Package Imports

In [81]:

```
import sys
import pandas as pd
import numpy as np
import sklearn
import matplotlib
import keras
```

In [82]:

```
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
```

Loading the data from cloud

In [83]:

```
# azureml-core of version 1.0.72 or higher is required
# azureml-dataprep[pandas] of version 1.1.34 or higher is required
from azureml.core import Workspace, Dataset

subscription_id = 'd8c9631a-72a2-4c02-bd86-46a64cba0932'
resource_group = 'CIT'
workspace_name = 'MCC'

workspace = Workspace(subscription_id, resource_group, workspace_name)

dataset = Dataset.get_by_name(workspace, name='heart disease')
```

Pre-processing

In [84]:

```
df = dataset.to_pandas_dataframe()
```

In [85]:

```
print( 'Shape of DataFrame: {}'.format(df.shape))
print (df.loc[1])
```

```
Shape of DataFrame: (303, 14)
age          37.0
sex           1.0
cp            2.0
trestbps    130.0
chol        250.0
fbs           0.0
restecg       1.0
thalach     187.0
exang         0.0
oldpeak       3.5
slope         0.0
ca            0.0
thal          2.0
target        1.0
Name: 1, dtype: float64
```

In [86]:

```
data = df[~df.isin(['?'])]
```

Droping Null Values

In [87]:

```
data = data.dropna(axis=0)
```

Expolatory Data Analysis

In [88]:

```
print(data.shape)
print(data.dtypes)
```

```
(303, 14)
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

In [89]:

```
data = data.apply(pd.to_numeric)
data.dtypes
```

Out[89]:

```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

In [90]:

```
data.describe()
```

Out[90]:

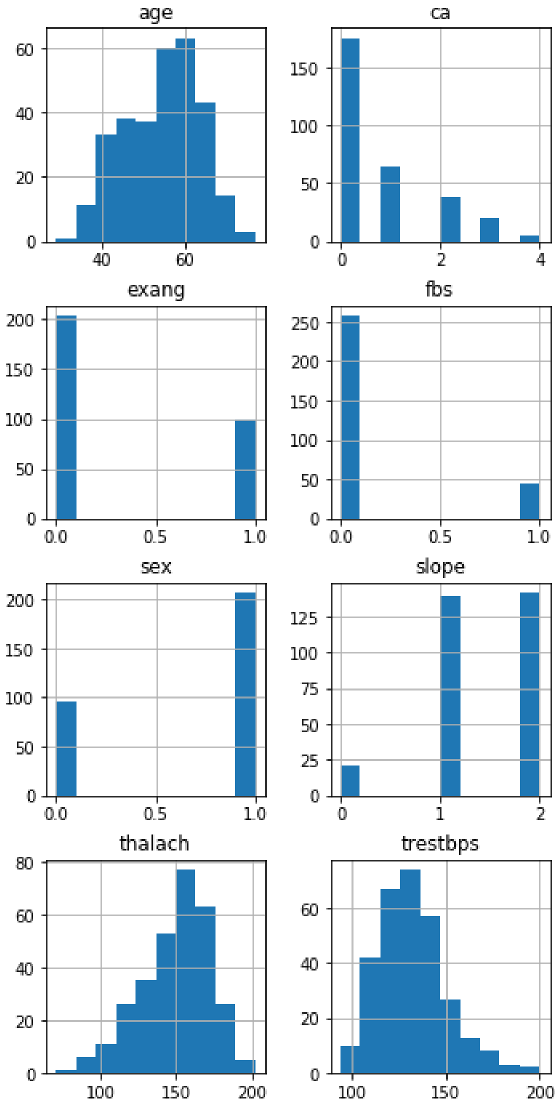
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.39
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.61
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.00

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.0(
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.0(
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.0(
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.0(

Histogram

In [91]:

```
data.hist(figsize = (12, 12))
plt.show()
```



Train and Test split

In [92]:

```
from sklearn import model_selection

X = np.array(data.drop(['target'], 1))
y = np.array(data['target'])

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

Model: Categorical Classification Model

Model: Categorical Classification Model

In [93]:

```
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])
```

```
(242, 2)
[[1. 0.]
 [1. 0.]
 [1. 0.]
 [0. 1.]
 [0. 1.]
 [1. 0.]
 [0. 1.]
 [1. 0.]
 [0. 1.]
 [0. 1.]]
```

Importing the Model

In [94]:

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

Model Creation

In [95]:

```
def create_model():
    # create model
    model = Sequential()
    model.add(Dense(16, input_dim=13, kernel_initializer='normal', activation='relu'))
    model.add(Dense(8, kernel_initializer='normal', activation='relu'))
    model.add(Dense(2, 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())
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_13 (Dense)             (None, 16)                224       
_________________________________________________________________
dense_14 (Dense)             (None, 8)                 136       
_________________________________________________________________
dense_15 (Dense)             (None, 2)                 18        
=================================================================
Total params: 378
Trainable params: 378
Non-trainable params: 0
_________________________________________________________________
None
```

Model Training

In [96]:

```
history=model.fit(X_train, Y_train, validation_data=(X_test, Y_test),epochs=200, batch_size=10, ver
bose = 10)
```

```
Train on 242 samples, validate on 61 samples
Epoch 1/200
Epoch 2/200
Epoch 3/200
Epoch 4/200
Epoch 5/200
Epoch 6/200
Epoch 7/200
Epoch 8/200
Epoch 9/200
Epoch 10/200
Epoch 11/200
Epoch 12/200
Epoch 13/200
Epoch 14/200
Epoch 15/200
Epoch 16/200
Epoch 17/200
Epoch 18/200
Epoch 19/200
Epoch 20/200
Epoch 21/200
Epoch 22/200
Epoch 23/200
Epoch 24/200
Epoch 25/200
Epoch 26/200
Epoch 27/200
Epoch 28/200
Epoch 29/200
Epoch 30/200
Epoch 31/200
Epoch 32/200
Epoch 33/200
Epoch 34/200
Epoch 35/200
Epoch 36/200
Epoch 37/200
Epoch 38/200
Epoch 39/200
Epoch 40/200
Epoch 41/200
Epoch 42/200
Epoch 43/200
Epoch 44/200
Epoch 45/200
Epoch 46/200
Epoch 47/200
Epoch 48/200
Epoch 49/200
Epoch 50/200
Epoch 51/200
Epoch 52/200
Epoch 53/200
Epoch 54/200
Epoch 55/200
Epoch 56/200
Epoch 57/200
Epoch 58/200
Epoch 59/200
Epoch 60/200
Epoch 61/200
Epoch 62/200
Epoch 63/200
Epoch 64/200
Epoch 65/200
Epoch 66/200
Epoch 67/200
Epoch 68/200
Epoch 69/200
Epoch 70/200
Epoch 71/200
Epoch 72/200
Epoch 73/200
Epoch 74/200
Epoch 75/200
Epoch 76/200
Epoch 77/200
Epoch 78/200
Epoch 79/200
Epoch 80/200
Epoch 81/200
Epoch 82/200
Epoch 83/200
Epoch 84/200
Epoch 85/200
Epoch 86/200
Epoch 87/200
Epoch 88/200
Epoch 89/200
Epoch 90/200
Epoch 91/200
Epoch 92/200
Epoch 93/200
Epoch 94/200
Epoch 95/200
Epoch 96/200
Epoch 97/200
Epoch 98/200
Epoch 99/200
Epoch 100/200
Epoch 101/200
Epoch 102/200
Epoch 103/200
Epoch 104/200
Epoch 105/200
Epoch 106/200
Epoch 107/200
Epoch 108/200
Epoch 109/200
Epoch 110/200
Epoch 111/200
Epoch 112/200
Epoch 113/200
Epoch 114/200
Epoch 115/200
Epoch 116/200
Epoch 117/200
Epoch 118/200
Epoch 119/200
Epoch 120/200
Epoch 121/200
Epoch 122/200
Epoch 123/200
Epoch 124/200
Epoch 125/200
Epoch 126/200
Epoch 127/200
Epoch 128/200
Epoch 129/200
Epoch 130/200
Epoch 131/200
Epoch 132/200
Epoch 133/200
Epoch 134/200
Epoch 135/200
Epoch 136/200
Epoch 137/200
Epoch 138/200
Epoch 139/200
Epoch 140/200
Epoch 141/200
Epoch 142/200
Epoch 143/200
Epoch 144/200
Epoch 145/200
Epoch 146/200
Epoch 147/200
Epoch 148/200
Epoch 149/200
Epoch 150/200
Epoch 151/200
Epoch 152/200
Epoch 153/200
Epoch 154/200
Epoch 155/200
Epoch 156/200
Epoch 157/200
Epoch 158/200
Epoch 159/200
Epoch 160/200
Epoch 161/200
Epoch 162/200
Epoch 163/200
Epoch 164/200
Epoch 165/200
Epoch 166/200
Epoch 167/200
Epoch 168/200
Epoch 169/200
Epoch 170/200
Epoch 171/200
Epoch 172/200
Epoch 173/200
Epoch 174/200
Epoch 175/200
Epoch 176/200
Epoch 177/200
Epoch 178/200
Epoch 179/200
Epoch 180/200
Epoch 181/200
Epoch 182/200
Epoch 183/200
Epoch 184/200
Epoch 185/200
Epoch 186/200
Epoch 187/200
Epoch 188/200
Epoch 189/200
Epoch 190/200
Epoch 191/200
Epoch 192/200
Epoch 193/200
Epoch 194/200
Epoch 195/200
Epoch 196/200
Epoch 197/200
Epoch 198/200
Epoch 199/200
Epoch 200/200
```

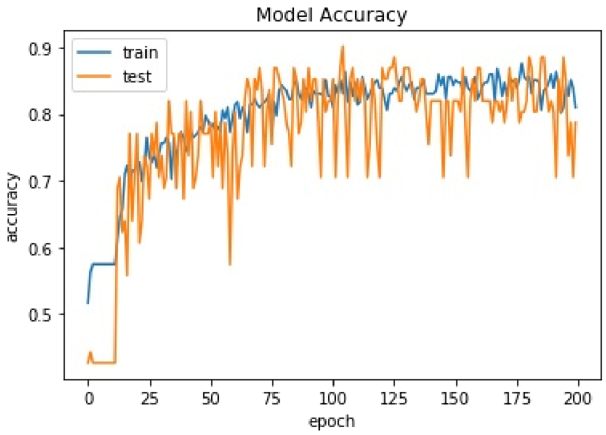
Model Accuracy rate

In [97]:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [98]:

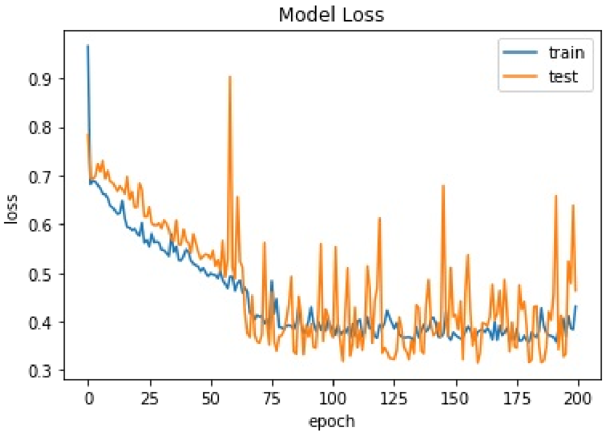
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'])
plt.show()
```



Model Loss rate

In [99]:

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'])
plt.show()
```



Evaluation Metrics

In [100]:

```
from sklearn.metrics import classification_report, accuracy_score

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

print('Acuuracy Score for Categorical Model')
print(accuracy_score(y_test, categorical_pred)*100)
print(classification_report(y_test, categorical_pred))
```

```
Acuuracy Score for Categorical Model
78.68852459016394
              precision    recall  f1-score   support

           0       1.00      0.63      0.77        35
           1       0.67      1.00      0.80        26

   micro avg       0.79      0.79      0.79        61
   macro avg       0.83      0.81      0.79        61
weighted avg       0.86      0.79      0.78        61
```