layer_eval

May 28, 2020

[2]:

```
import itertools
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# for data scaling and splitting
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
# for neural net
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
# for evaluation
from sklearn.model_selection import KFold, cross_val_score, GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix,␣
↪ConfusionMatrixDisplay
```

[3]:

```
data = pd.read_csv("data/combined_expression.csv")
data.head()
```

[3]:

```
   CELL_LINE_NAME  cluster    TSPAN6      TNMD       DPM1     SCYL3  C1orf112  \
0         1240123        2  8.319417  3.111183   9.643558  4.757258  3.919757
1         1240131        1  7.611268  2.704739  10.276079  3.650299  3.481567
2         1240132        1  7.678658  2.845781  10.180954  3.573048  3.431235
3         1240134        1  3.265063  3.063746  10.490285  3.340791  3.676912
4         1240140        1  7.090138  2.988043  10.264692  4.119555  3.432585

        FGR       CFH     FUCA2  …   C6orf10   TMEM225    NOTCH4      PBX2  \
0  3.602185  3.329644  9.076950  …  3.085394  3.462811  3.339030  4.614897
1  3.145538  3.565127  7.861068  …  2.801456  2.985889  3.180068  5.415729
2  3.090781  4.116643  8.121190  …  2.934962  2.952937  3.164655  5.707506
3  3.512821  3.873922  8.790851  …  3.041839  3.398847  3.106710  5.773963
4  3.308033  3.318371  6.927761  …  3.028787  3.225982  3.275820  5.334283

       AGER      RNF5    AGPAT1    DFNB59     PRRT1     FKBPL
0  3.395845  3.419193  3.971646  3.729310  3.320022  6.447316
1  3.299858  3.028414  3.877889  3.911516  3.379405  4.729557
```

```
2  3.434295  2.961345  4.272194  3.085696  3.002557  5.653588
3  3.412641  3.136110  4.422262  3.522122  3.509437  5.953242
4  3.864678  3.259242  3.840581  5.809553  3.674587  5.577503

[5 rows x 16384 columns]
```

[4]:
```
data['cluster'].replace([1, 2],[0, 1],inplace=True)
data.shape
```

[4]: (541, 16384)

[5]:
```
selected_genes = pd.read_csv('cleaned/boruta.csv')
selected_genes = selected_genes.values.tolist()
selected_genes = list(itertools.chain(*selected_genes))
```

[36]:
```
# retrieving proper columns
X = data.loc[:, selected_genes]
y = data['cluster'].values
# scaling the data
scalar = MinMaxScaler()
x_scaled = scalar.fit_transform(X)
# splitting data (20% test, 80% train)
X_train, X_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.2)
sm = SMOTE()
X_train, y_train = sm.fit_sample(X_train, y_train)
```

1 Confusion Matrix Plotting Function

[137]:
```
def plot_confusion_matrix(cm, target_names, title='Confusion matrix',␣
 ↪cmap=None, normalize=True):
    import matplotlib.pyplot as plt
    import numpy as np
    import itertools

    accuracy = np.trace(cm) / np.sum(cm).astype('float')
    misclass = 1 - accuracy

    if cmap is None:
        cmap = plt.get_cmap('Blues')

    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

    if target_names is not None:
```

```
        tick_marks = np.arange(len(target_names))
        plt.xticks(tick_marks, target_names)
        plt.yticks(tick_marks, target_names)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]


    thresh = cm.max() / 1.5 if normalize else cm.max() / 2
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        if normalize:
            plt.text(j, i, "{:0.4f}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")
        else:
            plt.text(j, i, "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")


    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label\naccuracy={:0.4f}; misclass={:0.4f}'.
↪format(accuracy, misclass))
    plt.show()
```

2 5 Hidden Layers

[43]:
```
def hidden5(optimizer='adam', init='normal', dropout=0.3):
    model = Sequential()
    # adding layers and adding droplayers to avoid overfitting
    hidden_layers = len(selected_genes)

    model.add(Dense(hidden_layers*2, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(dropout))

    model.add(Dense(hidden_layers*4, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(dropout))

    model.add(Dense(hidden_layers*4, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(dropout))

    model.add(Dense(hidden_layers*2, activation='relu'))
```

```
    model.add(BatchNormalization())
    model.add(Dropout(dropout))

    model.add(Dense(hidden_layers, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(dropout))

    model.add(Dense(1, activation='sigmoid'))
    # compiling
    model.compile(optimizer=optimizer, loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
    return model
```

[79]:
```
# parameters selected from previous gridsearch
model5 = KerasClassifier(build_fn=hidden5, epochs=50, batch_size=16,␣
 ↪optimizer='adagrad',init='normal')
# kfold = KFold(n_splits=3, shuffle=True)
# results = cross_val_score(model, X_train, y_train, cv=kfold)
# print("Baseline Accuracy: %.2f%% (%.2f%%)" % (results.mean()*100, results.
 ↪std()*100))
```

[80]:
```
history5 = model5.fit(X_train, y_train, validation_data=(X_test,y_test),␣
 ↪shuffle=True)
y_pred5 = model5.predict(X_test)
```

```
Train on 556 samples, validate on 109 samples
Epoch 1/50
556/556 [==============================] - 5s 8ms/sample - loss: 0.5551 -
accuracy: 0.7824 - val_loss: 0.5859 - val_accuracy: 0.8899
Epoch 2/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.4121 -
accuracy: 0.8327 - val_loss: 0.5702 - val_accuracy: 0.8532
Epoch 3/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.4177 -
accuracy: 0.8291 - val_loss: 0.5915 - val_accuracy: 0.7156
Epoch 4/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.2938 -
accuracy: 0.8849 - val_loss: 0.6436 - val_accuracy: 0.5688
Epoch 5/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.3202 -
accuracy: 0.8831 - val_loss: 0.5867 - val_accuracy: 0.6697
Epoch 6/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.3200 -
accuracy: 0.8687 - val_loss: 0.5760 - val_accuracy: 0.6972
Epoch 7/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.2751 -
accuracy: 0.8903 - val_loss: 0.6061 - val_accuracy: 0.6330
```

```
Epoch 8/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.3060 -
accuracy: 0.8705 - val_loss: 0.5396 - val_accuracy: 0.6789
Epoch 9/50
556/556 [==============================] - 2s 3ms/sample - loss: 0.2680 -
accuracy: 0.8921 - val_loss: 0.4676 - val_accuracy: 0.7431
Epoch 10/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.2277 -
accuracy: 0.9029 - val_loss: 0.4617 - val_accuracy: 0.7523
Epoch 11/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.2680 -
accuracy: 0.8957 - val_loss: 0.4033 - val_accuracy: 0.8257
Epoch 12/50
556/556 [==============================] - 2s 3ms/sample - loss: 0.2088 -
accuracy: 0.9227 - val_loss: 0.3181 - val_accuracy: 0.8716
Epoch 13/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.2652 -
accuracy: 0.8975 - val_loss: 0.3068 - val_accuracy: 0.8899
Epoch 14/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.2081 -
accuracy: 0.9317 - val_loss: 0.2863 - val_accuracy: 0.8807
Epoch 15/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.2283 -
accuracy: 0.9101 - val_loss: 0.2882 - val_accuracy: 0.8991
Epoch 16/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.2202 -
accuracy: 0.9101 - val_loss: 0.2939 - val_accuracy: 0.8991
Epoch 17/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.2051 -
accuracy: 0.9227 - val_loss: 0.3176 - val_accuracy: 0.8807
Epoch 18/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.2307 -
accuracy: 0.9101 - val_loss: 0.3264 - val_accuracy: 0.8899
Epoch 19/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1771 -
accuracy: 0.9442 - val_loss: 0.3160 - val_accuracy: 0.8899
Epoch 20/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1898 -
accuracy: 0.9227 - val_loss: 0.2912 - val_accuracy: 0.8991
Epoch 21/50
556/556 [==============================] - 2s 3ms/sample - loss: 0.2020 -
accuracy: 0.9119 - val_loss: 0.3101 - val_accuracy: 0.8807
Epoch 22/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1544 -
accuracy: 0.9371 - val_loss: 0.3137 - val_accuracy: 0.8624
Epoch 23/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1874 -
accuracy: 0.9424 - val_loss: 0.3148 - val_accuracy: 0.8807
```

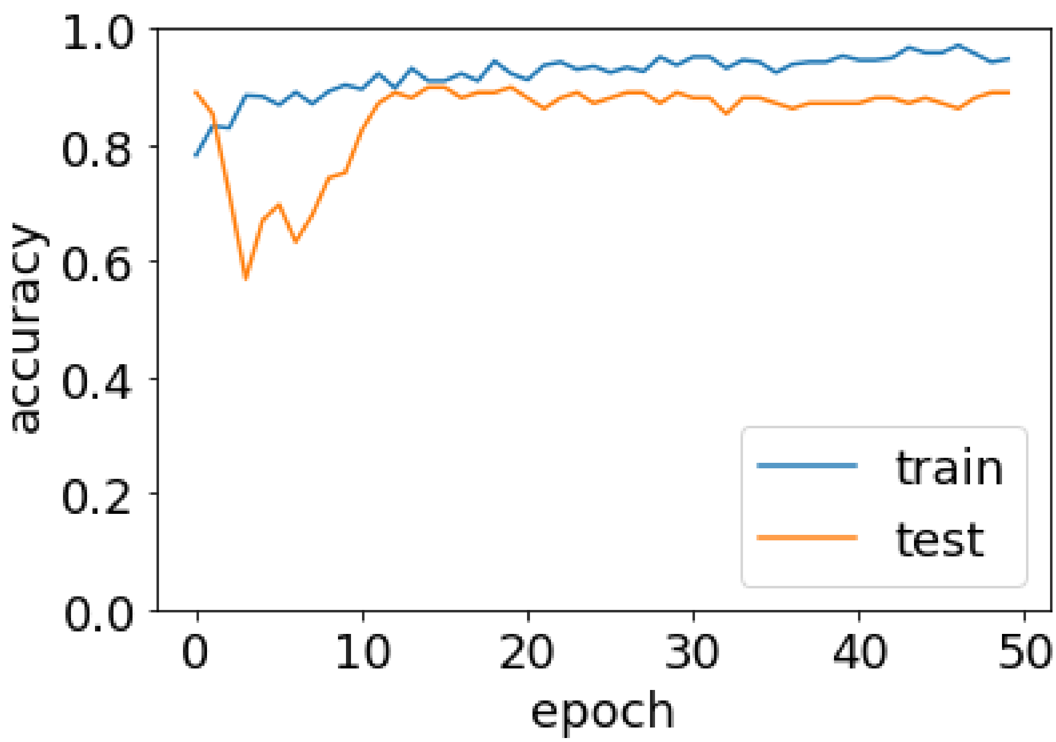
```
Epoch 24/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1895 -
accuracy: 0.9299 - val_loss: 0.3406 - val_accuracy: 0.8899
Epoch 25/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1603 -
accuracy: 0.9353 - val_loss: 0.3664 - val_accuracy: 0.8716
Epoch 26/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1947 -
accuracy: 0.9245 - val_loss: 0.3603 - val_accuracy: 0.8807
Epoch 27/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1812 -
accuracy: 0.9335 - val_loss: 0.3598 - val_accuracy: 0.8899
Epoch 28/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1671 -
accuracy: 0.9263 - val_loss: 0.3647 - val_accuracy: 0.8899
Epoch 29/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1287 -
accuracy: 0.9514 - val_loss: 0.3838 - val_accuracy: 0.8716
Epoch 30/50
556/556 [==============================] - 2s 3ms/sample - loss: 0.1568 -
accuracy: 0.9371 - val_loss: 0.3414 - val_accuracy: 0.8899
Epoch 31/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1375 -
accuracy: 0.9514 - val_loss: 0.3297 - val_accuracy: 0.8807
Epoch 32/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1256 -
accuracy: 0.9514 - val_loss: 0.3269 - val_accuracy: 0.8807
Epoch 33/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1708 -
accuracy: 0.9317 - val_loss: 0.3972 - val_accuracy: 0.8532
Epoch 34/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1463 -
accuracy: 0.9460 - val_loss: 0.3237 - val_accuracy: 0.8807
Epoch 35/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1419 -
accuracy: 0.9424 - val_loss: 0.3439 - val_accuracy: 0.8807
Epoch 36/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1699 -
accuracy: 0.9245 - val_loss: 0.3368 - val_accuracy: 0.8716
Epoch 37/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1378 -
accuracy: 0.9388 - val_loss: 0.3379 - val_accuracy: 0.8624
Epoch 38/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1381 -
accuracy: 0.9424 - val_loss: 0.3458 - val_accuracy: 0.8716
Epoch 39/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1508 -
accuracy: 0.9424 - val_loss: 0.3743 - val_accuracy: 0.8716
```

```
Epoch 40/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1434 -
accuracy: 0.9532 - val_loss: 0.4046 - val_accuracy: 0.8716
Epoch 41/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1395 -
accuracy: 0.9460 - val_loss: 0.3786 - val_accuracy: 0.8716
Epoch 42/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1662 -
accuracy: 0.9460 - val_loss: 0.3588 - val_accuracy: 0.8807
Epoch 43/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1298 -
accuracy: 0.9496 - val_loss: 0.3533 - val_accuracy: 0.8807
Epoch 44/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.0996 -
accuracy: 0.9676 - val_loss: 0.3598 - val_accuracy: 0.8716
Epoch 45/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1295 -
accuracy: 0.9586 - val_loss: 0.3634 - val_accuracy: 0.8807
Epoch 46/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1334 -
accuracy: 0.9586 - val_loss: 0.3504 - val_accuracy: 0.8716
Epoch 47/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.0999 -
accuracy: 0.9712 - val_loss: 0.3657 - val_accuracy: 0.8624
Epoch 48/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1073 -
accuracy: 0.9568 - val_loss: 0.3662 - val_accuracy: 0.8807
Epoch 49/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1526 -
accuracy: 0.9424 - val_loss: 0.3642 - val_accuracy: 0.8899
Epoch 50/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.1505 -
accuracy: 0.9478 - val_loss: 0.3792 - val_accuracy: 0.8899
```

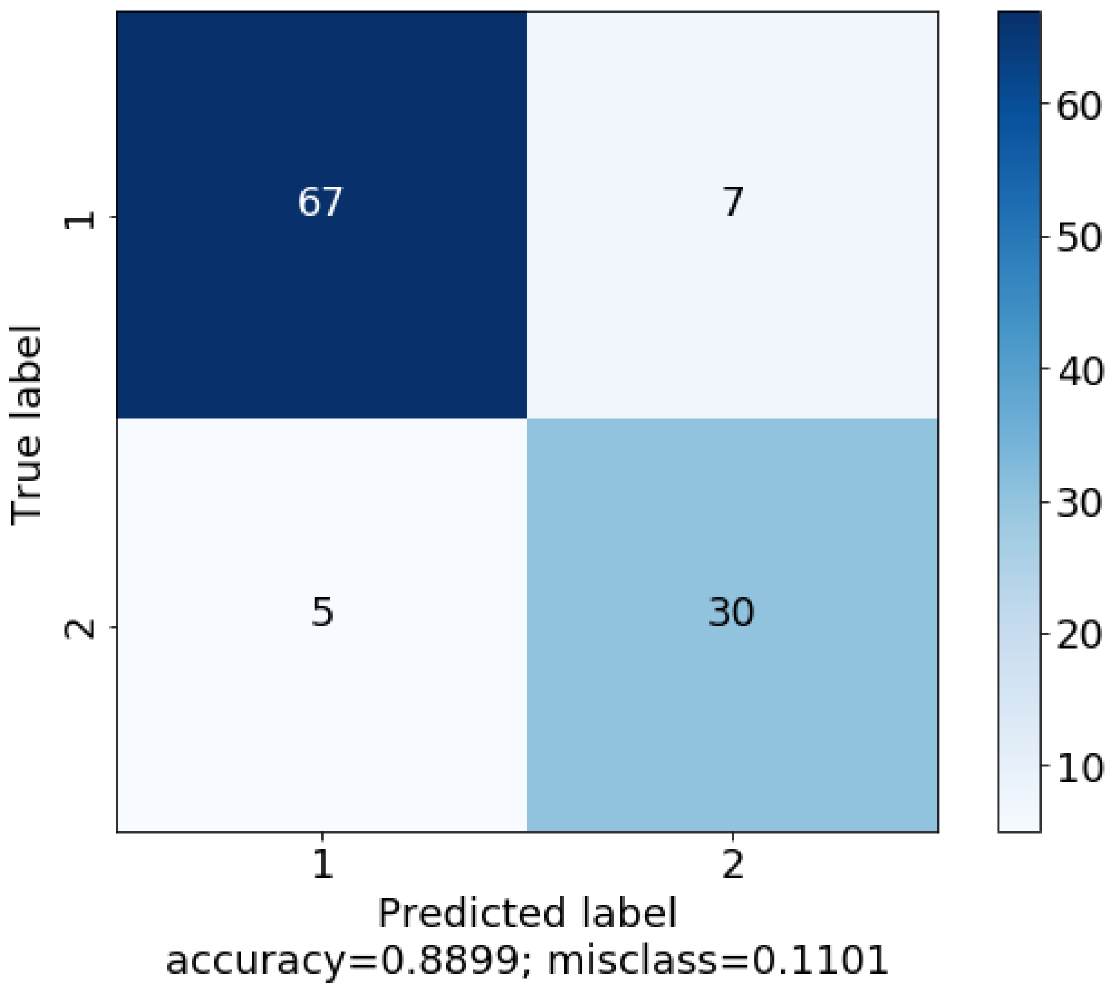
[81]:
```
print(classification_report(y_test, y_pred5))
```

```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92        74
           1       0.81      0.86      0.83        35

    accuracy                           0.89       109
   macro avg       0.87      0.88      0.88       109
weighted avg       0.89      0.89      0.89       109
```

[82]:
```
plt.plot(history5.history['accuracy'])
plt.plot(history5.history['val_accuracy'])
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='lower right')
plt.ylim(0, 1)
plt.show()
```



[136]:
```
cm = confusion_matrix(y_test, y_pred5)
plt.rcParams.update({'font.size': 18})
plot_confusion_matrix(cm, ['1', '2'], title='', normalize=False)
```



[130]:
```
model5.model.save('models/hidden5.h5')
```

3 4 Hidden Layers

[110]:
```
def hidden4(optimizer='adam', init='normal', dropout=0.3):
    model = Sequential()
    # adding layers and adding droplayers to avoid overfitting
    hidden_layers = len(selected_genes)

    model.add(Dense(hidden_layers*2, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(dropout))

    model.add(Dense(hidden_layers*4, activation='relu'))
    model.add(BatchNormalization())
```

```
    model.add(Dropout(dropout))

    model.add(Dense(hidden_layers*4, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(dropout))

    model.add(Dense(hidden_layers*2, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(dropout))

    model.add(Dense(1, activation='sigmoid'))
    # compiling
    model.compile(optimizer=optimizer, loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
    return model
```

[111]:
```
# parameters selected from previous gridsearch
model4 = KerasClassifier(build_fn=hidden4, epochs=50, batch_size=32,␣
 ↪optimizer='adagrad',init='normal')
# kfold = KFold(n_splits=3, shuffle=True)
# results = cross_val_score(model, X_train, y_train, cv=kfold)
# print("Baseline Accuracy: %.2f%% (%.2f%%)" % (results.mean()*100, results.
 ↪std()*100))
```

[112]:
```
history4 = model4.fit(X_train, y_train, validation_data=(X_test,y_test),␣
 ↪shuffle=True)
y_pred4 = model4.predict(X_test)
```

```
Train on 556 samples, validate on 109 samples
Epoch 1/50
556/556 [==============================] - 3s 6ms/sample - loss: 0.5264 -
accuracy: 0.7968 - val_loss: 0.6035 - val_accuracy: 0.7890
Epoch 2/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.2955 -
accuracy: 0.8795 - val_loss: 0.6137 - val_accuracy: 0.7982
Epoch 3/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.2730 -
accuracy: 0.8885 - val_loss: 0.7158 - val_accuracy: 0.4679
Epoch 4/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.2088 -
accuracy: 0.9083 - val_loss: 0.6617 - val_accuracy: 0.6147
Epoch 5/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.1910 -
accuracy: 0.9317 - val_loss: 0.6629 - val_accuracy: 0.5963
Epoch 6/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.1901 -
accuracy: 0.9388 - val_loss: 0.5643 - val_accuracy: 0.7706
```

```
Epoch 7/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.1643 -
accuracy: 0.9353 - val_loss: 0.5797 - val_accuracy: 0.7156
Epoch 8/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.1442 -
accuracy: 0.9442 - val_loss: 0.5747 - val_accuracy: 0.7064
Epoch 9/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.1618 -
accuracy: 0.9388 - val_loss: 0.6764 - val_accuracy: 0.5321
Epoch 10/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.1204 -
accuracy: 0.9622 - val_loss: 0.6647 - val_accuracy: 0.5413
Epoch 11/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.1243 -
accuracy: 0.9532 - val_loss: 0.6100 - val_accuracy: 0.6514
Epoch 12/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.1205 -
accuracy: 0.9478 - val_loss: 0.6543 - val_accuracy: 0.6147
Epoch 13/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0962 -
accuracy: 0.9658 - val_loss: 0.4594 - val_accuracy: 0.7706
Epoch 14/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.1073 -
accuracy: 0.9568 - val_loss: 0.5773 - val_accuracy: 0.6972
Epoch 15/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0899 -
accuracy: 0.9658 - val_loss: 0.4324 - val_accuracy: 0.7798
Epoch 16/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.1058 -
accuracy: 0.9604 - val_loss: 0.6464 - val_accuracy: 0.6881
Epoch 17/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0806 -
accuracy: 0.9712 - val_loss: 0.4995 - val_accuracy: 0.7798
Epoch 18/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0917 -
accuracy: 0.9712 - val_loss: 0.4531 - val_accuracy: 0.7890
Epoch 19/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0671 -
accuracy: 0.9802 - val_loss: 0.3946 - val_accuracy: 0.7982
Epoch 20/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0660 -
accuracy: 0.9784 - val_loss: 0.3714 - val_accuracy: 0.7982
Epoch 21/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0736 -
accuracy: 0.9748 - val_loss: 0.3697 - val_accuracy: 0.7798
Epoch 22/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0648 -
accuracy: 0.9784 - val_loss: 0.3811 - val_accuracy: 0.7706
```

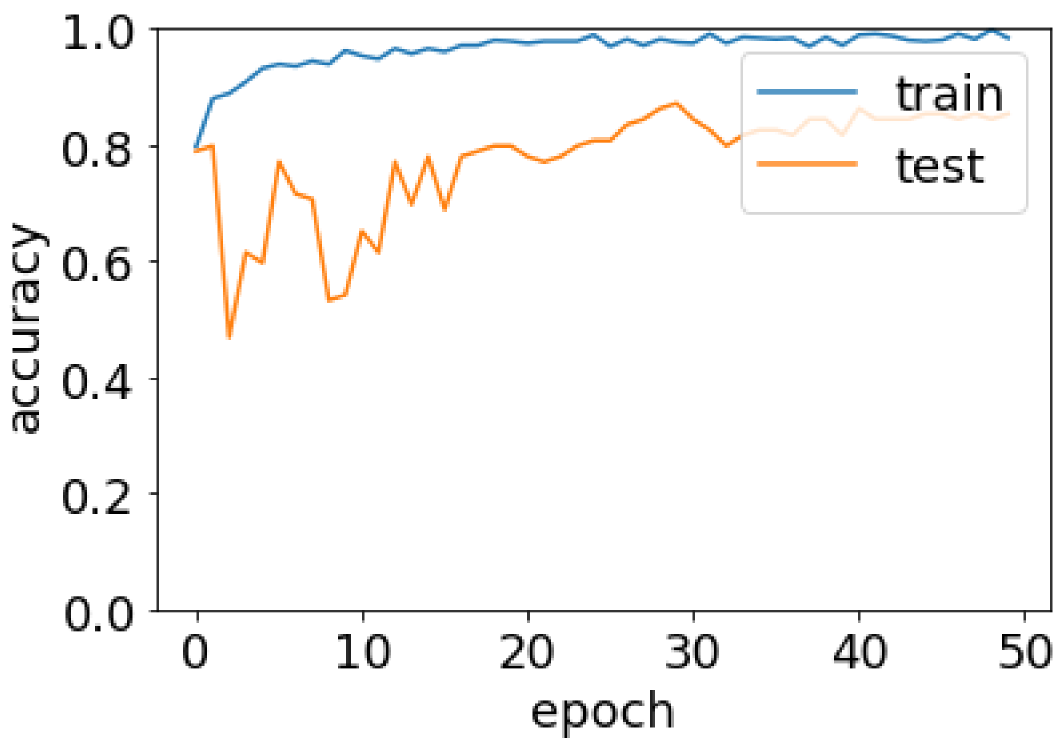
```
Epoch 23/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0653 -
accuracy: 0.9784 - val_loss: 0.3747 - val_accuracy: 0.7798
Epoch 24/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0640 -
accuracy: 0.9784 - val_loss: 0.3839 - val_accuracy: 0.7982
Epoch 25/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0425 -
accuracy: 0.9892 - val_loss: 0.3826 - val_accuracy: 0.8073
Epoch 26/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0828 -
accuracy: 0.9694 - val_loss: 0.4437 - val_accuracy: 0.8073
Epoch 27/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0566 -
accuracy: 0.9820 - val_loss: 0.3379 - val_accuracy: 0.8349
Epoch 28/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0704 -
accuracy: 0.9712 - val_loss: 0.3381 - val_accuracy: 0.8440
Epoch 29/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0502 -
accuracy: 0.9820 - val_loss: 0.3562 - val_accuracy: 0.8624
Epoch 30/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0545 -
accuracy: 0.9766 - val_loss: 0.3405 - val_accuracy: 0.8716
Epoch 31/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0671 -
accuracy: 0.9748 - val_loss: 0.3727 - val_accuracy: 0.8440
Epoch 32/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0370 -
accuracy: 0.9910 - val_loss: 0.4084 - val_accuracy: 0.8257
Epoch 33/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0572 -
accuracy: 0.9748 - val_loss: 0.4727 - val_accuracy: 0.7982
Epoch 34/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0448 -
accuracy: 0.9856 - val_loss: 0.4305 - val_accuracy: 0.8165
Epoch 35/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0479 -
accuracy: 0.9838 - val_loss: 0.4392 - val_accuracy: 0.8257
Epoch 36/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0547 -
accuracy: 0.9820 - val_loss: 0.4335 - val_accuracy: 0.8257
Epoch 37/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0547 -
accuracy: 0.9838 - val_loss: 0.3974 - val_accuracy: 0.8165
Epoch 38/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0687 -
accuracy: 0.9694 - val_loss: 0.3990 - val_accuracy: 0.8440
```

```
Epoch 39/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0461 -
accuracy: 0.9856 - val_loss: 0.3942 - val_accuracy: 0.8440
Epoch 40/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0577 -
accuracy: 0.9712 - val_loss: 0.4376 - val_accuracy: 0.8165
Epoch 41/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0282 -
accuracy: 0.9892 - val_loss: 0.3965 - val_accuracy: 0.8624
Epoch 42/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0297 -
accuracy: 0.9910 - val_loss: 0.3760 - val_accuracy: 0.8440
Epoch 43/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0319 -
accuracy: 0.9874 - val_loss: 0.3898 - val_accuracy: 0.8440
Epoch 44/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0481 -
accuracy: 0.9802 - val_loss: 0.3970 - val_accuracy: 0.8440
Epoch 45/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0580 -
accuracy: 0.9784 - val_loss: 0.4293 - val_accuracy: 0.8532
Epoch 46/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0426 -
accuracy: 0.9802 - val_loss: 0.3951 - val_accuracy: 0.8532
Epoch 47/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0336 -
accuracy: 0.9910 - val_loss: 0.4208 - val_accuracy: 0.8440
Epoch 48/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0435 -
accuracy: 0.9820 - val_loss: 0.4200 - val_accuracy: 0.8532
Epoch 49/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0220 -
accuracy: 0.9982 - val_loss: 0.4248 - val_accuracy: 0.8440
Epoch 50/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.0403 -
accuracy: 0.9838 - val_loss: 0.4302 - val_accuracy: 0.8532
```

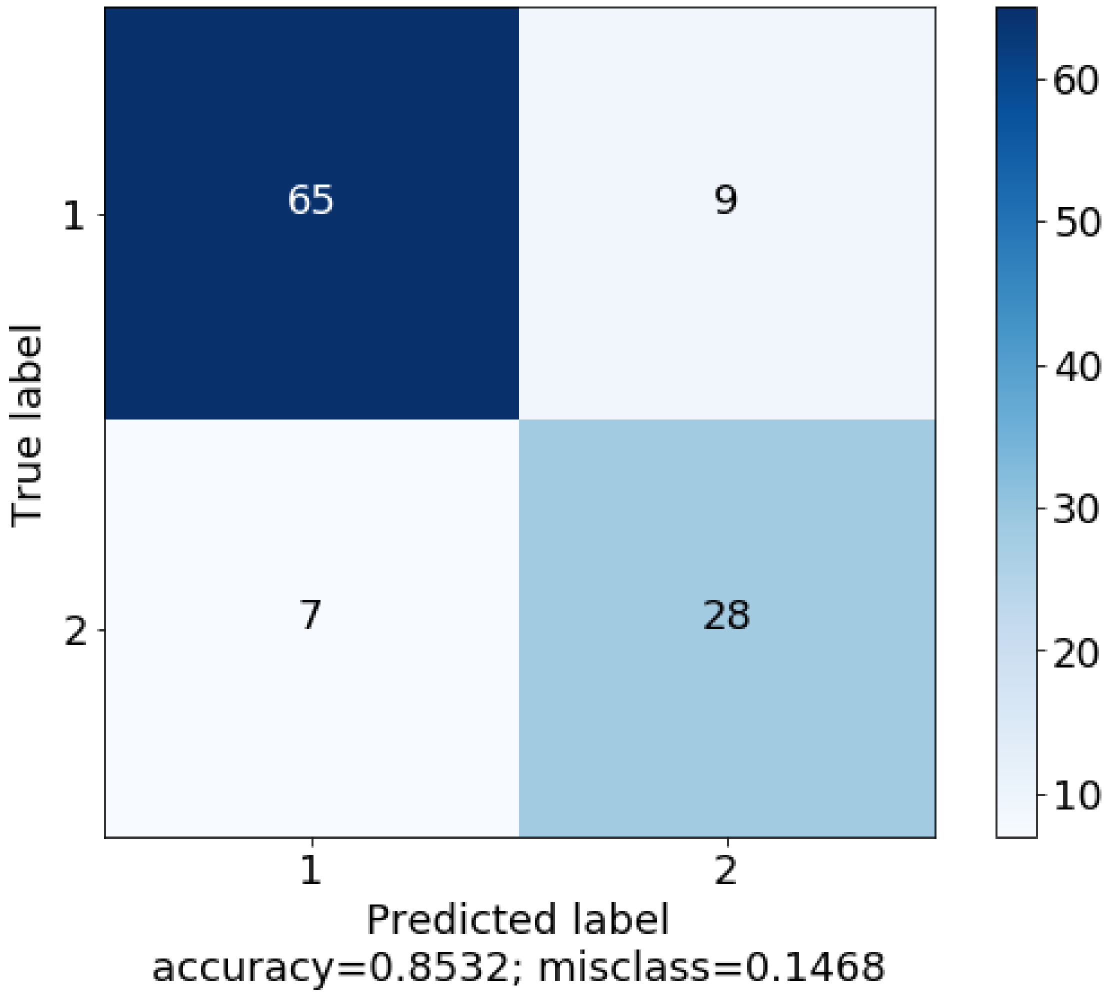
[113]:
```
print(classification_report(y_test, y_pred4))
```

```
              precision    recall  f1-score   support

           0       0.90      0.88      0.89        74
           1       0.76      0.80      0.78        35

    accuracy                           0.85       109
   macro avg       0.83      0.84      0.83       109
weighted avg       0.86      0.85      0.85       109
```

[114]:
```
plt.plot(history4.history['accuracy'])
plt.plot(history4.history['val_accuracy'])
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.ylim(0, 1)
plt.show()
```



[115]:
```
cm = confusion_matrix(y_test, y_pred4)
plt.rcParams.update({'font.size': 18})
plot_confusion_matrix(cm, ['1', '2'], title='', normalize=False)
```



[131]:
```
model4.model.save('models/hidden4.h5')
```

4 3 Hidden Layers

[96]:
```
def hidden3(optimizer='rmsprop',init='glorot_uniform', dropout=0.3):
    model = Sequential()
    # adding layers and adding droplayers to avoid overfitting
    hidden_layers = len(selected_genes)
    model.add(Dense(hidden_layers*2, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(dropout))

    model.add(Dense(hidden_layers*4, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(dropout))
```

```
    model.add(Dense(hidden_layers*2, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(dropout))

    model.add(Dense(1, activation='sigmoid'))
    # compiling
    model.compile(optimizer=optimizer, loss='binary_crossentropy',␣
↪metrics=['accuracy'])
    return model
```

[103]:
```
# parameters selected from previous gridsearch
model3 = KerasClassifier(build_fn=hidden3, epochs=50, batch_size=32,␣
↪optimizer='adagrad', init='normal')
# kfold = KFold(n_splits=3, shuffle=True)
# results = cross_val_score(model, X_train, y_train, cv=kfold)
# print("Baseline Accuracy: %.2f%% (%.2f%%)" % (results.mean()*100, results.
↪std()*100))
```

[104]:
```
history3 = model3.fit(X_train, y_train, validation_data=(X_test, y_test),␣
↪shuffle=True)
y_pred3 = model3.predict(X_test)
```

```
Train on 556 samples, validate on 109 samples
Epoch 1/50
556/556 [==============================] - 3s 6ms/sample - loss: 0.5827 -
accuracy: 0.7806 - val_loss: 0.5764 - val_accuracy: 0.8440
Epoch 2/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.3243 -
accuracy: 0.8759 - val_loss: 0.6116 - val_accuracy: 0.6789
Epoch 3/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.2702 -
accuracy: 0.8867 - val_loss: 0.5972 - val_accuracy: 0.6789
Epoch 4/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.2768 -
accuracy: 0.8813 - val_loss: 0.6440 - val_accuracy: 0.5413
Epoch 5/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.2195 -
accuracy: 0.9101 - val_loss: 0.5650 - val_accuracy: 0.6972
Epoch 6/50
556/556 [==============================] - 1s 1ms/sample - loss: 0.1741 -
accuracy: 0.9299 - val_loss: 0.5475 - val_accuracy: 0.6789
Epoch 7/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.1594 -
accuracy: 0.9442 - val_loss: 0.5600 - val_accuracy: 0.6789
Epoch 8/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.1899 -
```

```
accuracy: 0.9335 - val_loss: 0.5149 - val_accuracy: 0.7064
Epoch 9/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.2236 -
accuracy: 0.9101 - val_loss: 0.5061 - val_accuracy: 0.6972
Epoch 10/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.2209 -
accuracy: 0.9083 - val_loss: 0.4163 - val_accuracy: 0.8073
Epoch 11/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.1829 -
accuracy: 0.9353 - val_loss: 0.3970 - val_accuracy: 0.8073
Epoch 12/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.1701 -
accuracy: 0.9406 - val_loss: 0.5102 - val_accuracy: 0.7248
Epoch 13/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.1321 -
accuracy: 0.9478 - val_loss: 0.3607 - val_accuracy: 0.8165
Epoch 14/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.1095 -
accuracy: 0.9676 - val_loss: 0.4806 - val_accuracy: 0.7431
Epoch 15/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.1167 -
accuracy: 0.9496 - val_loss: 0.3851 - val_accuracy: 0.7982
Epoch 16/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.1364 -
accuracy: 0.9550 - val_loss: 0.4020 - val_accuracy: 0.8073
Epoch 17/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.1039 -
accuracy: 0.9712 - val_loss: 0.3482 - val_accuracy: 0.8624
Epoch 18/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.1157 -
accuracy: 0.9586 - val_loss: 0.3597 - val_accuracy: 0.8624
Epoch 19/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.1073 -
accuracy: 0.9622 - val_loss: 0.3499 - val_accuracy: 0.8624
Epoch 20/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0483 -
accuracy: 0.9874 - val_loss: 0.3443 - val_accuracy: 0.8624
Epoch 21/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0875 -
accuracy: 0.9712 - val_loss: 0.3603 - val_accuracy: 0.8716
Epoch 22/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.1280 -
accuracy: 0.9550 - val_loss: 0.3767 - val_accuracy: 0.8624
Epoch 23/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.1218 -
accuracy: 0.9622 - val_loss: 0.3613 - val_accuracy: 0.8716
Epoch 24/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0838 -
```

```
accuracy: 0.9730 - val_loss: 0.3671 - val_accuracy: 0.8807
Epoch 25/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0711 -
accuracy: 0.9730 - val_loss: 0.3834 - val_accuracy: 0.8624
Epoch 26/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0727 -
accuracy: 0.9838 - val_loss: 0.3923 - val_accuracy: 0.8624
Epoch 27/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.1033 -
accuracy: 0.9622 - val_loss: 0.4050 - val_accuracy: 0.8624
Epoch 28/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0924 -
accuracy: 0.9676 - val_loss: 0.3892 - val_accuracy: 0.8624
Epoch 29/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0878 -
accuracy: 0.9586 - val_loss: 0.3901 - val_accuracy: 0.8624
Epoch 30/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0910 -
accuracy: 0.9658 - val_loss: 0.3949 - val_accuracy: 0.8624
Epoch 31/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0607 -
accuracy: 0.9838 - val_loss: 0.3728 - val_accuracy: 0.8624
Epoch 32/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0849 -
accuracy: 0.9676 - val_loss: 0.3907 - val_accuracy: 0.8624
Epoch 33/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0859 -
accuracy: 0.9694 - val_loss: 0.4096 - val_accuracy: 0.8624
Epoch 34/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0504 -
accuracy: 0.9892 - val_loss: 0.4138 - val_accuracy: 0.8624
Epoch 35/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0775 -
accuracy: 0.9766 - val_loss: 0.4173 - val_accuracy: 0.8440
Epoch 36/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0508 -
accuracy: 0.9784 - val_loss: 0.3965 - val_accuracy: 0.8624
Epoch 37/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0627 -
accuracy: 0.9802 - val_loss: 0.4081 - val_accuracy: 0.8532
Epoch 38/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0656 -
accuracy: 0.9802 - val_loss: 0.4367 - val_accuracy: 0.8440
Epoch 39/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0512 -
accuracy: 0.9838 - val_loss: 0.4281 - val_accuracy: 0.8624
Epoch 40/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0671 -
```

```
accuracy: 0.9802 - val_loss: 0.4457 - val_accuracy: 0.8349
Epoch 41/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0804 -
accuracy: 0.9730 - val_loss: 0.4543 - val_accuracy: 0.8440
Epoch 42/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0476 -
accuracy: 0.9874 - val_loss: 0.4158 - val_accuracy: 0.8624
Epoch 43/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0711 -
accuracy: 0.9748 - val_loss: 0.4333 - val_accuracy: 0.8532
Epoch 44/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0647 -
accuracy: 0.9838 - val_loss: 0.4326 - val_accuracy: 0.8532
Epoch 45/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0502 -
accuracy: 0.9802 - val_loss: 0.4248 - val_accuracy: 0.8624
Epoch 46/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0538 -
accuracy: 0.9892 - val_loss: 0.4243 - val_accuracy: 0.8624
Epoch 47/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0527 -
accuracy: 0.9856 - val_loss: 0.4273 - val_accuracy: 0.8532
Epoch 48/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0920 -
accuracy: 0.9640 - val_loss: 0.4205 - val_accuracy: 0.8532
Epoch 49/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0681 -
accuracy: 0.9820 - val_loss: 0.4338 - val_accuracy: 0.8532
Epoch 50/50
556/556 [==============================] - 1s 2ms/sample - loss: 0.0843 -
accuracy: 0.9676 - val_loss: 0.4269 - val_accuracy: 0.8624
```

[105]:

```
print(classification_report(y_test, y_pred3))
```

```
              precision    recall  f1-score   support

           0       0.89      0.91      0.90        74
           1       0.79      0.77      0.78        35

    accuracy                           0.86       109
   macro avg       0.84      0.84      0.84       109
weighted avg       0.86      0.86      0.86       109
```

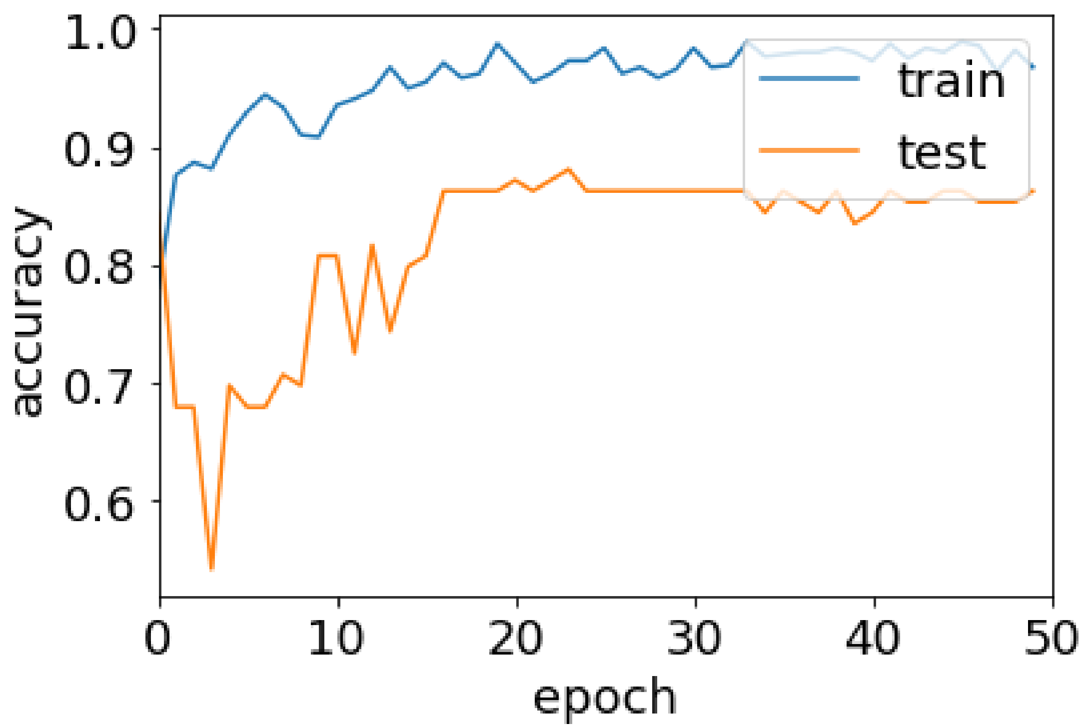
[106]:

```
print(confusion_matrix(y_test, y_pred3))
```

```
[[67  7]
 [ 8 27]]
```

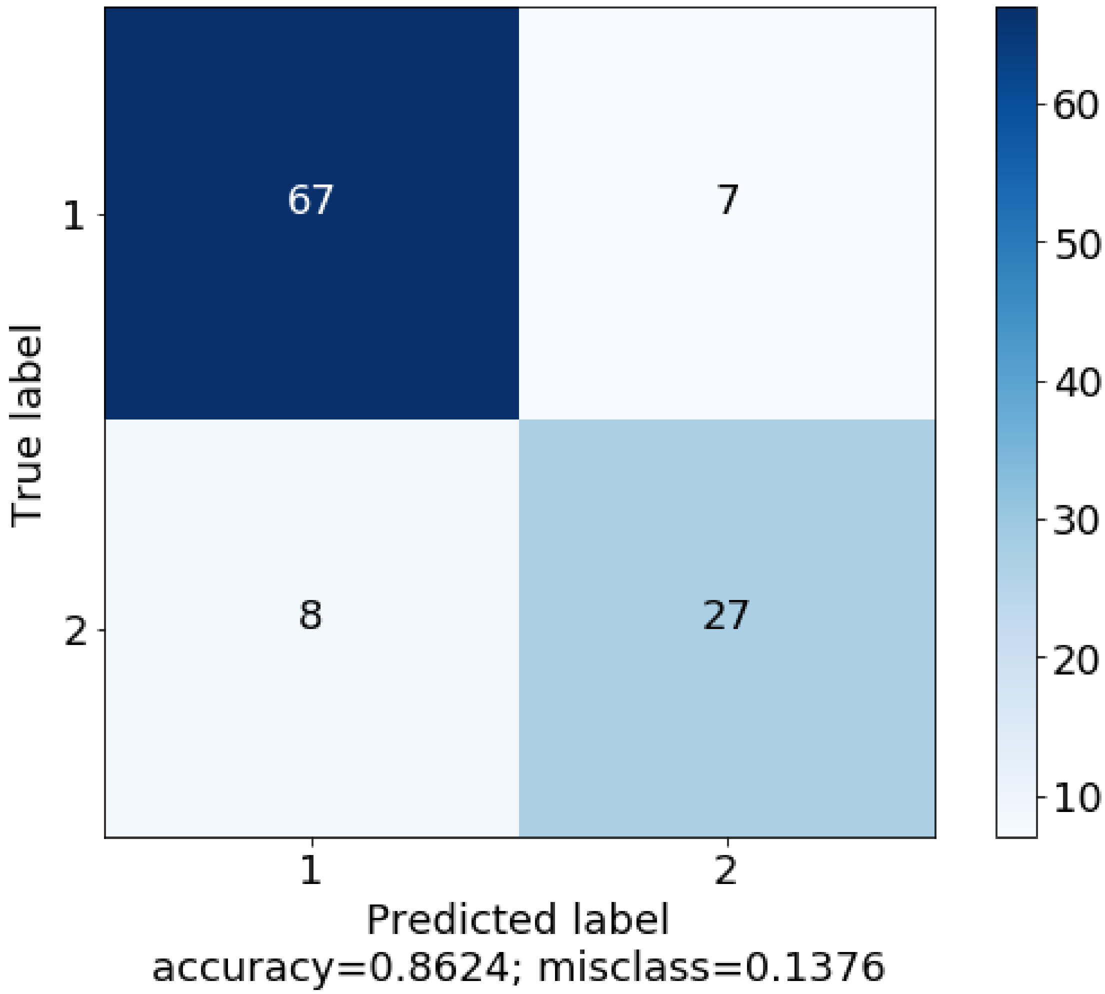
[107]:

```
plt.plot(history3.history['accuracy'])
plt.plot(history3.history['val_accuracy'])
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.xlim(0, 50)
plt.show()
```



[108]:

```
cm = confusion_matrix(y_test, y_pred3)
plt.rcParams.update({'font.size': 18})
plot_confusion_matrix(cm, ['1', '2'], title='', normalize=False)
```



[132]:
```
model3.model.save('models/hidden3.h5')
```

5 2 Hidden Layers

[116]:
```
def hidden2(optimizer='rmsprop',init='glorot_uniform', dropout=0.3):
    model = Sequential()
    # adding layers and adding droplayers to avoid overfitting
    hidden_layers = len(selected_genes)
    model.add(Dense(hidden_layers*2, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(dropout))

    model.add(Dense(hidden_layers*4, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(dropout))
```

```
    model.add(Dense(1, activation='sigmoid'))
    # compiling
    model.compile(optimizer=optimizer, loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
    return model
```

[117]:
```
# parameters selected from previous gridsearch
model2 = KerasClassifier(build_fn=hidden2, epochs=50, batch_size=32,␣
 ↪optimizer='adagrad',init='normal')
# kfold = KFold(n_splits=3, shuffle=True)
# results = cross_val_score(model, X_train, y_train, cv=kfold)
# print("Baseline Accuracy: %.2f%% (%.2f%%)" % (results.mean()*100, results.
 ↪std()*100))
```

[118]:
```
history2 = model2.fit(X_train, y_train, validation_data=(X_test, y_test),␣
 ↪shuffle=True)
y_pred2 = model2.predict(X_test)
```

```
Train on 556 samples, validate on 109 samples
Epoch 1/50
556/556 [==============================] - 2s 4ms/sample - loss: 0.5249 -
accuracy: 0.7914 - val_loss: 0.5775 - val_accuracy: 0.8899
Epoch 2/50
556/556 [==============================] - 0s 572us/sample - loss: 0.2664 -
accuracy: 0.8885 - val_loss: 0.5617 - val_accuracy: 0.8716
Epoch 3/50
556/556 [==============================] - 0s 572us/sample - loss: 0.1887 -
accuracy: 0.9227 - val_loss: 0.6212 - val_accuracy: 0.6972
Epoch 4/50
556/556 [==============================] - 0s 573us/sample - loss: 0.1727 -
accuracy: 0.9478 - val_loss: 0.5916 - val_accuracy: 0.7798
Epoch 5/50
556/556 [==============================] - 0s 572us/sample - loss: 0.1524 -
accuracy: 0.9317 - val_loss: 0.5295 - val_accuracy: 0.8532
Epoch 6/50
556/556 [==============================] - 0s 588us/sample - loss: 0.1466 -
accuracy: 0.9406 - val_loss: 0.5513 - val_accuracy: 0.7890
Epoch 7/50
556/556 [==============================] - 0s 620us/sample - loss: 0.1186 -
accuracy: 0.9604 - val_loss: 0.6230 - val_accuracy: 0.6147
Epoch 8/50
556/556 [==============================] - 0s 587us/sample - loss: 0.1348 -
accuracy: 0.9424 - val_loss: 0.5539 - val_accuracy: 0.7431
Epoch 9/50
556/556 [==============================] - 0s 582us/sample - loss: 0.0841 -
accuracy: 0.9712 - val_loss: 0.5368 - val_accuracy: 0.7156
```

```
Epoch 10/50
556/556 [==============================] - 0s 588us/sample - loss: 0.0914 -
accuracy: 0.9766 - val_loss: 0.4801 - val_accuracy: 0.7798
Epoch 11/50
556/556 [==============================] - 0s 590us/sample - loss: 0.1043 -
accuracy: 0.9712 - val_loss: 0.4058 - val_accuracy: 0.8440
Epoch 12/50
556/556 [==============================] - 0s 595us/sample - loss: 0.0605 -
accuracy: 0.9838 - val_loss: 0.3768 - val_accuracy: 0.8349
Epoch 13/50
556/556 [==============================] - 0s 592us/sample - loss: 0.0534 -
accuracy: 0.9874 - val_loss: 0.4098 - val_accuracy: 0.8257
Epoch 14/50
556/556 [==============================] - 0s 593us/sample - loss: 0.0635 -
accuracy: 0.9730 - val_loss: 0.3421 - val_accuracy: 0.8807
Epoch 15/50
556/556 [==============================] - 0s 605us/sample - loss: 0.0582 -
accuracy: 0.9874 - val_loss: 0.3176 - val_accuracy: 0.8899
Epoch 16/50
556/556 [==============================] - 0s 604us/sample - loss: 0.0614 -
accuracy: 0.9820 - val_loss: 0.3671 - val_accuracy: 0.8532
Epoch 17/50
556/556 [==============================] - 0s 599us/sample - loss: 0.0689 -
accuracy: 0.9802 - val_loss: 0.3609 - val_accuracy: 0.8440
Epoch 18/50
556/556 [==============================] - 0s 600us/sample - loss: 0.0640 -
accuracy: 0.9820 - val_loss: 0.3077 - val_accuracy: 0.8807
Epoch 19/50
556/556 [==============================] - 0s 612us/sample - loss: 0.0574 -
accuracy: 0.9820 - val_loss: 0.3933 - val_accuracy: 0.8440
Epoch 20/50
556/556 [==============================] - 0s 625us/sample - loss: 0.0353 -
accuracy: 0.9964 - val_loss: 0.3075 - val_accuracy: 0.8807
Epoch 21/50
556/556 [==============================] - 0s 642us/sample - loss: 0.0514 -
accuracy: 0.9874 - val_loss: 0.2947 - val_accuracy: 0.8899
Epoch 22/50
556/556 [==============================] - 0s 690us/sample - loss: 0.0573 -
accuracy: 0.9802 - val_loss: 0.3210 - val_accuracy: 0.8624
Epoch 23/50
556/556 [==============================] - 0s 687us/sample - loss: 0.0432 -
accuracy: 0.9946 - val_loss: 0.2817 - val_accuracy: 0.8807
Epoch 24/50
556/556 [==============================] - 0s 690us/sample - loss: 0.0477 -
accuracy: 0.9820 - val_loss: 0.2854 - val_accuracy: 0.8716
Epoch 25/50
556/556 [==============================] - 0s 660us/sample - loss: 0.0379 -
accuracy: 0.9874 - val_loss: 0.3211 - val_accuracy: 0.8899
```

```
Epoch 26/50
556/556 [==============================] - 0s 660us/sample - loss: 0.0464 -
accuracy: 0.9784 - val_loss: 0.3369 - val_accuracy: 0.8807
Epoch 27/50
556/556 [==============================] - 0s 654us/sample - loss: 0.0328 -
accuracy: 0.9892 - val_loss: 0.3853 - val_accuracy: 0.8532
Epoch 28/50
556/556 [==============================] - 0s 658us/sample - loss: 0.0324 -
accuracy: 0.9910 - val_loss: 0.3278 - val_accuracy: 0.8899
Epoch 29/50
556/556 [==============================] - 0s 664us/sample - loss: 0.0323 -
accuracy: 0.9928 - val_loss: 0.4191 - val_accuracy: 0.8440
Epoch 30/50
556/556 [==============================] - 0s 637us/sample - loss: 0.0458 -
accuracy: 0.9856 - val_loss: 0.3612 - val_accuracy: 0.8807
Epoch 31/50
556/556 [==============================] - 0s 648us/sample - loss: 0.0452 -
accuracy: 0.9892 - val_loss: 0.3764 - val_accuracy: 0.8624
Epoch 32/50
556/556 [==============================] - 0s 665us/sample - loss: 0.0407 -
accuracy: 0.9910 - val_loss: 0.3811 - val_accuracy: 0.8716
Epoch 33/50
556/556 [==============================] - 0s 642us/sample - loss: 0.0487 -
accuracy: 0.9892 - val_loss: 0.3714 - val_accuracy: 0.8716
Epoch 34/50
556/556 [==============================] - 0s 639us/sample - loss: 0.0356 -
accuracy: 0.9874 - val_loss: 0.3645 - val_accuracy: 0.8807
Epoch 35/50
556/556 [==============================] - 0s 644us/sample - loss: 0.0375 -
accuracy: 0.9892 - val_loss: 0.3735 - val_accuracy: 0.8807
Epoch 36/50
556/556 [==============================] - 0s 650us/sample - loss: 0.0294 -
accuracy: 0.9928 - val_loss: 0.3384 - val_accuracy: 0.8899
Epoch 37/50
556/556 [==============================] - 0s 692us/sample - loss: 0.0396 -
accuracy: 0.9892 - val_loss: 0.3592 - val_accuracy: 0.8991
Epoch 38/50
556/556 [==============================] - 0s 660us/sample - loss: 0.0342 -
accuracy: 0.9928 - val_loss: 0.3647 - val_accuracy: 0.8991
Epoch 39/50
556/556 [==============================] - 0s 649us/sample - loss: 0.0402 -
accuracy: 0.9856 - val_loss: 0.3556 - val_accuracy: 0.8991
Epoch 40/50
556/556 [==============================] - 0s 661us/sample - loss: 0.0242 -
accuracy: 0.9964 - val_loss: 0.3342 - val_accuracy: 0.8991
Epoch 41/50
556/556 [==============================] - 0s 646us/sample - loss: 0.0299 -
accuracy: 0.9910 - val_loss: 0.3457 - val_accuracy: 0.8991
```

```
Epoch 42/50
556/556 [==============================] - 0s 641us/sample - loss: 0.0257 -
accuracy: 0.9946 - val_loss: 0.3692 - val_accuracy: 0.8991
Epoch 43/50
556/556 [==============================] - 0s 647us/sample - loss: 0.0190 -
accuracy: 0.9964 - val_loss: 0.3593 - val_accuracy: 0.8991
Epoch 44/50
556/556 [==============================] - 0s 659us/sample - loss: 0.0192 -
accuracy: 0.9982 - val_loss: 0.3444 - val_accuracy: 0.8899
Epoch 45/50
556/556 [==============================] - 0s 663us/sample - loss: 0.0269 -
accuracy: 0.9964 - val_loss: 0.3578 - val_accuracy: 0.8899
Epoch 46/50
556/556 [==============================] - 0s 657us/sample - loss: 0.0179 -
accuracy: 0.9964 - val_loss: 0.3813 - val_accuracy: 0.8899
Epoch 47/50
556/556 [==============================] - 0s 654us/sample - loss: 0.0211 -
accuracy: 0.9946 - val_loss: 0.3829 - val_accuracy: 0.8899
Epoch 48/50
556/556 [==============================] - 0s 664us/sample - loss: 0.0372 -
accuracy: 0.9892 - val_loss: 0.3863 - val_accuracy: 0.8899
Epoch 49/50
556/556 [==============================] - 0s 659us/sample - loss: 0.0301 -
accuracy: 0.9910 - val_loss: 0.4123 - val_accuracy: 0.8807
Epoch 50/50
556/556 [==============================] - 0s 656us/sample - loss: 0.0198 -
accuracy: 0.9964 - val_loss: 0.3916 - val_accuracy: 0.8899
```

[104]:
```python
print(classification_report(y_test, y_pred2))
```

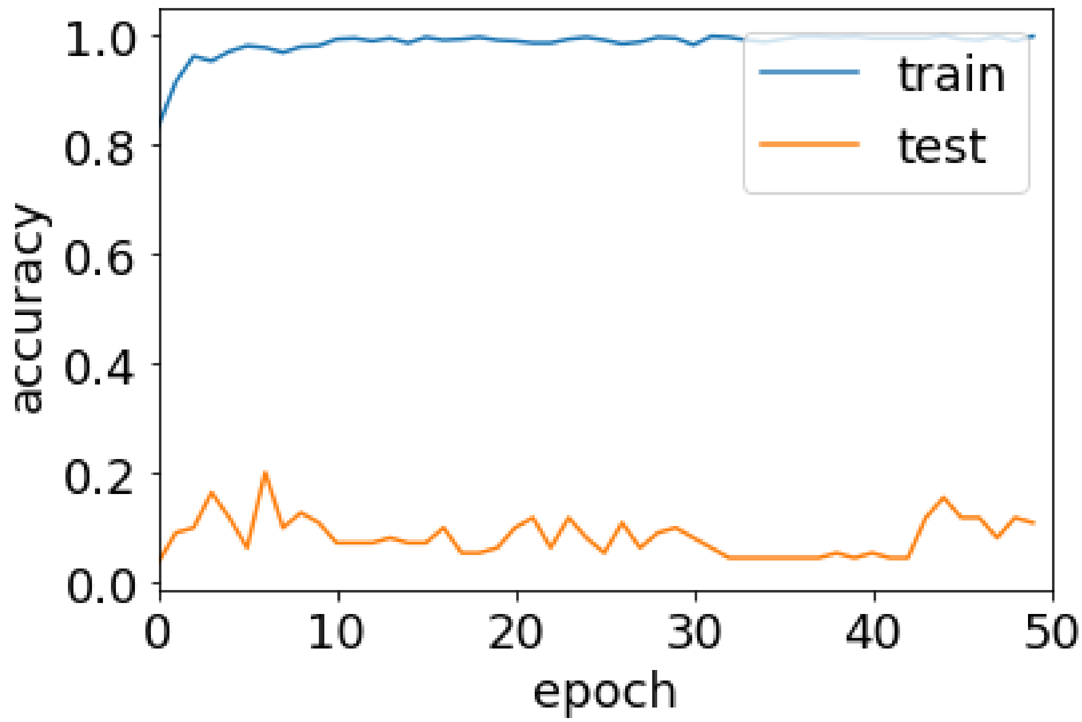
```
              precision    recall  f1-score   support

           1       0.84      0.81      0.82        63
           2       0.75      0.78      0.77        46

    accuracy                           0.80       109
   macro avg       0.79      0.80      0.79       109
weighted avg       0.80      0.80      0.80       109
```

[105]:
```python
print(confusion_matrix(y_test, y_pred2))
```

```
[[51 12]
 [10 36]]
```

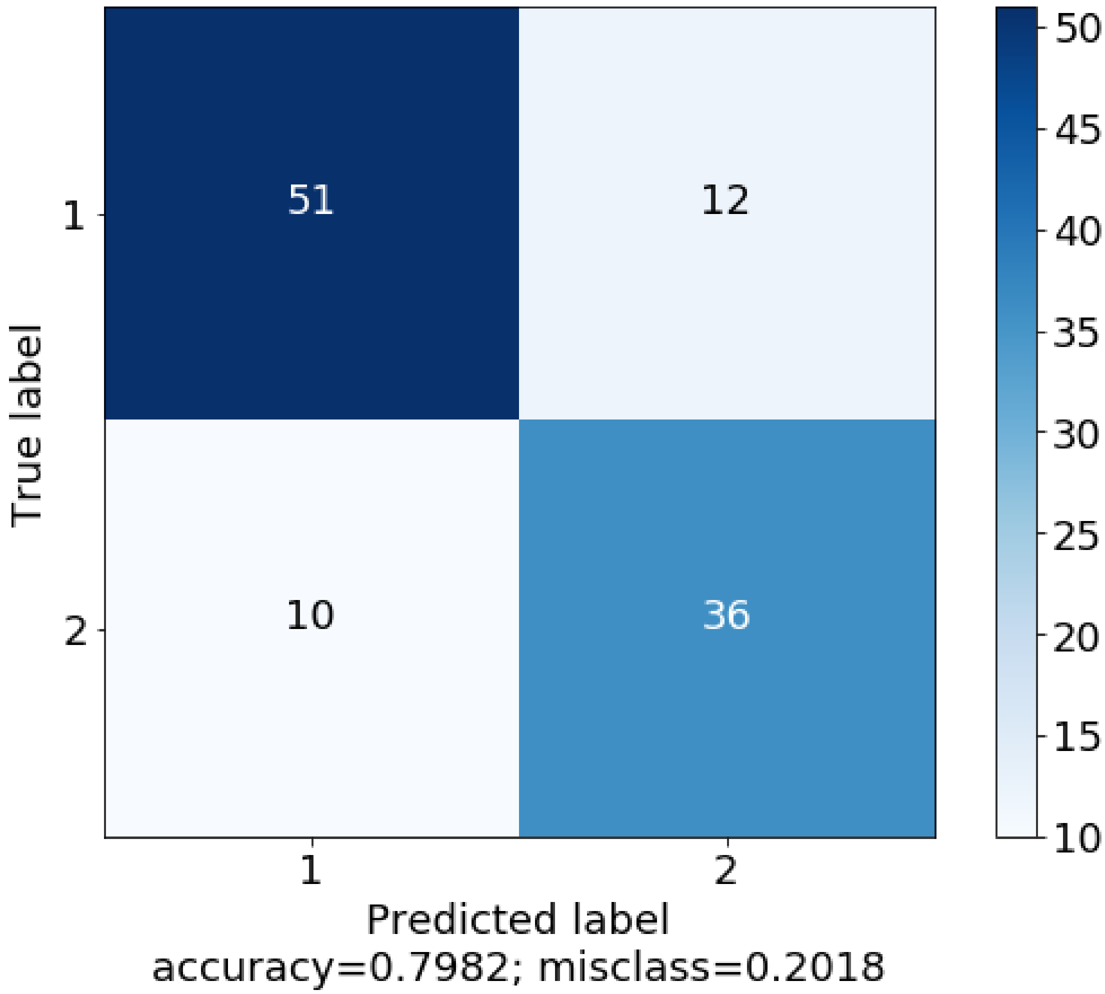
[106]:
```python
plt.plot(history2.history['accuracy'])
plt.plot(history2.history['val_accuracy'])
plt.ylabel('accuracy')
```

```python
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.xlim(0, 50)
plt.show()
```



[107]:

```python
cm = confusion_matrix(y_test, y_pred2)
plt.rcParams.update({'font.size': 18})
plot_confusion_matrix(cm, ['1', '2'], title='', normalize=False)
```



[133]:
```
model2.model.save('models/hidden2.h5')
```

6 1 Hidden Layer

[119]:
```
def hidden1(optimizer='rmsprop',init='glorot_uniform', dropout=0.3):
    model = Sequential()
    # adding layers and adding droplayers to avoid overfitting
    hidden_layers = len(selected_genes)
    model.add(Dense(hidden_layers*1.5, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(dropout))

    model.add(Dense(1, activation='sigmoid'))
    # compiling
```

```
    model.compile(optimizer=optimizer, loss='binary_crossentropy',␣
↪metrics=['accuracy'])
    return model
```

[125]:
```
# parameters selected from previous gridsearch
model1 = KerasClassifier(build_fn=hidden1, epochs=50, batch_size=32,␣
↪optimizer='adagrad',init='normal')
# kfold = KFold(n_splits=3, shuffle=True)
# results = cross_val_score(model, X_train, y_train, cv=kfold)
# print("Baseline Accuracy: %.2f%% (%.2f%%)" % (results.mean()*100, results.
↪std()*100))
```

[126]:
```
history1 = model1.fit(X_train, y_train, validation_data=(X_test, y_test),␣
↪shuffle=True)
y_pred1 = model1.predict(X_test)
```

```
Train on 556 samples, validate on 109 samples
Epoch 1/50
556/556 [==============================] - 1s 3ms/sample - loss: 0.4015 -
accuracy: 0.8345 - val_loss: 0.5112 - val_accuracy: 0.8165
Epoch 2/50
556/556 [==============================] - 0s 388us/sample - loss: 0.3253 -
accuracy: 0.8723 - val_loss: 0.5721 - val_accuracy: 0.7339
Epoch 3/50
556/556 [==============================] - 0s 359us/sample - loss: 0.2694 -
accuracy: 0.8993 - val_loss: 0.5022 - val_accuracy: 0.8349
Epoch 4/50
556/556 [==============================] - 0s 367us/sample - loss: 0.2458 -
accuracy: 0.9065 - val_loss: 0.5061 - val_accuracy: 0.8349
Epoch 5/50
556/556 [==============================] - 0s 367us/sample - loss: 0.2317 -
accuracy: 0.9083 - val_loss: 0.5156 - val_accuracy: 0.7890
Epoch 6/50
556/556 [==============================] - 0s 363us/sample - loss: 0.2153 -
accuracy: 0.9155 - val_loss: 0.4787 - val_accuracy: 0.8257
Epoch 7/50
556/556 [==============================] - 0s 376us/sample - loss: 0.2084 -
accuracy: 0.9263 - val_loss: 0.5145 - val_accuracy: 0.7615
Epoch 8/50
556/556 [==============================] - 0s 364us/sample - loss: 0.2136 -
accuracy: 0.9317 - val_loss: 0.4550 - val_accuracy: 0.8257
Epoch 9/50
556/556 [==============================] - 0s 370us/sample - loss: 0.1949 -
accuracy: 0.9353 - val_loss: 0.4561 - val_accuracy: 0.8257
Epoch 10/50
556/556 [==============================] - 0s 376us/sample - loss: 0.2026 -
accuracy: 0.9245 - val_loss: 0.4779 - val_accuracy: 0.7615
```

```
Epoch 11/50
556/556 [==============================] - 0s 363us/sample - loss: 0.2045 -
accuracy: 0.9245 - val_loss: 0.4421 - val_accuracy: 0.8257
Epoch 12/50
556/556 [==============================] - 0s 350us/sample - loss: 0.1819 -
accuracy: 0.9442 - val_loss: 0.4145 - val_accuracy: 0.8257
Epoch 13/50
556/556 [==============================] - 0s 353us/sample - loss: 0.1723 -
accuracy: 0.9424 - val_loss: 0.4161 - val_accuracy: 0.8257
Epoch 14/50
556/556 [==============================] - 0s 369us/sample - loss: 0.1608 -
accuracy: 0.9460 - val_loss: 0.3882 - val_accuracy: 0.8165
Epoch 15/50
556/556 [==============================] - 0s 373us/sample - loss: 0.1668 -
accuracy: 0.9442 - val_loss: 0.3590 - val_accuracy: 0.8532
Epoch 16/50
556/556 [==============================] - 0s 371us/sample - loss: 0.1692 -
accuracy: 0.9335 - val_loss: 0.3632 - val_accuracy: 0.8349
Epoch 17/50
556/556 [==============================] - 0s 369us/sample - loss: 0.1486 -
accuracy: 0.9514 - val_loss: 0.3522 - val_accuracy: 0.8440
Epoch 18/50
556/556 [==============================] - 0s 370us/sample - loss: 0.1680 -
accuracy: 0.9478 - val_loss: 0.3233 - val_accuracy: 0.8532
Epoch 19/50
556/556 [==============================] - 0s 373us/sample - loss: 0.1588 -
accuracy: 0.9496 - val_loss: 0.3149 - val_accuracy: 0.8532
Epoch 20/50
556/556 [==============================] - 0s 360us/sample - loss: 0.1623 -
accuracy: 0.9460 - val_loss: 0.3488 - val_accuracy: 0.8257
Epoch 21/50
556/556 [==============================] - 0s 376us/sample - loss: 0.1585 -
accuracy: 0.9586 - val_loss: 0.3334 - val_accuracy: 0.8440
Epoch 22/50
556/556 [==============================] - 0s 393us/sample - loss: 0.1301 -
accuracy: 0.9604 - val_loss: 0.3068 - val_accuracy: 0.8624
Epoch 23/50
556/556 [==============================] - 0s 380us/sample - loss: 0.1693 -
accuracy: 0.9460 - val_loss: 0.3744 - val_accuracy: 0.8349
Epoch 24/50
556/556 [==============================] - 0s 366us/sample - loss: 0.1263 -
accuracy: 0.9730 - val_loss: 0.3241 - val_accuracy: 0.8440
Epoch 25/50
556/556 [==============================] - 0s 376us/sample - loss: 0.1221 -
accuracy: 0.9676 - val_loss: 0.3090 - val_accuracy: 0.8532
Epoch 26/50
556/556 [==============================] - 0s 356us/sample - loss: 0.1304 -
accuracy: 0.9658 - val_loss: 0.3241 - val_accuracy: 0.8440
```

```
Epoch 27/50
556/556 [==============================] - 0s 355us/sample - loss: 0.1471 -
accuracy: 0.9514 - val_loss: 0.3130 - val_accuracy: 0.8440
Epoch 28/50
556/556 [==============================] - 0s 382us/sample - loss: 0.1077 -
accuracy: 0.9766 - val_loss: 0.3204 - val_accuracy: 0.8532
Epoch 29/50
556/556 [==============================] - 0s 390us/sample - loss: 0.1521 -
accuracy: 0.9568 - val_loss: 0.3219 - val_accuracy: 0.8440
Epoch 30/50
556/556 [==============================] - 0s 385us/sample - loss: 0.1373 -
accuracy: 0.9658 - val_loss: 0.3118 - val_accuracy: 0.8532
Epoch 31/50
556/556 [==============================] - 0s 373us/sample - loss: 0.1222 -
accuracy: 0.9640 - val_loss: 0.3384 - val_accuracy: 0.8257
Epoch 32/50
556/556 [==============================] - 0s 365us/sample - loss: 0.1036 -
accuracy: 0.9820 - val_loss: 0.3118 - val_accuracy: 0.8624
Epoch 33/50
556/556 [==============================] - 0s 364us/sample - loss: 0.1180 -
accuracy: 0.9586 - val_loss: 0.3152 - val_accuracy: 0.8532
Epoch 34/50
556/556 [==============================] - 0s 374us/sample - loss: 0.1107 -
accuracy: 0.9766 - val_loss: 0.3104 - val_accuracy: 0.8624
Epoch 35/50
556/556 [==============================] - 0s 363us/sample - loss: 0.1281 -
accuracy: 0.9622 - val_loss: 0.3048 - val_accuracy: 0.8716
Epoch 36/50
556/556 [==============================] - 0s 372us/sample - loss: 0.1117 -
accuracy: 0.9676 - val_loss: 0.3054 - val_accuracy: 0.8624
Epoch 37/50
556/556 [==============================] - 0s 368us/sample - loss: 0.1159 -
accuracy: 0.9658 - val_loss: 0.3108 - val_accuracy: 0.8624
Epoch 38/50
556/556 [==============================] - 0s 388us/sample - loss: 0.1156 -
accuracy: 0.9604 - val_loss: 0.3148 - val_accuracy: 0.8624
Epoch 39/50
556/556 [==============================] - 0s 383us/sample - loss: 0.0915 -
accuracy: 0.9784 - val_loss: 0.3142 - val_accuracy: 0.8624
Epoch 40/50
556/556 [==============================] - 0s 376us/sample - loss: 0.0996 -
accuracy: 0.9802 - val_loss: 0.3217 - val_accuracy: 0.8624
Epoch 41/50
556/556 [==============================] - 0s 372us/sample - loss: 0.1027 -
accuracy: 0.9730 - val_loss: 0.3215 - val_accuracy: 0.8532
Epoch 42/50
556/556 [==============================] - 0s 394us/sample - loss: 0.0983 -
accuracy: 0.9730 - val_loss: 0.3240 - val_accuracy: 0.8440
```

```
Epoch 43/50
556/556 [==============================] - 0s 390us/sample - loss: 0.1110 -
accuracy: 0.9712 - val_loss: 0.3272 - val_accuracy: 0.8440
Epoch 44/50
556/556 [==============================] - 0s 390us/sample - loss: 0.1008 -
accuracy: 0.9748 - val_loss: 0.3212 - val_accuracy: 0.8440
Epoch 45/50
556/556 [==============================] - 0s 396us/sample - loss: 0.1014 -
accuracy: 0.9748 - val_loss: 0.3293 - val_accuracy: 0.8349
Epoch 46/50
556/556 [==============================] - 0s 390us/sample - loss: 0.1125 -
accuracy: 0.9712 - val_loss: 0.3360 - val_accuracy: 0.8349
Epoch 47/50
556/556 [==============================] - 0s 397us/sample - loss: 0.1026 -
accuracy: 0.9712 - val_loss: 0.3232 - val_accuracy: 0.8624
Epoch 48/50
556/556 [==============================] - 0s 405us/sample - loss: 0.0982 -
accuracy: 0.9676 - val_loss: 0.3349 - val_accuracy: 0.8349
Epoch 49/50
556/556 [==============================] - 0s 396us/sample - loss: 0.1019 -
accuracy: 0.9748 - val_loss: 0.3354 - val_accuracy: 0.8440
Epoch 50/50
556/556 [==============================] - 0s 395us/sample - loss: 0.0936 -
accuracy: 0.9748 - val_loss: 0.3483 - val_accuracy: 0.8257
```

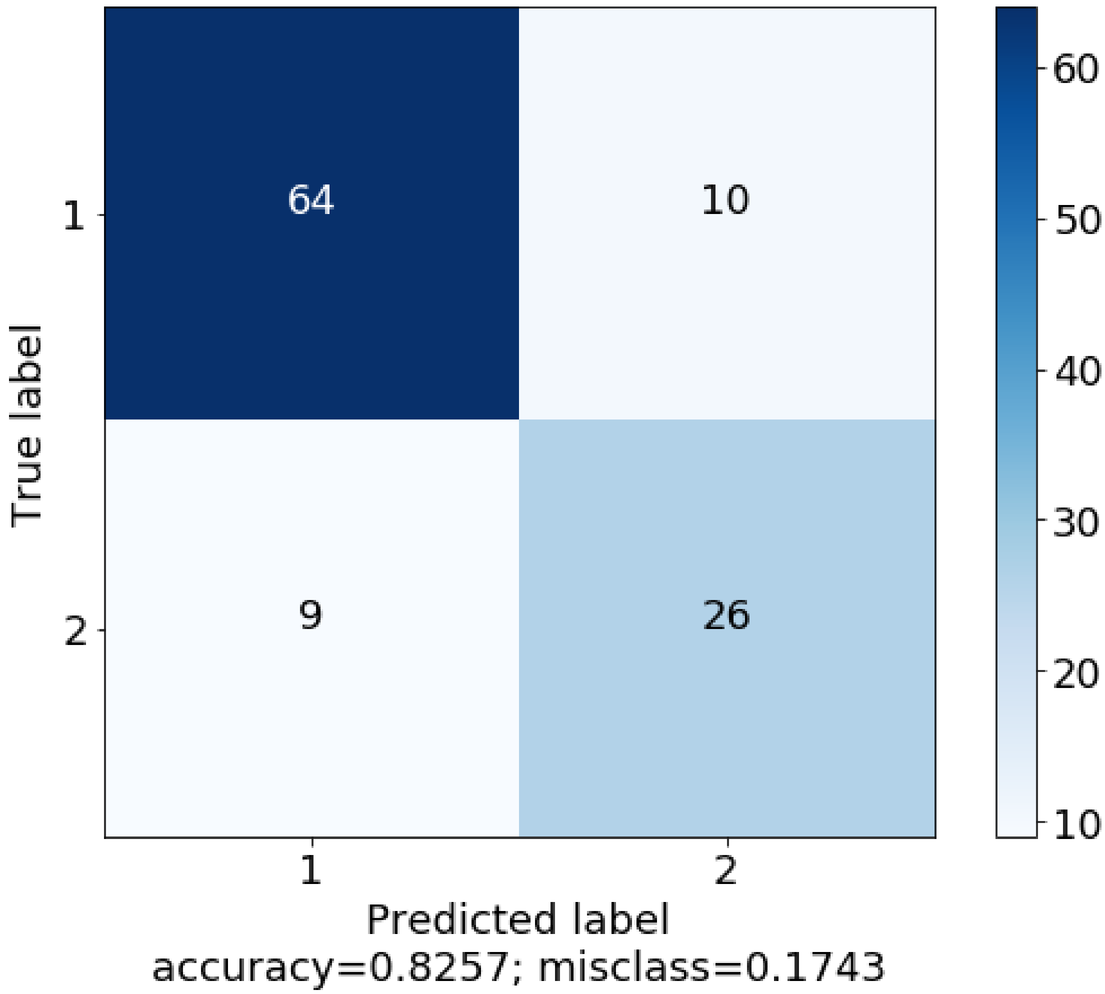
[127]:
```
print(classification_report(y_test, y_pred1))
```

```
              precision    recall  f1-score   support

           0       0.88      0.86      0.87        74
           1       0.72      0.74      0.73        35

    accuracy                           0.83       109
   macro avg       0.80      0.80      0.80       109
weighted avg       0.83      0.83      0.83       109
```

[128]:
```
print(confusion_matrix(y_test, y_pred1))
```

```
[[64 10]
 [ 9 26]]
```

[129]:
```
cm = confusion_matrix(y_test, y_pred1)
plt.rcParams.update({'font.size': 18})
plot_confusion_matrix(cm, ['1', '2'], title='', normalize=False)
```



[134]:
```
model1.model.save('models/hidden1.h5')
```

[ ]:
```
```