dl-diabetes

July 21, 2023

[1]:
```python
# Import Library

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout,BatchNormalization,Input
from tensorflow.keras.optimizers import Adam,Adagrad,Adadelta,RMSprop,SGD
from sklearn.metrics import accuracy_score,confusion_matrix
from matplotlib import pyplot
from tensorflow.keras.optimizers.schedules import ExponentialDecay
```

[2]:
```python
#Import Data
from google.colab import files
upload=files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving diabetes.csv to diabetes.csv
```

[3]:
```python
df=pd.read_csv('diabetes.csv')
```

[4]:
```python
df.head()
```

[4]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

<google.colab._quickchart_helpers.SectionTitle at 0x78dafcf770a0>

```
import numpy as np
from google.colab import autoviz
df_6795468589485816921 = autoviz.get_df('df_6795468589485816921')

def value_plot(df, y, sort_ascending=False, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  if sort_ascending:
    df = df.sort_values(y).reset_index(drop=True)
  _, ax = plt.subplots(figsize=figsize)
  df[y].plot(kind='line')
  plt.title(y)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = value_plot(df_6795468589485816921, *['Pregnancies'], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_6795468589485816921 = autoviz.get_df('df_6795468589485816921')

def value_plot(df, y, sort_ascending=False, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  if sort_ascending:
    df = df.sort_values(y).reset_index(drop=True)
  _, ax = plt.subplots(figsize=figsize)
  df[y].plot(kind='line')
  plt.title(y)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = value_plot(df_6795468589485816921, *['Glucose'], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_6795468589485816921 = autoviz.get_df('df_6795468589485816921')

def value_plot(df, y, sort_ascending=False, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  if sort_ascending:
    df = df.sort_values(y).reset_index(drop=True)
  _, ax = plt.subplots(figsize=figsize)
  df[y].plot(kind='line')
  plt.title(y)
```

```
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = value_plot(df_6795468589485816921, *['BloodPressure'], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_6795468589485816921 = autoviz.get_df('df_6795468589485816921')

def value_plot(df, y, sort_ascending=False, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  if sort_ascending:
    df = df.sort_values(y).reset_index(drop=True)
  _, ax = plt.subplots(figsize=figsize)
  df[y].plot(kind='line')
  plt.title(y)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = value_plot(df_6795468589485816921, *['SkinThickness'], **{})
chart
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x78dafcb7e170>
```

```
import numpy as np
from google.colab import autoviz
df_6795468589485816921 = autoviz.get_df('df_6795468589485816921')

def histogram(df, colname, num_bins=20, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  _, ax = plt.subplots(figsize=figsize)
  plt.hist(df[colname], bins=num_bins, histtype='stepfilled')
  plt.ylabel('count')
  plt.title(colname)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = histogram(df_6795468589485816921, *['Pregnancies'], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_6795468589485816921 = autoviz.get_df('df_6795468589485816921')

def histogram(df, colname, num_bins=20, figsize=(2, 1)):
  from matplotlib import pyplot as plt
```

```
  _, ax = plt.subplots(figsize=figsize)
  plt.hist(df[colname], bins=num_bins, histtype='stepfilled')
  plt.ylabel('count')
  plt.title(colname)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = histogram(df_6795468589485816921, *['Glucose'], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_6795468589485816921 = autoviz.get_df('df_6795468589485816921')

def histogram(df, colname, num_bins=20, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  _, ax = plt.subplots(figsize=figsize)
  plt.hist(df[colname], bins=num_bins, histtype='stepfilled')
  plt.ylabel('count')
  plt.title(colname)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = histogram(df_6795468589485816921, *['BloodPressure'], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_6795468589485816921 = autoviz.get_df('df_6795468589485816921')

def histogram(df, colname, num_bins=20, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  _, ax = plt.subplots(figsize=figsize)
  plt.hist(df[colname], bins=num_bins, histtype='stepfilled')
  plt.ylabel('count')
  plt.title(colname)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = histogram(df_6795468589485816921, *['SkinThickness'], **{})
chart
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x78dafc920610>
```

```
import numpy as np
from google.colab import autoviz
df_6795468589485816921 = autoviz.get_df('df_6795468589485816921')
```

```
def scatter_plots(df, colname_pairs, scatter_plot_size=2.5, size=8, alpha=.6):
  from matplotlib import pyplot as plt
  plt.figure(figsize=(len(colname_pairs) * scatter_plot_size, scatter_plot_size))
  for plot_i, (x_colname, y_colname) in enumerate(colname_pairs, start=1):
    ax = plt.subplot(1, len(colname_pairs), plot_i)
    ax.scatter(df[x_colname], df[y_colname], s=size, alpha=alpha)
    plt.xlabel(x_colname)
    plt.ylabel(y_colname)
    ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = scatter_plots(df_6795468589485816921, *[[['Pregnancies', 'Glucose'], ['Glucose', 'BloodPressure'], ['BloodPressure', 'SkinThickness'], ['SkinThickness', 'Insulin']]], **{})
chart
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x78db043d89a0>
```

```
import numpy as np
from google.colab import autoviz
df_6795468589485816921 = autoviz.get_df('df_6795468589485816921')

def time_series_multiline(df, timelike_colname, value_colname, series_colname, figsize=(2.5, 1.3), mpl_palette_name='Dark2'):
  from matplotlib import pyplot as plt
  import seaborn as sns
  palette = list(sns.palettes.mpl_palette(mpl_palette_name))
  def _plot_series(series, series_name, series_index=0):
    if value_colname == 'count()':
      counted = (series[timelike_colname]
                 .value_counts()
                 .reset_index(name='counts')
                 .rename({'index': timelike_colname}, axis=1)
                 .sort_values(timelike_colname, ascending=True))
      xs = counted[timelike_colname]
      ys = counted['counts']
    else:
      xs = series[timelike_colname]
      ys = series[value_colname]
    plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

  fig, ax = plt.subplots(figsize=figsize, layout='constrained')
  df = df.sort_values(timelike_colname, ascending=True)
  if series_colname:
    for i, (series_name, series) in enumerate(df.groupby(series_colname)):
```

```
      _plot_series(series, series_name, i)
    fig.legend(title=series_colname, bbox_to_anchor=(1, 1), loc='upper left')
  else:
    _plot_series(df, '')
  sns.despine(fig=fig, ax=ax)
  plt.xlabel(timelike_colname)
  plt.ylabel(value_colname)
  return autoviz.MplChart.from_current_mpl_state()

chart = time_series_multiline(df_6795468589485816921, *['Insulin',␣
↪'Pregnancies', None], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_6795468589485816921 = autoviz.get_df('df_6795468589485816921')

def time_series_multiline(df, timelike_colname, value_colname, series_colname,␣
↪figsize=(2.5, 1.3), mpl_palette_name='Dark2'):
  from matplotlib import pyplot as plt
  import seaborn as sns
  palette = list(sns.palettes.mpl_palette(mpl_palette_name))
  def _plot_series(series, series_name, series_index=0):
    if value_colname == 'count()':
      counted = (series[timelike_colname]
                 .value_counts()
                 .reset_index(name='counts')
                 .rename({'index': timelike_colname}, axis=1)
                 .sort_values(timelike_colname, ascending=True))
      xs = counted[timelike_colname]
      ys = counted['counts']
    else:
      xs = series[timelike_colname]
      ys = series[value_colname]
    plt.plot(xs, ys, label=series_name, color=palette[series_index %␣
↪len(palette)])

  fig, ax = plt.subplots(figsize=figsize, layout='constrained')
  df = df.sort_values(timelike_colname, ascending=True)
  if series_colname:
    for i, (series_name, series) in enumerate(df.groupby(series_colname)):
      _plot_series(series, series_name, i)
    fig.legend(title=series_colname, bbox_to_anchor=(1, 1), loc='upper left')
  else:
    _plot_series(df, '')
  sns.despine(fig=fig, ax=ax)
  plt.xlabel(timelike_colname)
  plt.ylabel(value_colname)
```

```
  return autoviz.MplChart.from_current_mpl_state()

chart = time_series_multiline(df_6795468589485816921, *['Insulin', 'Glucose',␣
↪None], **{})
chart

import numpy as np
from google.colab import autoviz
df_6795468589485816921 = autoviz.get_df('df_6795468589485816921')

def time_series_multiline(df, timelike_colname, value_colname, series_colname,␣
↪figsize=(2.5, 1.3), mpl_palette_name='Dark2'):
  from matplotlib import pyplot as plt
  import seaborn as sns
  palette = list(sns.palettes.mpl_palette(mpl_palette_name))
  def _plot_series(series, series_name, series_index=0):
    if value_colname == 'count()':
      counted = (series[timelike_colname]
                 .value_counts()
                 .reset_index(name='counts')
                 .rename({'index': timelike_colname}, axis=1)
                 .sort_values(timelike_colname, ascending=True))
      xs = counted[timelike_colname]
      ys = counted['counts']
    else:
      xs = series[timelike_colname]
      ys = series[value_colname]
    plt.plot(xs, ys, label=series_name, color=palette[series_index %␣
↪len(palette)])

  fig, ax = plt.subplots(figsize=figsize, layout='constrained')
  df = df.sort_values(timelike_colname, ascending=True)
  if series_colname:
    for i, (series_name, series) in enumerate(df.groupby(series_colname)):
      _plot_series(series, series_name, i)
    fig.legend(title=series_colname, bbox_to_anchor=(1, 1), loc='upper left')
  else:
    _plot_series(df, '')
  sns.despine(fig=fig, ax=ax)
  plt.xlabel(timelike_colname)
  plt.ylabel(value_colname)
  return autoviz.MplChart.from_current_mpl_state()

chart = time_series_multiline(df_6795468589485816921, *['Insulin',␣
↪'BloodPressure', None], **{})
chart

import numpy as np
```

```
from google.colab import autoviz
df_6795468589485816921 = autoviz.get_df('df_6795468589485816921')

def time_series_multiline(df, timelike_colname, value_colname, series_colname,␣
↪figsize=(2.5, 1.3), mpl_palette_name='Dark2'):
  from matplotlib import pyplot as plt
  import seaborn as sns
  palette = list(sns.palettes.mpl_palette(mpl_palette_name))
  def _plot_series(series, series_name, series_index=0):
    if value_colname == 'count()':
      counted = (series[timelike_colname]
                 .value_counts()
                 .reset_index(name='counts')
                 .rename({'index': timelike_colname}, axis=1)
                 .sort_values(timelike_colname, ascending=True))
      xs = counted[timelike_colname]
      ys = counted['counts']
    else:
      xs = series[timelike_colname]
      ys = series[value_colname]
    plt.plot(xs, ys, label=series_name, color=palette[series_index %␣
↪len(palette)])

  fig, ax = plt.subplots(figsize=figsize, layout='constrained')
  df = df.sort_values(timelike_colname, ascending=True)
  if series_colname:
    for i, (series_name, series) in enumerate(df.groupby(series_colname)):
      _plot_series(series, series_name, i)
    fig.legend(title=series_colname, bbox_to_anchor=(1, 1), loc='upper left')
  else:
    _plot_series(df, '')
  sns.despine(fig=fig, ax=ax)
  plt.xlabel(timelike_colname)
  plt.ylabel(value_colname)
  return autoviz.MplChart.from_current_mpl_state()

chart = time_series_multiline(df_6795468589485816921, *['Insulin',␣
↪'SkinThickness', None], **{})
chart
```

[5]:
```
df.isna().mean()*100
```

[5]:
```
Pregnancies                 0.0
Glucose                     0.0
BloodPressure               0.0
SkinThickness               0.0
Insulin                     0.0
```

```
BMI                         0.0
DiabetesPedigreeFunction    0.0
Age                         0.0
Outcome                     0.0
dtype: float64
```

[6]:
```
df.describe()
```

[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin
count	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479
std	3.369578	31.972618	19.355807	15.952218	115.244002
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000
75%	6.000000	140.250000	80.000000	32.000000	127.250000
max	17.000000	199.000000	122.000000	99.000000	846.000000

	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000
mean	31.992578	0.471876	33.240885	0.348958
std	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.078000	21.000000	0.000000
25%	27.300000	0.243750	24.000000	0.000000
50%	32.000000	0.372500	29.000000	0.000000
75%	36.600000	0.626250	41.000000	1.000000
max	67.100000	2.420000	81.000000	1.000000

```
<google.colab._quickchart_helpers.SectionTitle at 0x78dafcc018a0>
```

```
import numpy as np
from google.colab import autoviz
df_8685259749289173088 = autoviz.get_df('df_8685259749289173088')

def value_plot(df, y, sort_ascending=False, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  if sort_ascending:
    df = df.sort_values(y).reset_index(drop=True)
  _, ax = plt.subplots(figsize=figsize)
  df[y].plot(kind='line')
  plt.title(y)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = value_plot(df_8685259749289173088, *['Pregnancies'], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_8685259749289173088 = autoviz.get_df('df_8685259749289173088')

def value_plot(df, y, sort_ascending=False, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  if sort_ascending:
    df = df.sort_values(y).reset_index(drop=True)
  _, ax = plt.subplots(figsize=figsize)
  df[y].plot(kind='line')
  plt.title(y)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = value_plot(df_8685259749289173088, *['Glucose'], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_8685259749289173088 = autoviz.get_df('df_8685259749289173088')

def value_plot(df, y, sort_ascending=False, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  if sort_ascending:
    df = df.sort_values(y).reset_index(drop=True)
  _, ax = plt.subplots(figsize=figsize)
  df[y].plot(kind='line')
  plt.title(y)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = value_plot(df_8685259749289173088, *['BloodPressure'], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_8685259749289173088 = autoviz.get_df('df_8685259749289173088')

def value_plot(df, y, sort_ascending=False, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  if sort_ascending:
    df = df.sort_values(y).reset_index(drop=True)
  _, ax = plt.subplots(figsize=figsize)
  df[y].plot(kind='line')
  plt.title(y)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
```

```
  return autoviz.MplChart.from_current_mpl_state()

chart = value_plot(df_8685259749289173088, *['SkinThickness'], **{})
chart
```

<google.colab._quickchart_helpers.SectionTitle at 0x78db0c55feb0>

```
import numpy as np
from google.colab import autoviz
df_8685259749289173088 = autoviz.get_df('df_8685259749289173088')

def histogram(df, colname, num_bins=20, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  _, ax = plt.subplots(figsize=figsize)
  plt.hist(df[colname], bins=num_bins, histtype='stepfilled')
  plt.ylabel('count')
  plt.title(colname)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = histogram(df_8685259749289173088, *['Pregnancies'], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_8685259749289173088 = autoviz.get_df('df_8685259749289173088')

def histogram(df, colname, num_bins=20, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  _, ax = plt.subplots(figsize=figsize)
  plt.hist(df[colname], bins=num_bins, histtype='stepfilled')
  plt.ylabel('count')
  plt.title(colname)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = histogram(df_8685259749289173088, *['Glucose'], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_8685259749289173088 = autoviz.get_df('df_8685259749289173088')

def histogram(df, colname, num_bins=20, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  _, ax = plt.subplots(figsize=figsize)
  plt.hist(df[colname], bins=num_bins, histtype='stepfilled')
  plt.ylabel('count')
```

```
  plt.title(colname)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = histogram(df_8685259749289173088, *['BloodPressure'], **{})
chart
```

```
import numpy as np
from google.colab import autoviz
df_8685259749289173088 = autoviz.get_df('df_8685259749289173088')

def histogram(df, colname, num_bins=20, figsize=(2, 1)):
  from matplotlib import pyplot as plt
  _, ax = plt.subplots(figsize=figsize)
  plt.hist(df[colname], bins=num_bins, histtype='stepfilled')
  plt.ylabel('count')
  plt.title(colname)
  ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = histogram(df_8685259749289173088, *['SkinThickness'], **{})
chart
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x78db0794b490>
```

```
import numpy as np
from google.colab import autoviz
df_8685259749289173088 = autoviz.get_df('df_8685259749289173088')

def scatter_plots(df, colname_pairs, scatter_plot_size=2.5, size=8, alpha=.6):
  from matplotlib import pyplot as plt
  plt.figure(figsize=(len(colname_pairs) * scatter_plot_size, scatter_plot_size))
  for plot_i, (x_colname, y_colname) in enumerate(colname_pairs, start=1):
    ax = plt.subplot(1, len(colname_pairs), plot_i)
    ax.scatter(df[x_colname], df[y_colname], s=size, alpha=alpha)
    plt.xlabel(x_colname)
    plt.ylabel(y_colname)
    ax.spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = scatter_plots(df_8685259749289173088, *[[['Pregnancies', 'Glucose'],
 ↪['Glucose', 'BloodPressure'], ['BloodPressure', 'SkinThickness'],
 ↪['SkinThickness', 'Insulin']]], **{})
chart
```

[7]:

```
df.shape
```

[7]: (768, 9)

[8]:
```
df.value_counts('Outcome')
```

[8]:
```
Outcome
0    500
1    268
dtype: int64
```

[9]:
```
X=df.drop(['Outcome'],axis=1)
y=df.Outcome
```

[10]:
```
# Split X and Y
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.10,␣
↪stratify = y, random_state = 987)
X_train, X_val, y_train, y_val = train_test_split(X_train,y_train,test_size = 0.
↪15, stratify = y_train, random_state = 987)
print(X_train.shape,y_train.shape)
print(X_val.shape,y_val.shape)
print(X_test.shape,y_test.shape)
```

```
(587, 8) (587,)
(104, 8) (104,)
(77, 8) (77,)
```

[11]:
```
#Now Scaling Dataset
scaler = StandardScaler()
X_train_scale = scaler.fit_transform(X_train)
X_val_scale = scaler.transform(X_val)
X_test_scale = scaler.transform(X_test)
```

[12]:
```
# NUmber of Features
print(X_train_scale.shape[1],)
```

```
8
```

[13]:
```
#lets Build Model
model=Sequential()
# number of input will be=(Total_number of train Example,8)
model.add(Input(shape=(X_train_scale.shape[1],)))
# Hidden Layers
model.add(Dense(units=64,activation='relu'))
# Hidden Layers
model.add(Dense(units=32,activation='relu'))
# Hidden Layers
model.add(Dense(units=16,activation='relu'))
#Dropout Layers,This is classification
```

```
model.add(Dense(units=1,activation='sigmoid'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                576

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 16)                528

 dense_3 (Dense)             (None, 1)                 17

=================================================================
Total params: 3,201
Trainable params: 3,201
Non-trainable params: 0
_________________________________________________________________
```

[14]:

```
from sklearn import metrics
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

[15]:

```
history=model.
 ↪fit(X_train_scale,y_train,validation_data=(X_val_scale,y_val),epochs=100,verbose=1)
```

```
Epoch 1/100
19/19 [==============================] - 2s 15ms/step - loss: 0.6576 - accuracy:
0.6559 - val_loss: 0.6071 - val_accuracy: 0.6731
Epoch 2/100
19/19 [==============================] - 0s 4ms/step - loss: 0.5841 - accuracy:
0.7155 - val_loss: 0.5563 - val_accuracy: 0.7019
Epoch 3/100
19/19 [==============================] - 0s 4ms/step - loss: 0.5265 - accuracy:
0.7615 - val_loss: 0.5208 - val_accuracy: 0.7308
Epoch 4/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4892 - accuracy:
0.7717 - val_loss: 0.4935 - val_accuracy: 0.7404
Epoch 5/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4641 - accuracy:
0.7853 - val_loss: 0.4762 - val_accuracy: 0.7500
Epoch 6/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4473 - accuracy:
0.7871 - val_loss: 0.4761 - val_accuracy: 0.7500
Epoch 7/100
```

```
19/19 [==============================] - 0s 5ms/step - loss: 0.4364 - accuracy:
0.7888 - val_loss: 0.4647 - val_accuracy: 0.7596
Epoch 8/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4305 - accuracy:
0.7956 - val_loss: 0.4629 - val_accuracy: 0.7692
Epoch 9/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4259 - accuracy:
0.7905 - val_loss: 0.4632 - val_accuracy: 0.7788
Epoch 10/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4204 - accuracy:
0.7990 - val_loss: 0.4585 - val_accuracy: 0.7692
Epoch 11/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4154 - accuracy:
0.8007 - val_loss: 0.4574 - val_accuracy: 0.7500
Epoch 12/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4110 - accuracy:
0.8041 - val_loss: 0.4586 - val_accuracy: 0.7500
Epoch 13/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4063 - accuracy:
0.8092 - val_loss: 0.4576 - val_accuracy: 0.7404
Epoch 14/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4016 - accuracy:
0.8126 - val_loss: 0.4490 - val_accuracy: 0.7500
Epoch 15/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3982 - accuracy:
0.8109 - val_loss: 0.4494 - val_accuracy: 0.7500
Epoch 16/100
19/19 [==============================] - 0s 5ms/step - loss: 0.3946 - accuracy:
0.8160 - val_loss: 0.4594 - val_accuracy: 0.7692
Epoch 17/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3914 - accuracy:
0.8143 - val_loss: 0.4548 - val_accuracy: 0.7596
Epoch 18/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3866 - accuracy:
0.8177 - val_loss: 0.4594 - val_accuracy: 0.7692
Epoch 19/100
19/19 [==============================] - 0s 5ms/step - loss: 0.3845 - accuracy:
0.8126 - val_loss: 0.4556 - val_accuracy: 0.7788
Epoch 20/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3801 - accuracy:
0.8194 - val_loss: 0.4623 - val_accuracy: 0.7692
Epoch 21/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3777 - accuracy:
0.8143 - val_loss: 0.4503 - val_accuracy: 0.7692
Epoch 22/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3740 - accuracy:
0.8228 - val_loss: 0.4564 - val_accuracy: 0.7692
Epoch 23/100
```

```
19/19 [==============================] - 0s 4ms/step - loss: 0.3706 - accuracy:
0.8177 - val_loss: 0.4620 - val_accuracy: 0.7788
Epoch 24/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3669 - accuracy:
0.8279 - val_loss: 0.4577 - val_accuracy: 0.7692
Epoch 25/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3662 - accuracy:
0.8194 - val_loss: 0.4560 - val_accuracy: 0.7500
Epoch 26/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3616 - accuracy:
0.8296 - val_loss: 0.4711 - val_accuracy: 0.7596
Epoch 27/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3593 - accuracy:
0.8348 - val_loss: 0.4554 - val_accuracy: 0.7596
Epoch 28/100
19/19 [==============================] - 0s 5ms/step - loss: 0.3536 - accuracy:
0.8348 - val_loss: 0.4594 - val_accuracy: 0.7692
Epoch 29/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3564 - accuracy:
0.8313 - val_loss: 0.4645 - val_accuracy: 0.7596
Epoch 30/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3497 - accuracy:
0.8330 - val_loss: 0.4585 - val_accuracy: 0.7692
Epoch 31/100
19/19 [==============================] - 0s 5ms/step - loss: 0.3472 - accuracy:
0.8365 - val_loss: 0.4723 - val_accuracy: 0.7596
Epoch 32/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3450 - accuracy:
0.8433 - val_loss: 0.4601 - val_accuracy: 0.7596
Epoch 33/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3423 - accuracy:
0.8433 - val_loss: 0.4734 - val_accuracy: 0.7692
Epoch 34/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3376 - accuracy:
0.8433 - val_loss: 0.4692 - val_accuracy: 0.7596
Epoch 35/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3318 - accuracy:
0.8433 - val_loss: 0.4621 - val_accuracy: 0.7692
Epoch 36/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3300 - accuracy:
0.8501 - val_loss: 0.4697 - val_accuracy: 0.7788
Epoch 37/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3258 - accuracy:
0.8518 - val_loss: 0.4640 - val_accuracy: 0.7692
Epoch 38/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3211 - accuracy:
0.8552 - val_loss: 0.4724 - val_accuracy: 0.7596
Epoch 39/100
```

```
19/19 [==============================] - 0s 5ms/step - loss: 0.3209 - accuracy:
0.8637 - val_loss: 0.4705 - val_accuracy: 0.7692
Epoch 40/100
19/19 [==============================] - 0s 5ms/step - loss: 0.3189 - accuracy:
0.8518 - val_loss: 0.4889 - val_accuracy: 0.7596
Epoch 41/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3170 - accuracy:
0.8501 - val_loss: 0.4773 - val_accuracy: 0.7596
Epoch 42/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3081 - accuracy:
0.8603 - val_loss: 0.4764 - val_accuracy: 0.7596
Epoch 43/100
19/19 [==============================] - 0s 5ms/step - loss: 0.3046 - accuracy:
0.8722 - val_loss: 0.4808 - val_accuracy: 0.7692
Epoch 44/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3085 - accuracy:
0.8637 - val_loss: 0.4917 - val_accuracy: 0.7308
Epoch 45/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3019 - accuracy:
0.8756 - val_loss: 0.4829 - val_accuracy: 0.7692
Epoch 46/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2972 - accuracy:
0.8654 - val_loss: 0.4946 - val_accuracy: 0.7596
Epoch 47/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2921 - accuracy:
0.8739 - val_loss: 0.4916 - val_accuracy: 0.7788
Epoch 48/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2870 - accuracy:
0.8722 - val_loss: 0.4954 - val_accuracy: 0.7596
Epoch 49/100
19/19 [==============================] - 0s 5ms/step - loss: 0.2850 - accuracy:
0.8756 - val_loss: 0.5023 - val_accuracy: 0.7692
Epoch 50/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2819 - accuracy:
0.8773 - val_loss: 0.4877 - val_accuracy: 0.7596
Epoch 51/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2786 - accuracy:
0.8825 - val_loss: 0.5156 - val_accuracy: 0.7596
Epoch 52/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2760 - accuracy:
0.8756 - val_loss: 0.4978 - val_accuracy: 0.7692
Epoch 53/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2697 - accuracy:
0.8807 - val_loss: 0.5034 - val_accuracy: 0.7885
Epoch 54/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2712 - accuracy:
0.8756 - val_loss: 0.5286 - val_accuracy: 0.7404
Epoch 55/100
```

```
19/19 [==============================] - 0s 4ms/step - loss: 0.2646 - accuracy:
0.8825 - val_loss: 0.5040 - val_accuracy: 0.7596
Epoch 56/100
19/19 [==============================] - 0s 5ms/step - loss: 0.2576 - accuracy:
0.8825 - val_loss: 0.5227 - val_accuracy: 0.7692
Epoch 57/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2543 - accuracy:
0.8842 - val_loss: 0.5207 - val_accuracy: 0.7788
Epoch 58/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2508 - accuracy:
0.8876 - val_loss: 0.5429 - val_accuracy: 0.7404
Epoch 59/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2462 - accuracy:
0.8961 - val_loss: 0.5388 - val_accuracy: 0.7596
Epoch 60/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2438 - accuracy:
0.8995 - val_loss: 0.5402 - val_accuracy: 0.7596
Epoch 61/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2423 - accuracy:
0.8961 - val_loss: 0.5305 - val_accuracy: 0.7692
Epoch 62/100
19/19 [==============================] - 0s 5ms/step - loss: 0.2407 - accuracy:
0.8944 - val_loss: 0.5568 - val_accuracy: 0.7404
Epoch 63/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2306 - accuracy:
0.9012 - val_loss: 0.5489 - val_accuracy: 0.7500
Epoch 64/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2317 - accuracy:
0.8995 - val_loss: 0.5444 - val_accuracy: 0.7500
Epoch 65/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2245 - accuracy:
0.8961 - val_loss: 0.5740 - val_accuracy: 0.7404
Epoch 66/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2258 - accuracy:
0.9080 - val_loss: 0.5705 - val_accuracy: 0.7596
Epoch 67/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2157 - accuracy:
0.9080 - val_loss: 0.5579 - val_accuracy: 0.7692
Epoch 68/100
19/19 [==============================] - 0s 5ms/step - loss: 0.2121 - accuracy:
0.9046 - val_loss: 0.6119 - val_accuracy: 0.7308
Epoch 69/100
19/19 [==============================] - 0s 4ms/step - loss: 0.2011 - accuracy:
0.9233 - val_loss: 0.5782 - val_accuracy: 0.7404
Epoch 70/100
19/19 [==============================] - 0s 5ms/step - loss: 0.2081 - accuracy:
0.9097 - val_loss: 0.5847 - val_accuracy: 0.7596
Epoch 71/100
```

```
19/19 [==============================] - 0s 4ms/step - loss: 0.2017 - accuracy:
0.9080 - val_loss: 0.5956 - val_accuracy: 0.7500
Epoch 72/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1955 - accuracy:
0.9250 - val_loss: 0.6040 - val_accuracy: 0.7596
Epoch 73/100
19/19 [==============================] - 0s 3ms/step - loss: 0.1886 - accuracy:
0.9199 - val_loss: 0.6126 - val_accuracy: 0.7500
Epoch 74/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1844 - accuracy:
0.9284 - val_loss: 0.6258 - val_accuracy: 0.7500
Epoch 75/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1803 - accuracy:
0.9336 - val_loss: 0.6408 - val_accuracy: 0.7692
Epoch 76/100
19/19 [==============================] - 0s 5ms/step - loss: 0.1858 - accuracy:
0.9267 - val_loss: 0.6642 - val_accuracy: 0.7308
Epoch 77/100
19/19 [==============================] - 0s 5ms/step - loss: 0.1825 - accuracy:
0.9250 - val_loss: 0.6197 - val_accuracy: 0.7404
Epoch 78/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1752 - accuracy:
0.9421 - val_loss: 0.6771 - val_accuracy: 0.7308
Epoch 79/100
19/19 [==============================] - 0s 5ms/step - loss: 0.1678 - accuracy:
0.9353 - val_loss: 0.6664 - val_accuracy: 0.7500
Epoch 80/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1651 - accuracy:
0.9421 - val_loss: 0.6592 - val_accuracy: 0.7500
Epoch 81/100
19/19 [==============================] - 0s 5ms/step - loss: 0.1663 - accuracy:
0.9387 - val_loss: 0.6717 - val_accuracy: 0.7596
Epoch 82/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1577 - accuracy:
0.9421 - val_loss: 0.6640 - val_accuracy: 0.7404
Epoch 83/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1545 - accuracy:
0.9489 - val_loss: 0.6746 - val_accuracy: 0.7692
Epoch 84/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1515 - accuracy:
0.9506 - val_loss: 0.6870 - val_accuracy: 0.7500
Epoch 85/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1479 - accuracy:
0.9523 - val_loss: 0.7100 - val_accuracy: 0.7404
Epoch 86/100
19/19 [==============================] - 0s 5ms/step - loss: 0.1426 - accuracy:
0.9557 - val_loss: 0.7126 - val_accuracy: 0.7404
Epoch 87/100
```

```
19/19 [==============================] - 0s 5ms/step - loss: 0.1440 - accuracy:
0.9455 - val_loss: 0.7509 - val_accuracy: 0.7596
Epoch 88/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1402 - accuracy:
0.9557 - val_loss: 0.7379 - val_accuracy: 0.7596
Epoch 89/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1353 - accuracy:
0.9523 - val_loss: 0.7501 - val_accuracy: 0.7596
Epoch 90/100
19/19 [==============================] - 0s 5ms/step - loss: 0.1260 - accuracy:
0.9608 - val_loss: 0.7559 - val_accuracy: 0.7500
Epoch 91/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1294 - accuracy:
0.9625 - val_loss: 0.7516 - val_accuracy: 0.7404
Epoch 92/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1238 - accuracy:
0.9608 - val_loss: 0.7479 - val_accuracy: 0.7212
Epoch 93/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1155 - accuracy:
0.9625 - val_loss: 0.7635 - val_accuracy: 0.7308
Epoch 94/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1136 - accuracy:
0.9744 - val_loss: 0.7748 - val_accuracy: 0.7212
Epoch 95/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1174 - accuracy:
0.9642 - val_loss: 0.7839 - val_accuracy: 0.7308
Epoch 96/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1206 - accuracy:
0.9608 - val_loss: 0.7616 - val_accuracy: 0.7500
Epoch 97/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1081 - accuracy:
0.9693 - val_loss: 0.7649 - val_accuracy: 0.7500
Epoch 98/100
19/19 [==============================] - 0s 4ms/step - loss: 0.1065 - accuracy:
0.9693 - val_loss: 0.7972 - val_accuracy: 0.7404
Epoch 99/100
19/19 [==============================] - 0s 4ms/step - loss: 0.0969 - accuracy:
0.9796 - val_loss: 0.8124 - val_accuracy: 0.7404
Epoch 100/100
19/19 [==============================] - 0s 4ms/step - loss: 0.0944 - accuracy:
0.9779 - val_loss: 0.7916 - val_accuracy: 0.7404
```

[16]:
```
y_pred_train=model.predict(X_train_scale)
y_test_pred=model.predict(X_test_scale)
```

```
19/19 [==============================] - 0s 2ms/step
3/3 [==============================] - 0s 5ms/step
```

[17]:
```
cm = confusion_matrix(y_pred=y_test_pred> 0.5,y_true=y_test)
cm
```

[17]:
```
array([[34, 16],
       [ 9, 18]])
```

[18]:
```
cm = confusion_matrix(y_pred=y_pred_train> 0.5,y_true=y_train)
cm
```

[18]:
```
array([[378,   4],
       [  7, 198]])
```

[19]:
```
print(confusion_matrix(y_pred_train>0.5,y_train))
```

```
[[378   7]
 [  4 198]]
```

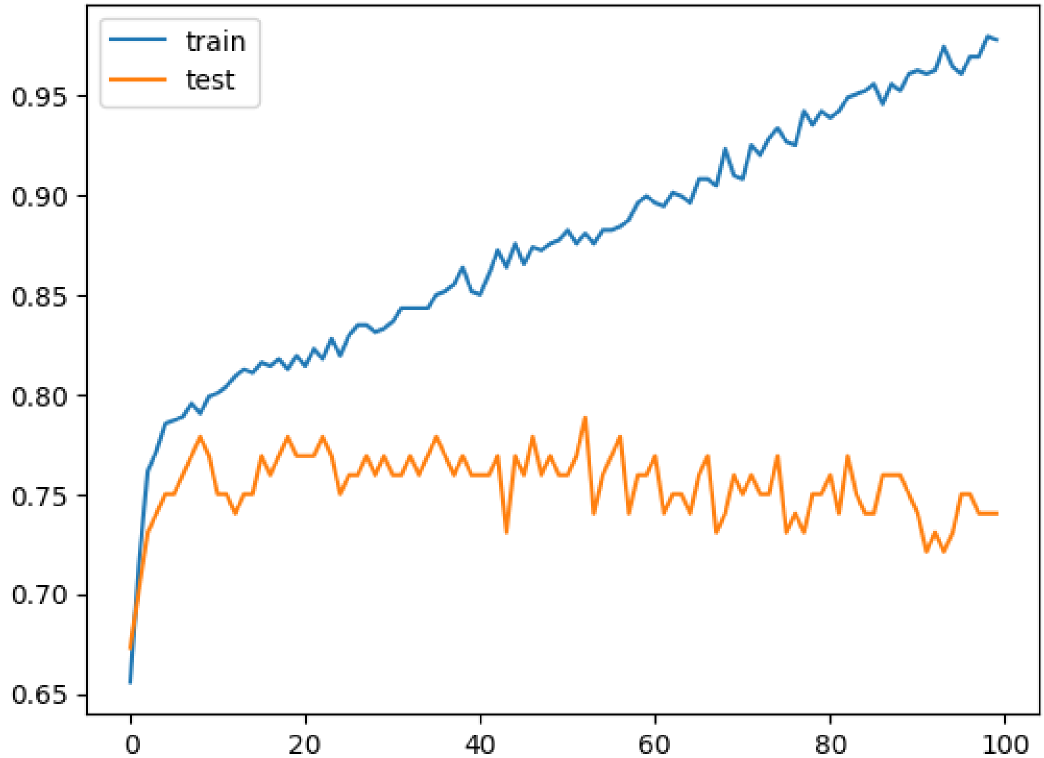
[20]:
```
accuracy_score(y_pred_train>0.5,y_train)
```

[20]:
```
0.9812606473594548
```

[21]:
```
accuracy_score(y_test_pred>0.5,y_test)
```

[21]:
```
0.6753246753246753
```

[22]:
```
from sqlalchemy import label
pyplot.plot(history.history['accuracy'],label='train')
pyplot.plot(history.history['val_accuracy'],label='test')
pyplot.legend()
pyplot.show()
```



[23]:
```
#Let Add Regularization -DropOut
model=Sequential()
#model input with 8 features
model.add(Input(shape=(X_train_scale.shape[1],)))
# Hidden Layer1 with Dropout
model.add(Dense(units=64,activation='relu'))
model.add(Dropout(0.2))
# Hidden Layer1 with Dropout
model.add(Dense(units=32,activation='relu'))
model.add(Dropout(0.2))
# Hidden Layer1 with Dropout
model.add(Dense(units=16,activation='relu'))
model.add(Dropout(0.2))
#output layer
model.add(Dense(units=1,activation='sigmoid'))
# model Summary
model.summary()
```

```
Model: "sequential_1"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
 dense_4 (Dense)             (None, 64)                576

 dropout (Dropout)           (None, 64)                0

 dense_5 (Dense)             (None, 32)                2080

 dropout_1 (Dropout)         (None, 32)                0

 dense_6 (Dense)             (None, 16)                528

 dropout_2 (Dropout)         (None, 16)                0

 dense_7 (Dense)             (None, 1)                 17

=================================================================
Total params: 3,201
Trainable params: 3,201
Non-trainable params: 0
_________________________________________________________________
```

[24]:
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

[25]:
```
history=model.
 ↪fit(X_train_scale,y_train,validation_data=(X_val_scale,y_val),epochs=50,verbose=1)
```

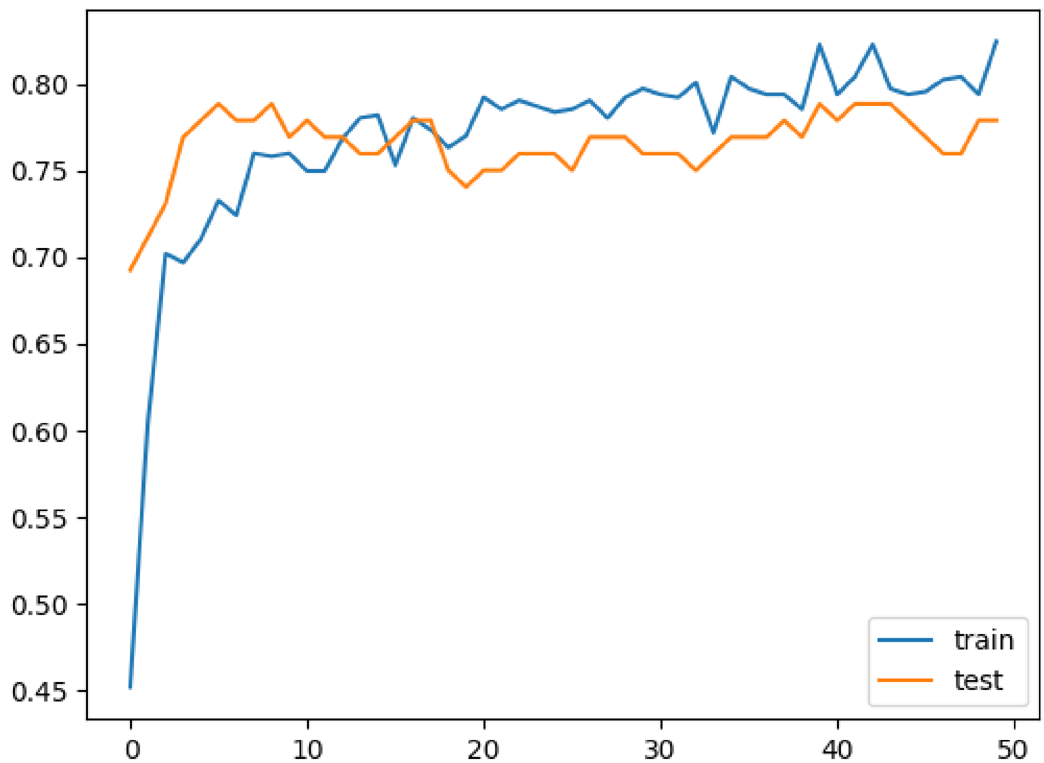
```
Epoch 1/50
19/19 [==============================] - 1s 14ms/step - loss: 0.7362 - accuracy:
0.4514 - val_loss: 0.6783 - val_accuracy: 0.6923
Epoch 2/50
19/19 [==============================] - 0s 4ms/step - loss: 0.6770 - accuracy:
0.6048 - val_loss: 0.6485 - val_accuracy: 0.7115
Epoch 3/50
19/19 [==============================] - 0s 5ms/step - loss: 0.6399 - accuracy:
0.7019 - val_loss: 0.6138 - val_accuracy: 0.7308
Epoch 4/50
19/19 [==============================] - 0s 4ms/step - loss: 0.6185 - accuracy:
0.6968 - val_loss: 0.5702 - val_accuracy: 0.7692
Epoch 5/50
19/19 [==============================] - 0s 5ms/step - loss: 0.5750 - accuracy:
0.7104 - val_loss: 0.5202 - val_accuracy: 0.7788
Epoch 6/50
19/19 [==============================] - 0s 5ms/step - loss: 0.5387 - accuracy:
0.7325 - val_loss: 0.4814 - val_accuracy: 0.7885
Epoch 7/50
19/19 [==============================] - 0s 4ms/step - loss: 0.5250 - accuracy:
0.7240 - val_loss: 0.4677 - val_accuracy: 0.7788
Epoch 8/50
19/19 [==============================] - 0s 5ms/step - loss: 0.5050 - accuracy:
```

```
0.7598 - val_loss: 0.4546 - val_accuracy: 0.7788
Epoch 9/50
19/19 [==============================] - 0s 4ms/step - loss: 0.5032 - accuracy:
0.7581 - val_loss: 0.4535 - val_accuracy: 0.7885
Epoch 10/50
19/19 [==============================] - 0s 5ms/step - loss: 0.4934 - accuracy:
0.7598 - val_loss: 0.4462 - val_accuracy: 0.7692
Epoch 11/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4913 - accuracy:
0.7496 - val_loss: 0.4406 - val_accuracy: 0.7788
Epoch 12/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4821 - accuracy:
0.7496 - val_loss: 0.4426 - val_accuracy: 0.7692
Epoch 13/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4736 - accuracy:
0.7683 - val_loss: 0.4494 - val_accuracy: 0.7692
Epoch 14/50
19/19 [==============================] - 0s 5ms/step - loss: 0.4584 - accuracy:
0.7802 - val_loss: 0.4451 - val_accuracy: 0.7596
Epoch 15/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4636 - accuracy:
0.7819 - val_loss: 0.4393 - val_accuracy: 0.7596
Epoch 16/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4979 - accuracy:
0.7530 - val_loss: 0.4366 - val_accuracy: 0.7692
Epoch 17/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4636 - accuracy:
0.7802 - val_loss: 0.4393 - val_accuracy: 0.7788
Epoch 18/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4594 - accuracy:
0.7734 - val_loss: 0.4422 - val_accuracy: 0.7788
Epoch 19/50
19/19 [==============================] - 0s 5ms/step - loss: 0.4600 - accuracy:
0.7632 - val_loss: 0.4430 - val_accuracy: 0.7500
Epoch 20/50
19/19 [==============================] - 0s 5ms/step - loss: 0.4743 - accuracy:
0.7700 - val_loss: 0.4500 - val_accuracy: 0.7404
Epoch 21/50
19/19 [==============================] - 0s 5ms/step - loss: 0.4555 - accuracy:
0.7922 - val_loss: 0.4519 - val_accuracy: 0.7500
Epoch 22/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4548 - accuracy:
0.7853 - val_loss: 0.4433 - val_accuracy: 0.7500
Epoch 23/50
19/19 [==============================] - 0s 5ms/step - loss: 0.4402 - accuracy:
0.7905 - val_loss: 0.4433 - val_accuracy: 0.7596
Epoch 24/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4549 - accuracy:
```

```
0.7871 - val_loss: 0.4461 - val_accuracy: 0.7596
Epoch 25/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4480 - accuracy:
0.7836 - val_loss: 0.4461 - val_accuracy: 0.7596
Epoch 26/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4553 - accuracy:
0.7853 - val_loss: 0.4434 - val_accuracy: 0.7500
Epoch 27/50
19/19 [==============================] - 0s 5ms/step - loss: 0.4379 - accuracy:
0.7905 - val_loss: 0.4470 - val_accuracy: 0.7692
Epoch 28/50
19/19 [==============================] - 0s 5ms/step - loss: 0.4566 - accuracy:
0.7802 - val_loss: 0.4515 - val_accuracy: 0.7692
Epoch 29/50
19/19 [==============================] - 0s 5ms/step - loss: 0.4502 - accuracy:
0.7922 - val_loss: 0.4553 - val_accuracy: 0.7692
Epoch 30/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4257 - accuracy:
0.7973 - val_loss: 0.4482 - val_accuracy: 0.7596
Epoch 31/50
19/19 [==============================] - 0s 5ms/step - loss: 0.4314 - accuracy:
0.7939 - val_loss: 0.4532 - val_accuracy: 0.7596
Epoch 32/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4352 - accuracy:
0.7922 - val_loss: 0.4597 - val_accuracy: 0.7596
Epoch 33/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4197 - accuracy:
0.8007 - val_loss: 0.4572 - val_accuracy: 0.7500
Epoch 34/50
19/19 [==============================] - 0s 5ms/step - loss: 0.4287 - accuracy:
0.7717 - val_loss: 0.4540 - val_accuracy: 0.7596
Epoch 35/50
19/19 [==============================] - 0s 5ms/step - loss: 0.4252 - accuracy:
0.8041 - val_loss: 0.4592 - val_accuracy: 0.7692
Epoch 36/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4174 - accuracy:
0.7973 - val_loss: 0.4686 - val_accuracy: 0.7692
Epoch 37/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4197 - accuracy:
0.7939 - val_loss: 0.4721 - val_accuracy: 0.7692
Epoch 38/50
19/19 [==============================] - 0s 6ms/step - loss: 0.4187 - accuracy:
0.7939 - val_loss: 0.4596 - val_accuracy: 0.7788
Epoch 39/50
19/19 [==============================] - 0s 5ms/step - loss: 0.4438 - accuracy:
0.7853 - val_loss: 0.4525 - val_accuracy: 0.7692
Epoch 40/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4131 - accuracy:
```

```
0.8228 - val_loss: 0.4500 - val_accuracy: 0.7885
Epoch 41/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4343 - accuracy:
0.7939 - val_loss: 0.4556 - val_accuracy: 0.7788
Epoch 42/50
19/19 [==============================] - 0s 5ms/step - loss: 0.4147 - accuracy:
0.8041 - val_loss: 0.4561 - val_accuracy: 0.7885
Epoch 43/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4073 - accuracy:
0.8228 - val_loss: 0.4639 - val_accuracy: 0.7885
Epoch 44/50
19/19 [==============================] - 0s 5ms/step - loss: 0.4267 - accuracy:
0.7973 - val_loss: 0.4655 - val_accuracy: 0.7885
Epoch 45/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4106 - accuracy:
0.7939 - val_loss: 0.4624 - val_accuracy: 0.7788
Epoch 46/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4089 - accuracy:
0.7956 - val_loss: 0.4639 - val_accuracy: 0.7692
Epoch 47/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4107 - accuracy:
0.8024 - val_loss: 0.4711 - val_accuracy: 0.7596
Epoch 48/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4075 - accuracy:
0.8041 - val_loss: 0.4706 - val_accuracy: 0.7596
Epoch 49/50
19/19 [==============================] - 0s 4ms/step - loss: 0.4087 - accuracy:
0.7939 - val_loss: 0.4705 - val_accuracy: 0.7788
Epoch 50/50
19/19 [==============================] - 0s 4ms/step - loss: 0.3879 - accuracy:
0.8245 - val_loss: 0.4751 - val_accuracy: 0.7788
```

[26]:
```
pyplot.plot(history.history['accuracy'],label='train')
pyplot.plot(history.history['val_accuracy'], label='test')
pyplot.legend()
pyplot.show()
```



[27]:
```
y_pred_train=model.predict(X_train_scale)
y_pred_test=model.predict(X_test_scale)
```

```
19/19 [==============================] - 0s 2ms/step
3/3 [==============================] - 0s 3ms/step
```

[28]:
```
cm = confusion_matrix(y_pred=y_pred_train > 0.5,y_true=y_train)
cm
```

[28]:
```
array([[344,  38],
       [ 67, 138]])
```

[29]:
```
cm = confusion_matrix(y_pred=y_pred_test > 0.5,y_true=y_test)
cm
```

[29]:
```
array([[37, 13],
       [10, 17]])
```

[30]:
```
# Add BachNorm
model=Sequential()
#Addinput
model.add(Input(shape=(X_train_scale.shape[1],)))
```

```python
#Add Layer
model.add(Dense(units=64,activation='relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
#Add Layer
model.add(Dense(units=32,activation='relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
#Add Layer
model.add(Dense(units=16,activation='relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
#output layer
model.add(Dense(units=1,activation='sigmoid'))
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_8 (Dense)             (None, 64)                576

 dropout_3 (Dropout)         (None, 64)                0

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 dense_9 (Dense)             (None, 32)                2080

 dropout_4 (Dropout)         (None, 32)                0

 batch_normalization_1 (Batc  (None, 32)               128
 hNormalization)

 dense_10 (Dense)            (None, 16)                528

 dropout_5 (Dropout)         (None, 16)                0

 batch_normalization_2 (Batc  (None, 16)               64
 hNormalization)

 dense_11 (Dense)            (None, 1)                 17

=================================================================
Total params: 3,649
Trainable params: 3,425
Non-trainable params: 224
```

----

[31]:
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

[32]:
```
history = model.fit(X_train_scale,y_train,validation_data=(X_val_scale,
    y_val),epochs=100,verbose=1)
```

```
Epoch 1/100
19/19 [==============================] - 2s 18ms/step - loss: 0.8384 - accuracy:
0.5026 - val_loss: 0.6843 - val_accuracy: 0.6346
Epoch 2/100
19/19 [==============================] - 0s 5ms/step - loss: 0.7344 - accuracy:
0.5690 - val_loss: 0.6474 - val_accuracy: 0.6923
Epoch 3/100
19/19 [==============================] - 0s 5ms/step - loss: 0.6738 - accuracy:
0.6474 - val_loss: 0.6172 - val_accuracy: 0.7212
Epoch 4/100
19/19 [==============================] - 0s 5ms/step - loss: 0.6013 - accuracy:
0.6746 - val_loss: 0.5902 - val_accuracy: 0.7308
Epoch 5/100
19/19 [==============================] - 0s 5ms/step - loss: 0.5909 - accuracy:
0.6865 - val_loss: 0.5713 - val_accuracy: 0.7308
Epoch 6/100
19/19 [==============================] - 0s 6ms/step - loss: 0.5906 - accuracy:
0.7087 - val_loss: 0.5557 - val_accuracy: 0.7404
Epoch 7/100
19/19 [==============================] - 0s 5ms/step - loss: 0.5709 - accuracy:
0.7223 - val_loss: 0.5489 - val_accuracy: 0.7404
Epoch 8/100
19/19 [==============================] - 0s 6ms/step - loss: 0.5826 - accuracy:
0.6934 - val_loss: 0.5401 - val_accuracy: 0.7308
Epoch 9/100
19/19 [==============================] - 0s 6ms/step - loss: 0.5232 - accuracy:
0.7513 - val_loss: 0.5345 - val_accuracy: 0.7500
Epoch 10/100
19/19 [==============================] - 0s 8ms/step - loss: 0.5203 - accuracy:
0.7359 - val_loss: 0.5202 - val_accuracy: 0.7500
Epoch 11/100
19/19 [==============================] - 0s 8ms/step - loss: 0.5317 - accuracy:
0.7513 - val_loss: 0.5166 - val_accuracy: 0.7500
Epoch 12/100
19/19 [==============================] - 0s 7ms/step - loss: 0.5282 - accuracy:
0.7376 - val_loss: 0.5177 - val_accuracy: 0.7596
Epoch 13/100
19/19 [==============================] - 0s 7ms/step - loss: 0.5000 - accuracy:
0.7615 - val_loss: 0.5223 - val_accuracy: 0.7500
Epoch 14/100
19/19 [==============================] - 0s 7ms/step - loss: 0.5031 - accuracy:
```

```
0.7496 - val_loss: 0.5180 - val_accuracy: 0.7500
Epoch 15/100
19/19 [==============================] - 0s 7ms/step - loss: 0.5074 - accuracy:
0.7530 - val_loss: 0.5117 - val_accuracy: 0.7500
Epoch 16/100
19/19 [==============================] - 0s 7ms/step - loss: 0.4891 - accuracy:
0.7734 - val_loss: 0.5085 - val_accuracy: 0.7500
Epoch 17/100
19/19 [==============================] - 0s 7ms/step - loss: 0.5000 - accuracy:
0.7615 - val_loss: 0.5012 - val_accuracy: 0.7596
Epoch 18/100
19/19 [==============================] - 0s 7ms/step - loss: 0.4881 - accuracy:
0.7325 - val_loss: 0.5008 - val_accuracy: 0.7692
Epoch 19/100
19/19 [==============================] - 0s 7ms/step - loss: 0.4836 - accuracy:
0.7462 - val_loss: 0.5045 - val_accuracy: 0.7692
Epoch 20/100
19/19 [==============================] - 0s 7ms/step - loss: 0.5038 - accuracy:
0.7376 - val_loss: 0.5191 - val_accuracy: 0.7692
Epoch 21/100
19/19 [==============================] - 0s 8ms/step - loss: 0.4962 - accuracy:
0.7513 - val_loss: 0.5107 - val_accuracy: 0.7692
Epoch 22/100
19/19 [==============================] - 0s 8ms/step - loss: 0.4592 - accuracy:
0.7768 - val_loss: 0.5066 - val_accuracy: 0.7692
Epoch 23/100
19/19 [==============================] - 0s 8ms/step - loss: 0.4932 - accuracy:
0.7598 - val_loss: 0.4991 - val_accuracy: 0.7692
Epoch 24/100
19/19 [==============================] - 0s 8ms/step - loss: 0.4455 - accuracy:
0.7973 - val_loss: 0.5029 - val_accuracy: 0.7692
Epoch 25/100
19/19 [==============================] - 0s 7ms/step - loss: 0.4871 - accuracy:
0.7598 - val_loss: 0.5025 - val_accuracy: 0.7692
Epoch 26/100
19/19 [==============================] - 0s 8ms/step - loss: 0.4573 - accuracy:
0.7785 - val_loss: 0.4998 - val_accuracy: 0.7596
Epoch 27/100
19/19 [==============================] - 0s 9ms/step - loss: 0.4640 - accuracy:
0.7700 - val_loss: 0.4998 - val_accuracy: 0.7596
Epoch 28/100
19/19 [==============================] - 0s 6ms/step - loss: 0.4555 - accuracy:
0.7768 - val_loss: 0.4912 - val_accuracy: 0.7596
Epoch 29/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4545 - accuracy:
0.7922 - val_loss: 0.4782 - val_accuracy: 0.7692
Epoch 30/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4612 - accuracy:
```

```
0.7905 - val_loss: 0.4688 - val_accuracy: 0.7692
Epoch 31/100
19/19 [==============================] - 0s 6ms/step - loss: 0.4600 - accuracy:
0.7768 - val_loss: 0.4801 - val_accuracy: 0.7692
Epoch 32/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4818 - accuracy:
0.7632 - val_loss: 0.4728 - val_accuracy: 0.7692
Epoch 33/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4607 - accuracy:
0.7819 - val_loss: 0.4666 - val_accuracy: 0.7788
Epoch 34/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4515 - accuracy:
0.7717 - val_loss: 0.4707 - val_accuracy: 0.7788
Epoch 35/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4459 - accuracy:
0.7990 - val_loss: 0.4655 - val_accuracy: 0.7692
Epoch 36/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4670 - accuracy:
0.7632 - val_loss: 0.4728 - val_accuracy: 0.7788
Epoch 37/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4579 - accuracy:
0.7888 - val_loss: 0.4723 - val_accuracy: 0.7788
Epoch 38/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4499 - accuracy:
0.7683 - val_loss: 0.4763 - val_accuracy: 0.7596
Epoch 39/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4501 - accuracy:
0.7888 - val_loss: 0.4778 - val_accuracy: 0.7596
Epoch 40/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4540 - accuracy:
0.7785 - val_loss: 0.4781 - val_accuracy: 0.7692
Epoch 41/100
19/19 [==============================] - 0s 6ms/step - loss: 0.4507 - accuracy:
0.7836 - val_loss: 0.4716 - val_accuracy: 0.7596
Epoch 42/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4660 - accuracy:
0.7632 - val_loss: 0.4717 - val_accuracy: 0.7596
Epoch 43/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4335 - accuracy:
0.7819 - val_loss: 0.4679 - val_accuracy: 0.7692
Epoch 44/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4364 - accuracy:
0.7802 - val_loss: 0.4690 - val_accuracy: 0.7692
Epoch 45/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4226 - accuracy:
0.7973 - val_loss: 0.4659 - val_accuracy: 0.7788
Epoch 46/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4615 - accuracy:
```

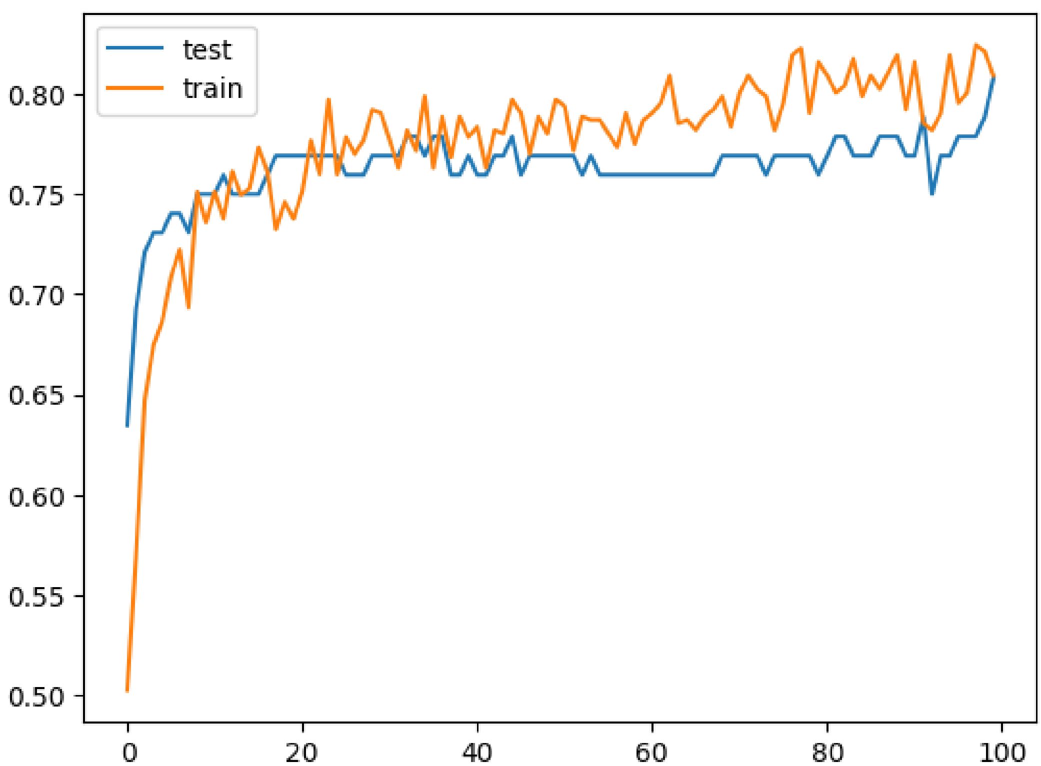
```
0.7905 - val_loss: 0.4730 - val_accuracy: 0.7596
Epoch 47/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4619 - accuracy:
0.7700 - val_loss: 0.4771 - val_accuracy: 0.7692
Epoch 48/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4623 - accuracy:
0.7888 - val_loss: 0.4798 - val_accuracy: 0.7692
Epoch 49/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4434 - accuracy:
0.7802 - val_loss: 0.4775 - val_accuracy: 0.7692
Epoch 50/100
19/19 [==============================] - 0s 6ms/step - loss: 0.4388 - accuracy:
0.7973 - val_loss: 0.4690 - val_accuracy: 0.7692
Epoch 51/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4286 - accuracy:
0.7939 - val_loss: 0.4728 - val_accuracy: 0.7692
Epoch 52/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4600 - accuracy:
0.7717 - val_loss: 0.4741 - val_accuracy: 0.7692
Epoch 53/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4346 - accuracy:
0.7888 - val_loss: 0.4715 - val_accuracy: 0.7596
Epoch 54/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4425 - accuracy:
0.7871 - val_loss: 0.4708 - val_accuracy: 0.7692
Epoch 55/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4562 - accuracy:
0.7871 - val_loss: 0.4691 - val_accuracy: 0.7596
Epoch 56/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4343 - accuracy:
0.7802 - val_loss: 0.4717 - val_accuracy: 0.7596
Epoch 57/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4462 - accuracy:
0.7734 - val_loss: 0.4785 - val_accuracy: 0.7596
Epoch 58/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4200 - accuracy:
0.7905 - val_loss: 0.4725 - val_accuracy: 0.7596
Epoch 59/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4521 - accuracy:
0.7751 - val_loss: 0.4655 - val_accuracy: 0.7596
Epoch 60/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4490 - accuracy:
0.7871 - val_loss: 0.4733 - val_accuracy: 0.7596
Epoch 61/100
19/19 [==============================] - 0s 6ms/step - loss: 0.4226 - accuracy:
0.7905 - val_loss: 0.4743 - val_accuracy: 0.7596
Epoch 62/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4227 - accuracy:
```

```
0.7956 - val_loss: 0.4710 - val_accuracy: 0.7596
Epoch 63/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4351 - accuracy:
0.8092 - val_loss: 0.4750 - val_accuracy: 0.7596
Epoch 64/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4357 - accuracy:
0.7853 - val_loss: 0.4726 - val_accuracy: 0.7596
Epoch 65/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4326 - accuracy:
0.7871 - val_loss: 0.4749 - val_accuracy: 0.7596
Epoch 66/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4409 - accuracy:
0.7819 - val_loss: 0.4791 - val_accuracy: 0.7596
Epoch 67/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4420 - accuracy:
0.7888 - val_loss: 0.4787 - val_accuracy: 0.7596
Epoch 68/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4109 - accuracy:
0.7922 - val_loss: 0.4755 - val_accuracy: 0.7596
Epoch 69/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4198 - accuracy:
0.7990 - val_loss: 0.4731 - val_accuracy: 0.7692
Epoch 70/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4245 - accuracy:
0.7836 - val_loss: 0.4684 - val_accuracy: 0.7692
Epoch 71/100
19/19 [==============================] - 0s 6ms/step - loss: 0.4186 - accuracy:
0.8007 - val_loss: 0.4700 - val_accuracy: 0.7692
Epoch 72/100
19/19 [==============================] - 0s 6ms/step - loss: 0.4176 - accuracy:
0.8092 - val_loss: 0.4716 - val_accuracy: 0.7692
Epoch 73/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4150 - accuracy:
0.8024 - val_loss: 0.4759 - val_accuracy: 0.7692
Epoch 74/100
19/19 [==============================] - 0s 6ms/step - loss: 0.4313 - accuracy:
0.7990 - val_loss: 0.4764 - val_accuracy: 0.7596
Epoch 75/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4037 - accuracy:
0.7819 - val_loss: 0.4743 - val_accuracy: 0.7692
Epoch 76/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4316 - accuracy:
0.7956 - val_loss: 0.4763 - val_accuracy: 0.7692
Epoch 77/100
19/19 [==============================] - 0s 5ms/step - loss: 0.3962 - accuracy:
0.8194 - val_loss: 0.4761 - val_accuracy: 0.7692
Epoch 78/100
19/19 [==============================] - 0s 6ms/step - loss: 0.3993 - accuracy:
```

```
0.8228 - val_loss: 0.4739 - val_accuracy: 0.7692
Epoch 79/100
19/19 [==============================] - 0s 6ms/step - loss: 0.4251 - accuracy:
0.7905 - val_loss: 0.4693 - val_accuracy: 0.7692
Epoch 80/100
19/19 [==============================] - 0s 6ms/step - loss: 0.4034 - accuracy:
0.8160 - val_loss: 0.4730 - val_accuracy: 0.7596
Epoch 81/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4164 - accuracy:
0.8092 - val_loss: 0.4669 - val_accuracy: 0.7692
Epoch 82/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4061 - accuracy:
0.8007 - val_loss: 0.4767 - val_accuracy: 0.7788
Epoch 83/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4076 - accuracy:
0.8041 - val_loss: 0.4754 - val_accuracy: 0.7788
Epoch 84/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3873 - accuracy:
0.8177 - val_loss: 0.4686 - val_accuracy: 0.7692
Epoch 85/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4104 - accuracy:
0.7990 - val_loss: 0.4742 - val_accuracy: 0.7692
Epoch 86/100
19/19 [==============================] - 0s 5ms/step - loss: 0.3946 - accuracy:
0.8092 - val_loss: 0.4830 - val_accuracy: 0.7692
Epoch 87/100
19/19 [==============================] - 0s 6ms/step - loss: 0.3963 - accuracy:
0.8024 - val_loss: 0.4786 - val_accuracy: 0.7788
Epoch 88/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4130 - accuracy:
0.8109 - val_loss: 0.4674 - val_accuracy: 0.7788
Epoch 89/100
19/19 [==============================] - 0s 4ms/step - loss: 0.4136 - accuracy:
0.8194 - val_loss: 0.4628 - val_accuracy: 0.7788
Epoch 90/100
19/19 [==============================] - 0s 5ms/step - loss: 0.3985 - accuracy:
0.7922 - val_loss: 0.4678 - val_accuracy: 0.7692
Epoch 91/100
19/19 [==============================] - 0s 5ms/step - loss: 0.3929 - accuracy:
0.8160 - val_loss: 0.4734 - val_accuracy: 0.7692
Epoch 92/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4011 - accuracy:
0.7853 - val_loss: 0.4805 - val_accuracy: 0.7885
Epoch 93/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4200 - accuracy:
0.7819 - val_loss: 0.4849 - val_accuracy: 0.7500
Epoch 94/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4184 - accuracy:
```

```
0.7905 - val_loss: 0.4755 - val_accuracy: 0.7692
Epoch 95/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4025 - accuracy:
0.8194 - val_loss: 0.4798 - val_accuracy: 0.7692
Epoch 96/100
19/19 [==============================] - 0s 6ms/step - loss: 0.4106 - accuracy:
0.7956 - val_loss: 0.4752 - val_accuracy: 0.7788
Epoch 97/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4200 - accuracy:
0.8007 - val_loss: 0.4816 - val_accuracy: 0.7788
Epoch 98/100
19/19 [==============================] - 0s 5ms/step - loss: 0.4046 - accuracy:
0.8245 - val_loss: 0.4830 - val_accuracy: 0.7788
Epoch 99/100
19/19 [==============================] - 0s 4ms/step - loss: 0.3982 - accuracy:
0.8211 - val_loss: 0.4790 - val_accuracy: 0.7885
Epoch 100/100
19/19 [==============================] - 0s 5ms/step - loss: 0.3927 - accuracy:
0.8092 - val_loss: 0.4731 - val_accuracy: 0.8077
```

[33]:
```
pyplot.plot(history.history['val_accuracy'], label='test')
pyplot.plot(history.history['accuracy'], label='train')
pyplot.legend()
pyplot.show()
```



[34]:
```
y_pred_train=model.predict(X_train_scale)
y_pred_test=model.predict(X_test_scale)
```

```
19/19 [==============================] - 0s 2ms/step
3/3 [==============================] - 0s 4ms/step
```

[35]:
```
cm=confusion_matrix(y_pred_train>0.5,y_train)
cm
```

[35]:
```
array([[353,  64],
       [ 29, 141]])
```

[36]:
```
cm=confusion_matrix(y_pred_test>0.5,y_test)
cm
```

[36]:
```
array([[37, 10],
       [13, 17]])
```

[37]:
```
accuracy_score(y_pred_test>0.5,y_test)
```

[37]: 0.7012987012987013

[38]:
```
accuracy_score(y_pred_train>0.5,y_train)
```

[38]: 0.8415672913117547

[39]:
```
def creat_batchnorm_drop_model():
  # Lets build the Model
    model = Sequential()
    # No of Input will be == (total number of train examples , 8)
    # where 8 = feature
    model.add(Input(shape=(X_train_scale.shape[1],)))

    # Hidden Layer 1
    model.add(Dense(units=64,activation='relu'))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())
    # Hidden Layer 2
    model.add(Dense(units=32,activation='relu'))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())
    # Hidden Layer 3
    model.add(Dense(units=16,activation='relu'))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())
```

```
    # Output Layer - this is a binary classification
    model.add(Dense(units=1,activation='sigmoid'))
    return model
```

[40]:
```
model.summary()
```

```
Model: "sequential_2"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_8 (Dense)             (None, 64)                576

 dropout_3 (Dropout)         (None, 64)                0

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 dense_9 (Dense)             (None, 32)                2080

 dropout_4 (Dropout)         (None, 32)                0

 batch_normalization_1 (Batc  (None, 32)               128
 hNormalization)

 dense_10 (Dense)            (None, 16)                528

 dropout_5 (Dropout)         (None, 16)                0

 batch_normalization_2 (Batc  (None, 16)               64
 hNormalization)

 dense_11 (Dense)            (None, 1)                 17

=================================================================
Total params: 3,649
Trainable params: 3,425
Non-trainable params: 224
_________________________________________________________________
```

[41]:
```
# Check if we have a reason loss value to start withs
m = creat_batchnorm_drop_model()
# Very very low lr and train
m.compile(optimizer=SGD(learning_rate=0.
 ↪00001),loss='binary_crossentropy',metrics=['accuracy'])
history = m.fit(X_train_scale,y_train,validation_data=(X_val_scale,␣
 ↪y_val),epochs=1,verbose=1)
```

```
# Question 1 - Does that accuracy value makes sense ??
# Question 2 - Does that loss value makes sense ??
```

```
19/19 [==============================] - 2s 15ms/step - loss: 0.8847 - accuracy:
0.4787 - val_loss: 0.7042 - val_accuracy: 0.4712
```