, 5.23659727e-03,
                 9.83754406e-04],
                [1.70788392e-01, 3.06327462e-01, 2.14115247e-01, 2.39519149e-01,
                 6.92496821e-02],
                [9.06477571e-01, 8.70160535e-02, 5.00054751e-03, 1.30012271e-03,
                 2.05693970e-04],
                [1.13011487e-01, 2.71215528e-01, 2.30411619e-01, 2.98486114e-01,
                 8.68752897e-02],
                [8.86667788e-01, 1.04494877e-01, 6.63056364e-03, 1.91386475e-03,
                 2.92797282e-04],
                [8.74389350e-01, 1.14840925e-01, 7.96464831e-03, 2.42676493e-03,
                 3.78298369e-04],
                [1.06043883e-01, 2.49819279e-01, 2.38442108e-01, 3.04836661e-01,
                 1.00858085e-01],
                [7.40871191e-01, 2.12562516e-01, 3.03196758e-02, 1.36149665e-02,
                 2.63170991e-03],
                [7.86449611e-01, 1.81655303e-01, 2.15843264e-02, 8.69893841e-03,
                 1.61177712e-03],
                [8.90560150e-01, 1.00513063e-01, 6.71979412e-03, 1.89320825e-03,
                 3.13772325e-04],
                [9.03989911e-01, 8.90645757e-02, 5.31880464e-03, 1.40023627e-03,
                 2.26360804e-04],
                [8.57582331e-01, 1.28445357e-01, 1.01387231e-02, 3.29650776e-03,
                 5.37161250e-04],
                [7.43690312e-01, 2.12609112e-01, 2.85454392e-02, 1.28174732e-02,
                 2.33773072e-03],
                [5.92802703e-01, 2.91502774e-01, 6.73934817e-02, 3.95577326e-02,
                 8.74337275e-03],
                [1.08115964e-01, 2.47630060e-01, 2.39566654e-01, 3.01667005e-01,
                 1.03020266e-01],
                [8.70782733e-01, 1.17978081e-01, 8.27959739e-03, 2.56324536e-03,
                 3.96283926e-04],
                [7.59053648e-01, 1.98871464e-01, 2.77820975e-02, 1.19030857e-02,
                 2.38960725e-03],
                [8.44498575e-01, 1.34342074e-01, 1.50505928e-02, 4.99778474e-03,
                 1.11100485e-03],
                [6.71156108e-01, 2.53151149e-01, 4.66687866e-02, 2.40013525e-02,
                 5.02254814e-03],
                [5.33971429e-01, 3.26808959e-01, 7.78735280e-02, 5.12168817e-02,
                 1.01292143e-02],
                [8.70443881e-01, 1.17023557e-01, 9.20970365e-03, 2.82991957e-03,
                 4.92981286e-04],
                [7.69905508e-01, 1.92171559e-01, 2.52804980e-02, 1.05787218e-02,
                 2.06370209e-03],
                [8.30866456e-01, 1.48217812e-01, 1.47424173e-02, 5.21692727e-03,
                 9.56410193e-04],
                [9.11621511e-01, 8.25882554e-02, 4.48215613e-03, 1.13246811e-03,
                 1.75669353e-04],
                [6.37527823e-01, 2.50932038e-01, 6.66824207e-02, 3.46858911e-02,
                 1.01718791e-02],
                [1.02937825e-01, 2.53119171e-01, 2.36675709e-01, 3.09640139e-01,
                 9.76271704e-02],
                [1.56325504e-01, 2.95393705e-01, 2.20237121e-01, 2.52701730e-01,
                 7.53418878e-02],
                [6.46143198e-01, 2.41424024e-01, 6.74633533e-02, 3.40964533e-02,
                 1.08729564e-02],
                [8.39222729e-01, 1.41632959e-01, 1.36041418e-02, 4.68115415e-03,
                 8.59017367e-04],
                [1.09037854e-01, 2.46659145e-01, 2.40053445e-01, 3.00265551e-01,
                 1.03983983e-01],
                [2.55965441e-01, 3.55194777e-01, 1.74702838e-01, 1.72652677e-01,
                 4.14842218e-02],
                [6.73872709e-01, 2.51807630e-01, 4.59172837e-02, 2.35070121e-02,
                 4.89531457e-03],
                [9.06672299e-01, 8.71637613e-02, 4.74287709e-03, 1.23610883e-03,
                 1.85016863e-04],
                [1.13389641e-01, 2.42101148e-01, 2.42241234e-01, 2.93720275e-01,
                 1.08547643e-01],
                [8.09805870e-01, 1.67246982e-01, 1.58957280e-02, 6.08204398e-03,
                 9.69371991e-04],
                [7.43020594e-01, 2.11406216e-01, 2.97286324e-02, 1.32969376e-02,
                 2.54769134e-03],
                [8.08218479e-01, 1.67094812e-01, 1.70564372e-02, 6.51578791e-03,
                 1.11444131e-03],
                [2.43963584e-01, 3.37263972e-01, 1.86235085e-01, 1.83049321e-01,
                 4.94880304e-02]], dtype=float32)
```

```
In [30]: # generate classification report using predictions for categorical model
         from sklearn.metrics import classification_report, accuracy_score

         categorical_pred = np.argmax(model.predict(X_test), axis=1)
```

```
In [31]: categorical_pred
```

```
Out[31]: array([3, 0, 0, 3, 3, 3, 0, 1, 0, 3, 0, 3, 0, 0, 0, 0, 1, 3, 0, 0, 0, 0,
                1, 0, 3, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 3, 1, 0, 0, 3, 1, 0, 0, 3, 0, 0, 0, 1], dtype=int64)
```

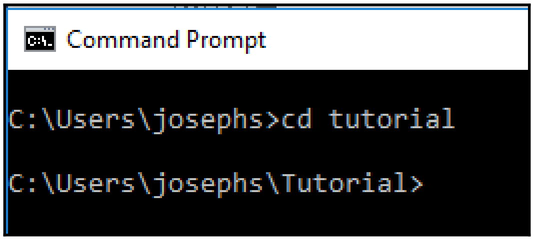
```
In [39]: # generate classification report using predictions for categorical model
         from sklearn.metrics import classification_report, accuracy_score

         categorical_pred = np.argmax(model.predict(X_test), axis=1)

         print('Results for Categorical Model')
         print(accuracy_score(y_test, categorical_pred))
         print(classification_report(y_test, categorical_pred))
```

```
Results for Categorical Model
0.6333333333333333
             precision    recall  f1-score   support

          0       0.84      0.86      0.85        36
          1       0.00      0.00      0.00         9
          2       0.00      0.00      0.00         5
          3       0.35      1.00      0.52         7
          4       0.00      0.00      0.00         3

avg / total       0.54      0.63      0.57        60
```

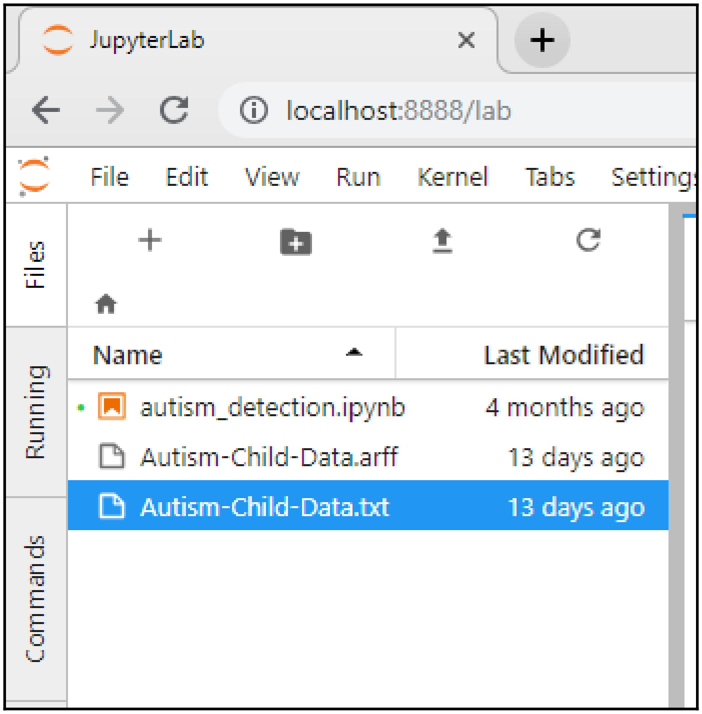
```
In [40]: # generate classification report using predictions for binary model
         binary_pred = np.round(binary_model.predict(X_test)).astype(int)

         print('Results for Binary Model')
         print(accuracy_score(Y_test_binary, binary_pred))
         print(classification_report(Y_test_binary, binary_pred))
```

```
Results for Binary Model
0.8
             precision    recall  f1-score   support

          0       0.83      0.83      0.83        36
          1       0.75      0.75      0.75        24

avg / total       0.80      0.80      0.80        60
```

# Chapter 5: Autism Screening With Machine Learning




| Attribute | Type | Description |
|---|---|---|
| Age | Number | years |
| Gender | String | Male or Female |
| Ethnicity | String | List of common ethnicities in text format |
| Born with jaundice | Boolean (yes or no) | Whether the case was born with jaundice |
| Family member with PDD | Boolean (yes or no) | Whether any immediate family member has a PDD |
| Who is completing the test | String | Parent, self, caregiver, medical staff, clinician, etc. |
| Country of residence | String | List of countries in text format |
| Used the screening app before | Boolean (yes or no) | Whether the user has used a screening app |
| Screening Method Type | Integer (0,1,2,3) | The type of screening methods chosen based on age category (0=toddler, 1=child, 2= adolescent, 3= adult) |
| Question 1 Answer | Binary (0, 1) | The answer code of the question based on the screening method used |
| Question 2 Answer | Binary (0, 1) | The answer code of the question based on the screening method used |
| Question 3 Answer | Binary (0, 1) | The answer code of the question based on the screening method used |
| Question 4 Answer | Binary (0, 1) | The answer code of the question based on the screening method used |
| Question 5 Answer | Binary (0, 1) | The answer code of the question based on the screening method used |
| Question 6 Answer | Binary (0, 1) | The answer code of the question based on the screening method used |
| Question 7 Answer | Binary (0, 1) | The answer code of the question based on the screening method used |
| Question 8 Answer | Binary (0, 1) | The answer code of the question based on the screening method used |
| Question 9 Answer | Binary (0, 1) | The answer code of the question based on the screening method used |
| Question 10 Answer | Binary (0, 1) | The answer code of the question based on the screening method used |
| Screening Score | Integer | The final score obtained based on the scoring algorithm of the screening method used. This was computed in an automated manner |




```
In [1]: import sys
        import pandas as pd
        import sklearn
        import keras

        print 'Python: {}'.format(sys.version)
        print 'Pandas: {}'.format(pd.__version__)
        print 'Sklearn: {}'.format(sklearn.__version__)
        print 'Keras: {}'.format(keras.__version__)

        Using Theano backend.
        WARNING (theano.tensor.blas): Using NumPy C-API based implementation for BLAS functions.

        Python: 2.7.13 |Continuum Analytics, Inc.| (default, May 11 2017, 13:17:26) [MSC v.1500 64 bit (AMD64)]
        Pandas: 0.21.0
        Sklearn: 0.19.1
        Keras: 2.1.4
```

```
Shape of DataFrame: (292, 21)
A1_Score                              1
 A2_Score                             1
 A3_Score                             0
 A4_Score                             0
 A5_Score                             1
 A6_Score                             1
 A7_Score                             0
 A8_Score                             1
 A9_Score                             0
 A10_Score                            0
 age_numeric                          6
 gender                               m
 ethnicity                       Others
 jaundice                            no
 family_history_of_autism            no
 country_of_res                  Jordan
 used_app_before                     no
 result                               5
 age_desc                  '4-11 years'
 relation                        Parent
 Class/ASD                           NO
Name: 0, dtype: object
```

In [16]:

```
# print out multiple patients at the same time
data.loc[:10]
```

Out[16]:

| | A1_Score | A2_Score | A3_Score | A4_Score | A5_Score | A6_Score | A7_Score | A8_Score | A9_Score | A10_Score | ... | gender | ethnicity | jaundice | family_history_of_autism | country_of_res | used_app_before |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | ... | m | Others | no | no | Jordan | no |
| **1** | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | ... | m | 'Middle Eastern ' | no | no | Jordan | no |
| **2** | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | ... | m | ? | no | no | Jordan | yes |
| **3** | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | ... | f | ? | yes | no | Jordan | no |
| **4** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | ... | m | Others | yes | no | 'United States' | no |
| **5** | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | ... | m | ? | no | yes | Egypt | no |
| **6** | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | ... | m | White-European | no | no | 'United Kingdom' | no |
| **7** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | ... | f | 'Middle Eastern ' | no | no | Bahrain | no |
| **8** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | ... | f | 'Middle Eastern ' | no | no | Bahrain | no |
| **9** | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | ... | f | ? | no | yes | Austria | no |
| **10** | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | ... | m | White-European | yes | no | 'United Kingdom' | no |

11 rows × 21 columns

In [17]:

```
# print out a description of the dataframe
data.describe()
```

Out[17]:

| | A1_Score | A2_Score | A3_Score | A4_Score | A5_Score | A6_Score | A7_Score | A8_Score | A9_Score | A10_Score | result |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 292.000000 | 292.000000 | 292.000000 | 292.000000 | 292.000000 | 292.000000 | 292.000000 | 292.000000 | 292.000000 | 292.000000 | 292.000000 |
| **mean** | 0.633562 | 0.534247 | 0.743151 | 0.551370 | 0.743151 | 0.712329 | 0.606164 | 0.496575 | 0.493151 | 0.726027 | 6.239726 |
| **std** | 0.482658 | 0.499682 | 0.437646 | 0.498208 | 0.437646 | 0.453454 | 0.489438 | 0.500847 | 0.500811 | 0.446761 | 2.284882 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 5.000000 |
| **50%** | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 | 1.000000 | 6.000000 |
| **75%** | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 8.000000 |
| **max** | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 10.000000 |

In [24]:
```
data.dtypes
```

Out[24]:
```
A1_Score                    int64
 A2_Score                   int64
 A3_Score                   int64
 A4_Score                   int64
 A5_Score                   int64
 A6_Score                   int64
 A7_Score                   int64
 A8_Score                   int64
 A9_Score                   int64
 A10_Score                  int64
 age_numeric               object
 gender                    object
 ethnicity                 object
 jaundice                  object
 family_history_of_autism  object
 country_of_res            object
 used_app_before           object
 result                     int64
 age_desc                  object
 relation                  object
 Class/ASD                 object
dtype: object
```

In [52]:
```
x.loc[:10]
```

Out[52]:

| | A1_Score | A2_Score | A3_Score | A4_Score | A5_Score | A6_Score | A7_Score | A8_Score | A9_Score | A10_Score | age_numeric | gender | ethnicity | jaundice | family_history_of_autism | country_of_res | used_app_be |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 6 | m | Others | no | no | Jordan | |
| **1** | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 6 | m | 'Middle Eastern ' | no | no | Jordan | |
| **2** | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 6 | m | ? | no | no | Jordan | |
| **3** | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 5 | f | ? | yes | no | Jordan | |
| **4** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 5 | m | Others | yes | no | 'United States' | |
| **5** | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 4 | m | ? | no | yes | Egypt | |
| **6** | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 5 | m | White-European | no | no | 'United Kingdom' | |
| **7** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 5 | f | 'Middle Eastern ' | no | no | Bahrain | |
| **8** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 11 | f | 'Middle Eastern ' | no | no | Bahrain | |
| **9** | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 11 | f | ? | no | yes | Austria | |
| **10** | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 10 | m | White-European | yes | no | 'United Kingdom' | |

```
In [54]: # print the new categorical column labels
         X.columns.values
```

```
Out[54]: array(['A1_Score', ' A2_Score', ' A3_Score', ' A4_Score', ' A5_Score',
               ' A6_Score', ' A7_Score', ' A8_Score', ' A9_Score', ' A10_Score',
               ' age_numeric_10', ' age_numeric_11', ' age_numeric_4',
               ' age_numeric_5', ' age_numeric_6', ' age_numeric_7',
               ' age_numeric_8', ' age_numeric_9', ' age_numeric_?', ' gender_f',
               ' gender_m', " ethnicity_'Middle Eastern '",
               " ethnicity_'South Asian'", ' ethnicity_?', ' ethnicity_Asian',
               ' ethnicity_Black', ' ethnicity_Hispanic', ' ethnicity_Latino',
               ' ethnicity_Others', ' ethnicity_Pasifika', ' ethnicity_Turkish',
               ' ethnicity_White-European', ' jaundice_no', ' jaundice_yes',
               ' family_history_of_autism_no', ' family_history_of_autism_yes',
               " country_of_res_'Costa Rica'", " country_of_res_'Isle of Man'",
               " country_of_res_'New Zealand'", " country_of_res_'Saudi Arabia'",
               " country_of_res_'South Africa'", " country_of_res_'South Korea'",
               " country_of_res_'U.S. Outlying Islands'",
               " country_of_res_'United Arab Emirates'",
               " country_of_res_'United Kingdom'",
               " country_of_res_'United States'", ' country_of_res_Afghanistan',
               ' country_of_res_Argentina', ' country_of_res_Armenia',
               ' country_of_res_Australia', ' country_of_res_Austria',
               ' country_of_res_Bahrain', ' country_of_res_Bangladesh',
               ' country_of_res_Bhutan', ' country_of_res_Brazil',
               ' country_of_res_Bulgaria', ' country_of_res_Canada',
               ' country_of_res_China', ' country_of_res_Egypt',
               ' country_of_res_Europe', ' country_of_res_Georgia',
               ' country_of_res_Germany', ' country_of_res_Ghana',
               ' country_of_res_India', ' country_of_res_Iraq',
               ' country_of_res_Ireland', ' country_of_res_Italy',
               ' country_of_res_Japan', ' country_of_res_Jordan',
               ' country_of_res_Kuwait', ' country_of_res_Latvia',
               ' country_of_res_Lebanon', ' country_of_res_Libya',
               ' country_of_res_Malaysia', ' country_of_res_Malta',
               ' country_of_res_Mexico', ' country_of_res_Nepal',
               ' country_of_res_Netherlands', ' country_of_res_Nigeria',
               ' country_of_res_Oman', ' country_of_res_Pakistan',
               ' country_of_res_Philippines', ' country_of_res_Qatar',
               ' country_of_res_Romania', ' country_of_res_Russia',
               ' country_of_res_Sweden', ' country_of_res_Syria',
               ' country_of_res_Turkey', ' used_app_before_no',
               ' used_app_before_yes', " relation_'Health care professional'",
               ' relation_?', ' relation_Parent', ' relation_Relative',
               ' relation_Self', ' relation_self'], dtype=object)
```

```
In [19]: # print an example patient from the categorical data
         X.loc[1]
```

```
Out[19]: A1_Score                              1        contry_of_res_ Italy                         0
         A2_Score                              1        contry_of_res_ Japan                         0
         A3_Score                              0        contry_of_res_ Jordan                        1
         A4_Score                              0        contry_of_res_ Kuwait                        0
         A5_Score                              1        contry_of_res_ Latvia                        0
         A6_Score                              1        contry_of_res_ Lebanon                       0
         A7_Score                              0        contry_of_res_ Libya                         0
         A8_Score                              1        contry_of_res_ Malaysia                      0
         A9_Score                              0        contry_of_res_ Malta                         0
         A10_Score                             0        contry_of_res_ Mexico                        0
         age numeric_ 10                       0        contry_of_res_ Nepal                         0
         age numeric_ 11                       0        contry_of_res_ Netherlands                   0
         age numeric_ 4                        0        contry_of_res_ Nigeria                       0
         age numeric_ 5                        0        contry_of_res_ Oman                          0
         age numeric_ 6                        1        contry_of_res_ Pakistan                      0
         age numeric_ 7                        0        contry_of_res_ Philippines                   0
         age numeric_ 8                        0        contry_of_res_ Qatar                         0
         age numeric_ 9                        0        contry_of_res_ Romania                       0
         age numeric_ ?                        0        contry_of_res_ Russia                        0
         gender_ f                             0        contry_of_res_ Sweden                        0
         gender_ m                             1        contry_of_res_ Syria                         0
         ethnicity_ 'Middle Eastern '          1        contry_of_res_ Turkey                        0
         ethnicity_ 'South Asian'              0        used_app_before_ no                          1
         ethnicity_ ?                          0        used_app_before_ yes                         0
         ethnicity_ Asian                      0        relation_ 'Health care professional'         0
         ethnicity_ Black                      0        relation_ ?                                  0
         ethnicity_ Hispanic                   0        relation_ Parent                             1
         ethnicity_ Latino                     0        relation_ Relative                           0
         ethnicity_ Others                     0        relation_ Self                               0
         ethnicity_ Pasifika                   0        relation_ self                               0
                                              ..        Name: 1, Length: 96, dtype: int64
```

In [20]:
```python
# convert the class data to categorical values - one-hot-encoded vectors
Y = pd.get_dummies(y)
```

In [21]:
```python
Y.iloc[:10]
```

Out[21]:

| | NO | YES |
|---|---|---|
| **0** | 1 | 0 |
| **1** | 1 | 0 |
| **2** | 1 | 0 |
| **3** | 1 | 0 |
| **4** | 0 | 1 |
| **5** | 1 | 0 |
| **6** | 0 | 1 |
| **7** | 0 | 1 |
| **8** | 0 | 1 |
| **9** | 1 | 0 |

In [24]:
```python
print(X_train.shape)
print (X_test.shape)
print (Y_train.shape)
print (Y_test.shape)
```

```
(233, 96)
(59, 96)
(233, 2)
(59, 2)
```

```
In [34]: model = create_model()
         print(model.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 8)                 776
_________________________________________________________________
dense_5 (Dense)              (None, 4)                 36
_________________________________________________________________
dense_6 (Dense)              (None, 2)                 10
=================================================================
Total params: 822
Trainable params: 822
Non-trainable params: 0
_________________________________________________________________
None
```

```
Epoch 1/50
233/233 [==============================] - 0s 288us/step - loss: 0.6927 - acc: 0.5794
Epoch 2/50
233/233 [==============================] - 0s 245us/step - loss: 0.6910 - acc: 0.7210
Epoch 3/50
233/233 [==============================] - 0s 258us/step - loss: 0.6868 - acc: 0.7639
Epoch 4/50
233/233 [==============================] - 0s 236us/step - loss: 0.6779 - acc: 0.7082
Epoch 5/50
233/233 [==============================] - 0s 236us/step - loss: 0.6619 - acc: 0.8541
Epoch 6/50
233/233 [==============================] - 0s 305us/step - loss: 0.6340 - acc: 0.8283
Epoch 7/50
233/233 [==============================] - 0s 227us/step - loss: 0.5963 - acc: 0.8541
Epoch 8/50
233/233 [==============================] - 0s 305us/step - loss: 0.5446 - acc: 0.9399
Epoch 9/50
233/233 [==============================] - 0s 240us/step - loss: 0.4884 - acc: 0.8884
Epoch 10/50
233/233 [==============================] - 0s 227us/step - loss: 0.4220 - acc: 0.9227
Epoch 11/50
233/233 [==============================] - 0s 322us/step - loss: 0.3603 - acc: 0.9313
Epoch 12/50
233/233 [==============================] - 0s 245us/step - loss: 0.2935 - acc: 0.9614
Epoch 13/50
233/233 [==============================] - 0s 296us/step - loss: 0.2528 - acc: 0.9657
Epoch 14/50
233/233 [==============================] - 0s 330us/step - loss: 0.2087 - acc: 0.9657
Epoch 15/50
233/233 [==============================] - 0s 305us/step - loss: 0.1788 - acc: 0.9871
Epoch 16/50
233/233 [==============================] - 0s 313us/step - loss: 0.1605 - acc: 0.9700
Epoch 17/50
233/233 [==============================] - 0s 309us/step - loss: 0.1389 - acc: 0.9828
Epoch 18/50
233/233 [==============================] - 0s 335us/step - loss: 0.1258 - acc: 0.9785
Epoch 19/50
233/233 [==============================] - 0s 343us/step - loss: 0.1108 - acc: 0.9871
Epoch 20/50
233/233 [==============================] - 0s 399us/step - loss: 0.1004 - acc: 0.9871
Epoch 21/50
233/233 [==============================] - 0s 416us/step - loss: 0.0910 - acc: 0.9871
Epoch 22/50
233/233 [==============================] - 0s 343us/step - loss: 0.0820 - acc: 0.9871
Epoch 23/50
233/233 [==============================] - 0s 361us/step - loss: 0.0752 - acc: 0.9914
Epoch 24/50
233/233 [==============================] - 0s 356us/step - loss: 0.0714 - acc: 0.9957
Epoch 25/50
233/233 [==============================] - 0s 309us/step - loss: 0.0634 - acc: 0.9957
Epoch 26/50
233/233 [==============================] - 0s 339us/step - loss: 0.0585 - acc: 0.9957
Epoch 27/50
233/233 [==============================] - 0s 335us/step - loss: 0.0571 - acc: 1.0000
Epoch 28/50
233/233 [==============================] - 0s 429us/step - loss: 0.0526 - acc: 0.9957
Epoch 29/50
233/233 [==============================] - 0s 335us/step - loss: 0.0474 - acc: 1.0000
Epoch 30/50
233/233 [==============================] - 0s 322us/step - loss: 0.0463 - acc: 0.9957
Epoch 31/50
233/233 [==============================] - 0s 296us/step - loss: 0.0431 - acc: 1.0000
Epoch 32/50
233/233 [==============================] - 0s 348us/step - loss: 0.0381 - acc: 1.0000
Epoch 33/50
233/233 [==============================] - 0s 322us/step - loss: 0.0357 - acc: 1.0000
Epoch 34/50
233/233 [==============================] - 0s 292us/step - loss: 0.0331 - acc: 1.0000
Epoch 35/50
233/233 [==============================] - 0s 305us/step - loss: 0.0316 - acc: 1.0000
Epoch 36/50
233/233 [==============================] - 0s 335us/step - loss: 0.0294 - acc: 1.0000
Epoch 37/50
233/233 [==============================] - 0s 322us/step - loss: 0.0282 - acc: 1.0000
Epoch 38/50
233/233 [==============================] - 0s 236us/step - loss: 0.0281 - acc: 1.0000
Epoch 39/50
233/233 [==============================] - 0s 339us/step - loss: 0.0253 - acc: 1.0000
Epoch 40/50
233/233 [==============================] - 0s 223us/step - loss: 0.0252 - acc: 1.0000
Epoch 41/50
233/233 [==============================] - 0s 326us/step - loss: 0.0226 - acc: 1.0000
Epoch 42/50
233/233 [==============================] - 0s 326us/step - loss: 0.0213 - acc: 1.0000
Epoch 43/50
233/233 [==============================] - 0s 219us/step - loss: 0.0203 - acc: 1.0000
Epoch 44/50
233/233 [==============================] - 0s 215us/step - loss: 0.0193 - acc: 1.0000
Epoch 45/50
233/233 [==============================] - 0s 318us/step - loss: 0.0190 - acc: 1.0000
Epoch 46/50
233/233 [==============================] - 0s 232us/step - loss: 0.0176 - acc: 1.0000
Epoch 47/50
233/233 [==============================] - 0s 215us/step - loss: 0.0163 - acc: 1.0000
Epoch 48/50
233/233 [==============================] - 0s 202us/step - loss: 0.0161 - acc: 1.0000
Epoch 49/50
233/233 [==============================] - 0s 240us/step - loss: 0.0154 - acc: 1.0000
Epoch 50/50
233/233 [==============================] - 0s 223us/step - loss: 0.0150 - acc: 1.0000
```

```
In [25]: # generate classification report using predictions for categorical model
         from sklearn.metrics import classification_report, accuracy_score

         predictions = model.predict_classes(X_test)
         predictions

Out[25]: array([0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1,
                0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
                0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0], dtype=int64)
```

```
Prediction Results for Neural Network
0.8983050847457628
             precision    recall  f1-score   support

          0       0.85      0.97      0.90        29
          1       0.96      0.83      0.89        30

avg / total       0.91      0.90      0.90        59
```

# Index