```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
from functools import *
from google.colab import files
import matplotlib.pyplot as plt

np.set_printoptions(formatter = {'float': '{: 0.1f}'.format})


class color:
  BOLD = '\033[1m'
  END = '\033[0m'
```

Attribute Information

1) id: unique identifier
2) gender: "Male (0)", "Female(1)" or "Other(2)"
3) age: age of the patient
4) hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
5) heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
6) ever_married: "No (0)" or "Yes (1)"
7) work_type: "children (0)", "Govt_job (1)", "Never_worked (2)", "Private (3)" or "Self-employed (4)"
8) Residence_type: "Rural (0)" or "Urban (1)"
9) avg_glucose_level: average glucose level in blood
10) bmi: body mass index
11) smoking_status: "formerly smoked (0)", "never smoked (1)", "smokes (2)" or "Unknown (3)"*
12) stroke: 1 if the patient had a stroke or 0 if not\

*Note: "Unknown" in smoking_status means that the information is unavailable for this patient

Visualize Data

35 cells hidden

Data Normalization

[ ] *12 cells hidden*

Using SMOTE to Combat Undersampled Stroke Class

[ ] ↳ *16 cells hidden*

Set Up Neural Networks

[ ] ↳ *11 cells hidden*

Compile Neural Networks

```
model_baseline.compile(loss = 'binary_crossentropy', optimizer='rmsprop', metrics=['accuracy'
model_linearLASTn.compile(loss = 'mse', optimizer='rmsprop', metrics=['mae'] )
model_linearALLn.compile(loss = 'mse', optimizer='rmsprop', metrics=['mae'] )
model_sigmoidLASTn.compile(loss = 'binary_crossentropy', optimizer='rmsprop', metrics=['accur
model_sigmoidALLn.compile(loss = 'binary_crossentropy', optimizer='rmsprop', metrics=['accura
model_overfit.compile(loss = 'binary_crossentropy', optimizer='rmsprop', metrics=['accuracy']
```

```
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
```

```
from keras.models import load_model
```

Breakdown of early stopping:

```
Patience: keeps on training until the validation accuracy doesn't improve for n amount of times.
Batch size: is the sameple size. In this case it will be all of the validation data because of the l
Validation split: what percent of the trained data is being used. In this case we will go with all f
```

```
checkpoint_callback_baseline = ModelCheckpoint(filepath = 'model_baseline.hdf5', monitor = 'a
                                    save_best_only = True, save_weights_only = True, verbos

earlystop_callback_baseline = EarlyStopping(monitor='accuracy',
                                   patience=1000)
```

```
checkpoint_callback_linearLAST = ModelCheckpoint(filepath = 'linearLAST.hdf5', monitor = 'mae
                                    save_best_only = True, save_weights_only = True, verbos
```

```
earlystop_callback_linearLAST = EarlyStopping(monitor='mae',
                                    patience=1000)

checkpoint_callback_linearALL = ModelCheckpoint(filepath = 'linearALL.hdf5', monitor = 'mae',
                                    save_best_only = True, save_weights_only = True, verbos

earlystop_callback_linearALL = EarlyStopping(monitor='mae',
                                    patience=1000)


checkpoint_callback_sigmoidLAST = ModelCheckpoint(filepath = 'sigmoidLAST.hdf5', monitor = 'a
                                    save_best_only = True, save_weights_only = True, verbos

earlystop_callback_sigmoidLAST = EarlyStopping(monitor='accuracy',
                                    patience=1000)


checkpoint_callback_sigmoidALL = ModelCheckpoint(filepath = 'sigmoidALL.hdf5', monitor = 'acc
                                    save_best_only = True, save_weights_only = True, verbos

earlystop_callback_sigmoidALL = EarlyStopping(monitor='accuracy',
                                    patience=1000)


checkpoint_callback_overfit = ModelCheckpoint(filepath = 'overfit.hdf5', monitor = 'accuracy'
                                    save_best_only = True, save_weights_only = True, verbos

earlystop_callback_overfit = EarlyStopping(monitor='accuracy',
                                    patience=1000)


learningCURVEbaseline = model_baseline.fit(XTRAIN_respl, YTRAIN_respl.ravel(), validation_dat
                                        callbacks= [checkpoint_callback_baseline, earlystc



model_baseline.save_weights('model_baseline.hdf5')


model_baseline.load_weights('model_baseline.hdf5')


learningCURVElinearLAST = model_linearLASTn.fit(XTRAIN_respl, YTRAIN_respl.ravel(), validatic
                                            callbacks= [checkpoint_callback_linearLAST, e


model_linearLASTn.save_weights('linearLAST.hdf5')


model_linearLASTn.load_weights('linearLAST.hdf5')


learningCURVElinearALL = model_linearALLn.fit(XTRAIN_respl, YTRAIN_respl.ravel(), validation_
                                        callbacks= [checkpoint_callback_linearALL, earl
```

```
model_linearALLn.save_weights('linearALL.hdf5')
```

```
model_linearALLn.load_weights('linearALL.hdf5')
```

```
learningCURVEsigmoidLAST = model_sigmoidLASTn.fit(XTRAIN_respl, YTRAIN_respl.ravel(), validat
                                                  callbacks= [checkpoint_callback_sigmoidLAST
```

```
model_sigmoidLASTn.save_weights('sigmoidLAST.hdf5')
```

```
model_sigmoidLASTn.load_weights('sigmoidLAST.hdf5')
```

```
learningCURVEsigmoidALL = model_sigmoidALLn.fit(XTRAIN_respl, YTRAIN_respl.ravel(), validatic
                                                callbacks= [checkpoint_callback_sigmoidALL, e
```

```
model_sigmoidALLn.save_weights('sigmoidALL.hdf5')
```

```
model_sigmoidALLn.load_weights('sigmoidALL.hdf5')
```

```
learningCURVEoverfit = model_overfit.fit(XTRAIN_respl, YTRAIN_respl.ravel(), validation_data
                                         callbacks= [checkpoint_callback_overfit, earlystop_c
```

```
model_overfit.save_weights('overfit.hdf5')
```

```
model_overfit.load_weights('overfit.hdf5')
```

```
 model_baseline.evaluate(XVALID, YVALID)
```

```
     47/47 [==============================] - 0s 1ms/step - loss: 0.6356 - accuracy: 0.6884
     [0.6355522274971008, 0.6883910298347473]
```

```
model_linearLASTn.evaluate(XVALID, YVALID)
```

```
     47/47 [==============================] - 0s 1ms/step - loss: 0.2188 - mae: 0.3674
     [0.2187659591436386, 0.3674035370349884]
```

```
model_linearALLn.evaluate(XVALID, YVALID)
```

```
     47/47 [==============================] - 0s 2ms/step - loss: 0.1659 - mae: 0.3051
     [0.16594967246055603, 0.3051399886608124]
```

```
model_sigmoidLASTn.evaluate(XVALID, YVALID)
```

```
47/47 [==============================] - 0s 1ms/step - loss: 0.4946 - accuracy: 0.7230
[0.4946019649505615, 0.723014235496521]
```

```
model_sigmoidALLn.evaluate(XVALID, YVALID)
```

```
47/47 [==============================] - 0s 1ms/step - loss: 0.4492 - accuracy: 0.7800
[0.4492034614086151, 0.7800407409667969]
```

```
model_overfit.evaluate(XVALID, YVALID)
```

```
47/47 [==============================] - 0s 2ms/step - loss: 0.6072 - accuracy: 0.7298
[0.6072478890419006, 0.7298031449317932]
```

Learning Curves & Model Evaluation

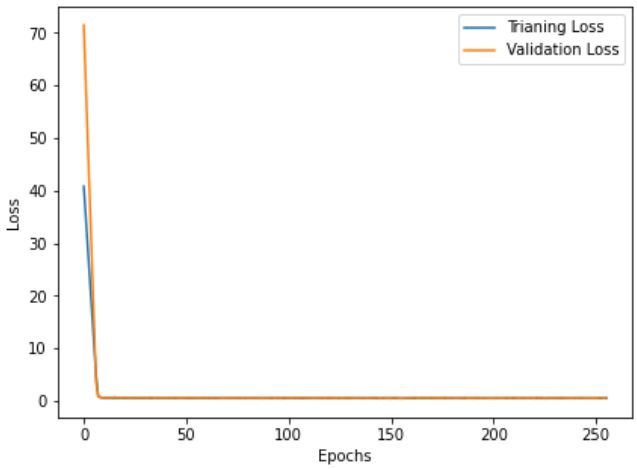
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, precisic
```

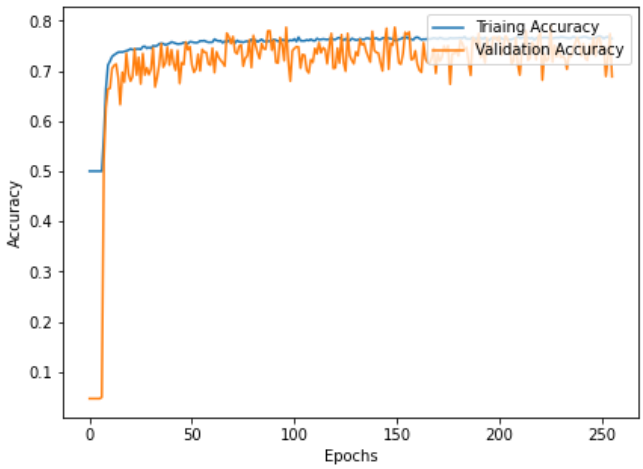
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

```
def logistic_learning_curve(modelHistory):
    fig, (x,y) = plt.subplots(1,2,
                              figsize = (15,5))

    x.plot(modelHistory.history['loss'])
    x.plot(modelHistory.history['val_loss'])

    x.set_xlabel('Epochs')
    x.set_ylabel('Loss')

    x.legend(['Trianing Loss', 'Validation Loss'],
             loc = 'upper right')


    y.plot(modelHistory.history['accuracy'])
    y.plot(modelHistory.history['val_accuracy'])

    y.set_xlabel('Epochs')
    y.set_ylabel('Accuracy')

    y.legend(['Triaing Accuracy', 'Validation Accuracy'],
             loc = 'upper right')

    fig.show()
```

```
def linear_learning_curve(modelHistory):
    fig, (x,y) = plt.subplots(1,2,
                              figsize = (15,5))

    x.plot(modelHistory.history['loss'])
    x.plot(modelHistory.history['val_loss'])

    x.set_xlabel('Epochs')
    x.set_ylabel('Loss')

    x.legend(['Trianing Loss', 'Validation Loss'],
             loc = 'upper right')


    y.plot(modelHistory.history['mae'])
    y.plot(modelHistory.history['val_mae'])

    y.set_xlabel('Epochs')
    y.set_ylabel('Mean Abosolute Error')

    y.legend(['Triaing Mean Abosolute Error', 'Validation Mean Abosolute Error'],
             loc = 'upper right')

    fig.show()


def modelEvaluation(xvalidation, yvalidation, modeltype):

    Ypredictions = modeltype.predict(xvalidation)
    Ypredictions = (Ypredictions > 0.5)

    confusion_matrix(yvalidation,Ypredictions)

    print(classification_report(yvalidation,Ypredictions))
    return Ypredictions



def rocauc_graph(yvalidation, model_eval, title='ROC Curve'):
  fpr, tpr, thresholds = roc_curve(YVALID, model_eval)

  roc_auc = auc(fpr, tpr)

  plt.title(title)

  plt.plot(fpr, tpr ,color="b", label='AUC = %0.2f'% roc_auc)
  plt.plot([0, 1], [0, 1], 'r--')

  plt.ylabel('True Positive Rate')
  plt.xlabel('False Positive Rate')
```

```
logistic_learning_curve(learningCURVEbaseline)
```

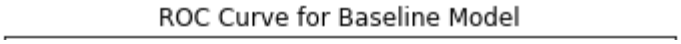





```
ypredictionsBASELINE = modelEvaluation(XVALID, YVALID, model_baseline)
```

```
                  precision    recall  f1-score   support

               0       0.99      0.68      0.81      1404
               1       0.12      0.86      0.20        69

        accuracy                           0.69      1473
       macro avg       0.55      0.77      0.51      1473
    weighted avg       0.95      0.69      0.78      1473
```

```
rocauc_graph(YVALID, ypredictionsBASELINE, 'ROC Curve for Baseline Model')
```



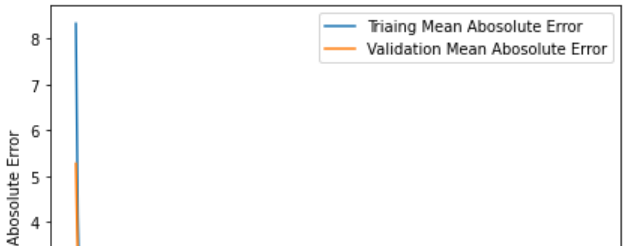

```
prediction_baseline = model_baseline.predict(XTRAIN)
accuracy_baseline = accuracy_score(YTRAIN, prediction_baseline.round())
precision_baseline = precision_score(YTRAIN, prediction_baseline.round())
recall_baseline = recall_score(YTRAIN, prediction_baseline.round())
f1score_baseline = f1_score(YTRAIN, prediction_baseline.round())

print(color.BOLD + "The Training Predictions:" + color.END)
print("Accuracy Score: %.2f%%" % (accuracy_baseline * 100))
print("Precision Score: %.2f%%" % (precision_baseline * 100))
print("Recall Score: %.2f%%" % (recall_baseline * 100))
print("F1-score: %.4f" % (f1score_baseline * 100 ))
```

The Training Predictions:
Accuracy Score: 68.39%
Precision Score: 9.98%
Recall Score: 84.29%
F1-score: 17.8517

```
prediction_baseline = model_baseline.predict(XVALID)
accuracy_baseline = accuracy_score(YVALID, prediction_baseline.round())
precision_baseline = precision_score(YVALID, prediction_baseline.round())
recall_baseline = recall_score(YVALID, prediction_baseline.round())
f1score_baseline = f1_score(YVALID, prediction_baseline.round())

print(color.BOLD + "The Validation Predictions:" + color.END)
print("Accuracy Score: %.2f%%" % (accuracy_baseline * 100))
print("Precision Score: %.2f%%" % (precision_baseline * 100))
print("Recall Score: %.2f%%" % (recall_baseline * 100))
print("F1-score: %.4f" % (f1score_baseline * 100))
```

The Validation Predictions:
Accuracy Score: 68.84%
Precision Score: 11.61%
Recall Score: 85.51%
F1-score: 20.4506

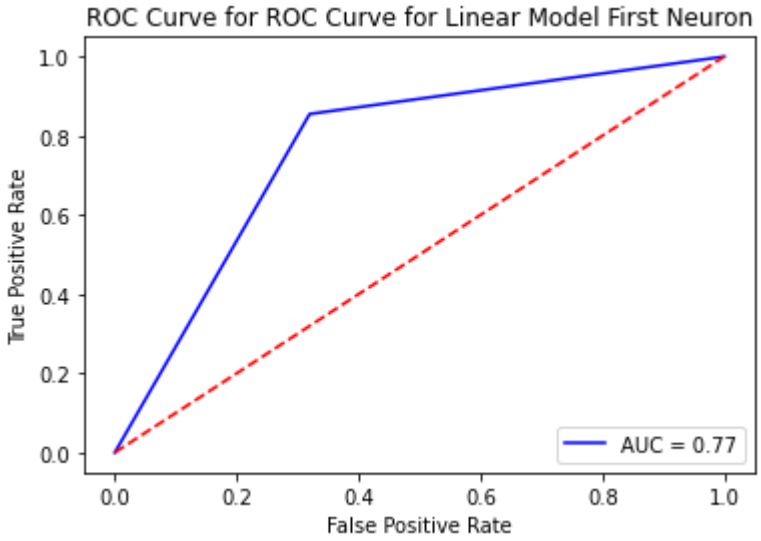
```
linear_learning_curve(learningCURVElinearLAST)
```




```
ypredictionsLINEARlastn = modelEvaluation(XVALID, YVALID, model_linearLASTn)
```

```
                precision    recall  f1-score   support

             0       0.99      0.68      0.81      1404
             1       0.12      0.86      0.20        69

      accuracy                           0.69      1473
     macro avg       0.55      0.77      0.51      1473
  weighted avg       0.95      0.69      0.78      1473
```

```
rocauc_graph(YVALID, ypredictionsLINEARlastn, 'ROC Curve for ROC Curve for Linear Model First
```




```
prediction_linearLAST = model_linearLASTn.predict(XTRAIN)
accuracy_linearLAST = accuracy_score(YTRAIN, prediction_linearLAST.round())
precision_linearLAST = precision_score(YTRAIN, prediction_linearLAST.round(), average='micro'
recall_linearLAST = recall_score(YTRAIN, prediction_linearLAST.round(), average='micro')
f1score_linearLAST = f1_score(YTRAIN, prediction_linearLAST.round(), average='micro')

print(color.BOLD + "The Training Predictions:" + color.END)
print("Accuracy Score: %.2f%%" % (accuracy_linearLAST * 100))
print("Precision Score: %.2f%%" % (precision_linearLAST * 100))
print("Recall Score: %.2f%%" % (recall_linearLAST * 100))
print("F1-score: %.4f" % (f1score_linearLAST * 100 ))
```

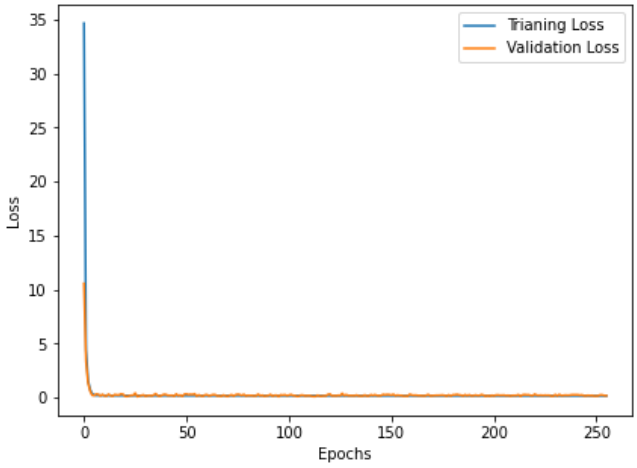
The Training Predictions:
Accuracy Score: 68.98%

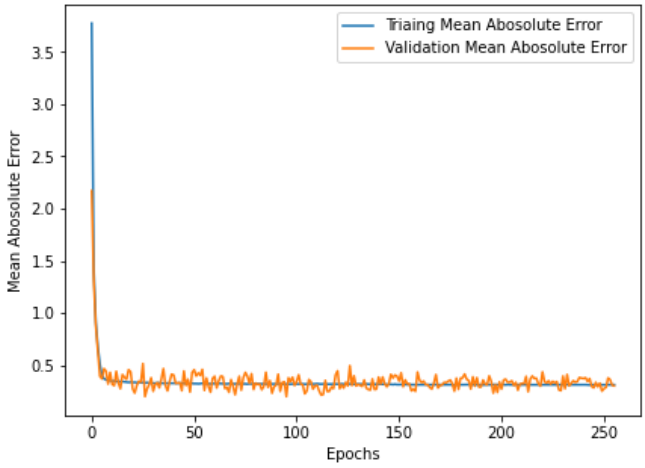
```
Precision Score: 68.98%
Recall Score: 68.98%
F1-score: 68.9756
```

```
prediction_linearLAST = model_linearLASTn.predict(XVALID)
accuracy_linearLAST = accuracy_score(YVALID, prediction_linearLAST.round())
precision_linearLAST = precision_score(YVALID, prediction_linearLAST.round(), average='micro'
recall_linearLAST = recall_score(YVALID, prediction_linearLAST.round(), average='micro')
f1score_linearLAST = f1_score(YVALID, prediction_linearLAST.round(), average='micro')

print(color.BOLD + "The Validation Predictions:" + color.END)
print("Accuracy Score: %.2f%%" % (accuracy_linearLAST * 100))
print("Precision Score: %.2f%%" % (precision_linearLAST * 100))
print("Recall Score: %.2f%%" % (recall_linearLAST * 100))
print("F1-score: %.4f" % (f1score_linearLAST * 100 ))
```

```
The Validation Predictions:
Accuracy Score: 68.70%
Precision Score: 68.70%
Recall Score: 68.70%
F1-score: 68.7033
```

```
linear_learning_curve(learningCURVElinearALL)
```



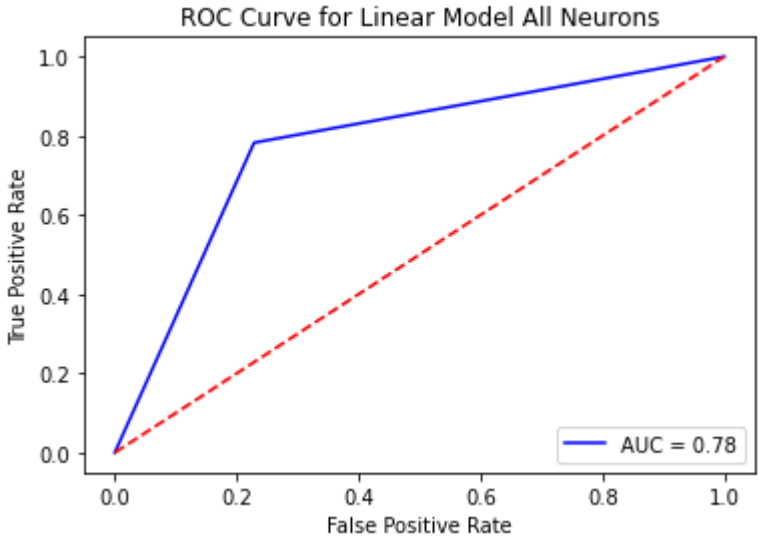



```
ypredictionslinearALLn = modelEvaluation(XVALID, YVALID, model_linearALLn)
```

```
              precision    recall  f1-score   support

           0       0.99      0.77      0.87      1404
           1       0.14      0.78      0.24        69

    accuracy                           0.77      1473
```

```
           macro avg       0.57      0.78      0.55      1473
        weighted avg       0.95      0.77      0.84      1473
```

```
rocauc_graph(YVALID, ypredictionslinearALLn, 'ROC Curve for Linear Model All Neurons')
```




```
prediction_linearALL = model_linearALLn.predict(XTRAIN)
accuracy_linearALL = accuracy_score(YTRAIN, prediction_linearALL.round())
precision_linearALL = precision_score(YTRAIN, prediction_linearALL.round(), average='micro')
recall_linearALL = recall_score(YTRAIN, prediction_linearALL.round(), average='micro')
f1score_linearALL = f1_score(YTRAIN, prediction_linearALL.round(), average='micro')

print(color.BOLD + "The Training Predictions:" + color.END)
print("Accuracy Score: %.2f%%" % (accuracy_linearALL * 100))
print("Precision Score: %.2f%%" % (precision_linearALL * 100))
print("Recall Score: %.2f%%" % (recall_linearALL * 100))
print("F1-score: %.4f" % (f1score_linearALL * 100 ))
```

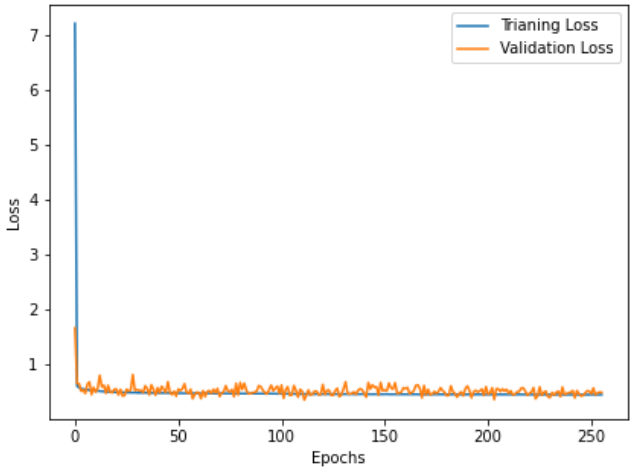
```
The Training Predictions:
Accuracy Score: 78.17%
Precision Score: 78.17%
Recall Score: 78.17%
F1-score: 78.1723
```

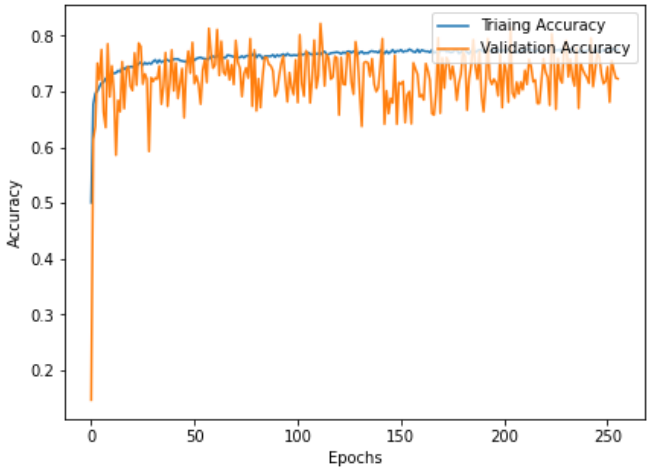
```
prediction_linearALL = model_linearALLn.predict(XVALID)
accuracy_linearALL = accuracy_score(YVALID, prediction_linearALL.round())
precision_linearALL = precision_score(YVALID, prediction_linearALL.round(), average='micro')
recall_linearALL = recall_score(YVALID, prediction_linearALL.round(), average='micro')
f1score_linearALL = f1_score(YVALID, prediction_linearALL.round(), average='micro')

print(color.BOLD + "The Validation Predictions:" + color.END)
print("Accuracy Score: %.2f%%" % (accuracy_linearALL * 100))
print("Precision Score: %.2f%%" % (precision_linearALL * 100))
```

```
print("Recall Score: %.2f%%" % (recall_linearALL * 100))
print("F1-score: %.4f" % (f1score_linearALL * 100 ))
```

The Validation Predictions:
Accuracy Score: 77.05%
Precision Score: 77.05%
Recall Score: 77.05%
F1-score: 77.0536

```
logistic_learning_curve(learningCURVEsigmoidLAST)
```

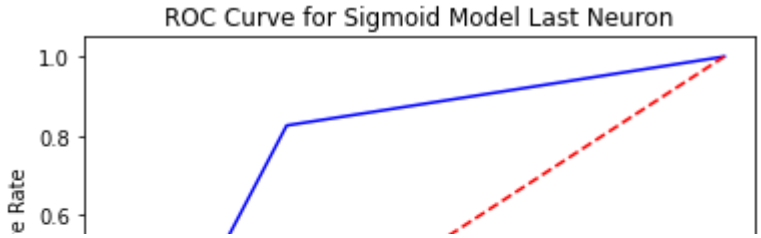





```
ypredictionssigmoidLASTn = modelEvaluation(XVALID, YVALID, model_sigmoidLASTn)
```

	precision	recall	f1-score	support
0	0.99	0.72	0.83	1404
1	0.13	0.83	0.22	69
accuracy			0.72	1473
macro avg	0.56	0.77	0.53	1473
weighted avg	0.95	0.72	0.80	1473

```
rocauc_graph(YVALID, ypredictionssigmoidLASTn, 'ROC Curve for Sigmoid Model Last Neuron')
```



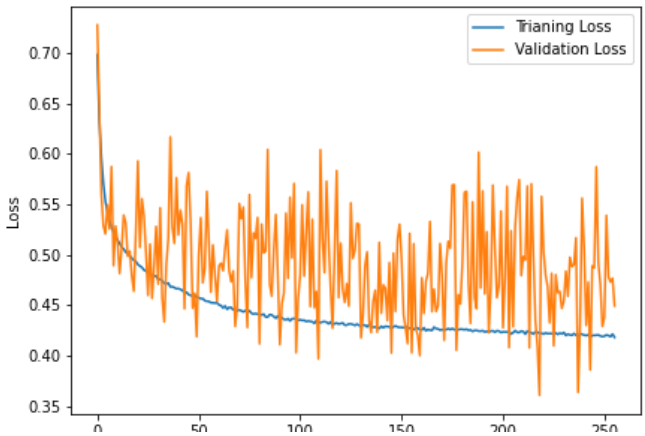

```
prediction_sigmoidLAST = model_sigmoidLASTn.predict(XTRAIN)
accuracy_sigmoidLAST = accuracy_score(YTRAIN, prediction_sigmoidLAST.round())
precision_sigmoidLAST = precision_score(YTRAIN, prediction_sigmoidLAST.round(), average='micr
recall_sigmoidLAST = recall_score(YTRAIN, prediction_sigmoidLAST.round(), average='micro')
f1score_sigmoidLAST = f1_score(YTRAIN, prediction_sigmoidLAST.round(), average='micro')

print(color.BOLD + "The Training Predictions:" + color.END)
print("Accuracy Score: %.2f%%" % (accuracy_sigmoidLAST * 100))
print("Precision Score: %.2f%%" % (precision_sigmoidLAST * 100))
print("Recall Score: %.2f%%" % (recall_sigmoidLAST * 100))
print("F1-score: %.4f" % (f1score_sigmoidLAST * 100 ))
```

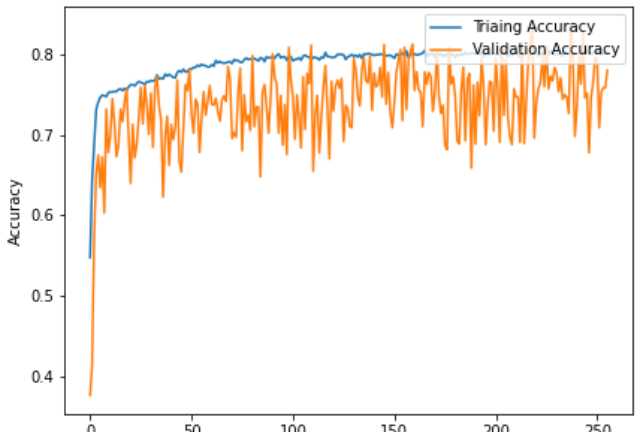
```
The Training Predictions:
Accuracy Score: 72.88%
Precision Score: 72.88%
Recall Score: 72.88%
F1-score: 72.8754
```

```
prediction_sigmoidLAST = model_sigmoidLASTn.predict(XVALID)
accuracy_sigmoidLAST = accuracy_score(YVALID, prediction_sigmoidLAST.round())
precision_sigmoidLAST = precision_score(YVALID, prediction_sigmoidLAST.round(), average='micr
recall_sigmoidLAST = recall_score(YVALID, prediction_sigmoidLAST.round(), average='micro')
f1score_sigmoidLAST = f1_score(YVALID, prediction_sigmoidLAST.round(), average='micro')

print(color.BOLD + "The Validation Predictions:" + color.END)
print("Accuracy Score: %.2f%%" % (accuracy_sigmoidLAST * 100))
print("Precision Score: %.2f%%" % (precision_sigmoidLAST * 100))
print("Recall Score: %.2f%%" % (recall_sigmoidLAST * 100))
print("F1-score: %.4f" % (f1score_sigmoidLAST * 100 ))
```

```
The Validation Predictions:
Accuracy Score: 72.30%
Precision Score: 72.30%
Recall Score: 72.30%
F1-score: 72.3014
```

```
logistic_learning_curve(learningCURVEsigmoidALL)
```

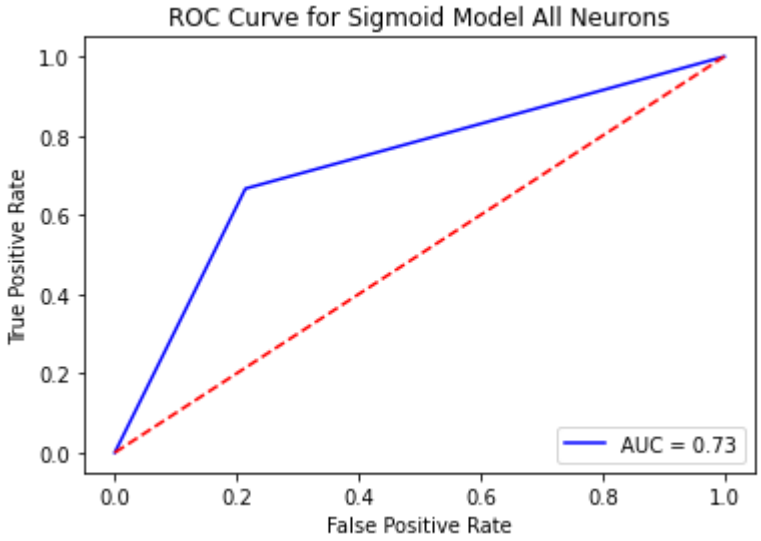





```
ypredictionssigmoidALLn = modelEvaluation(XVALID, YVALID, model_sigmoidALLn)
```

```
                precision    recall  f1-score   support

           0       0.98      0.79      0.87      1404
           1       0.13      0.67      0.22        69

    accuracy                           0.78      1473
   macro avg       0.56      0.73      0.55      1473
weighted avg       0.94      0.78      0.84      1473
```

```
rocauc_graph(YVALID, ypredictionssigmoidALLn, 'ROC Curve for Sigmoid Model All Neurons')
```




```
prediction_sigmoidALL = model_sigmoidALLn.predict(XTRAIN)
accuracy_sigmoidALL = accuracy_score(YTRAIN, prediction_sigmoidALL.round())
precision_sigmoidALL = precision_score(YTRAIN, prediction_sigmoidALL.round(), average='micro'
recall_sigmoidALL = recall_score(YTRAIN, prediction_sigmoidALL.round(), average='micro')
f1score_sigmoidALL = f1_score(YTRAIN, prediction_sigmoidALL.round(), average='micro')

print(color.BOLD + "The Training Predictions:" + color.END)
print("Accuracy Score: %.2f%%" % (accuracy_sigmoidALL * 100))
print("Precision Score: %.2f%%" % (precision_sigmoidALL * 100))
```

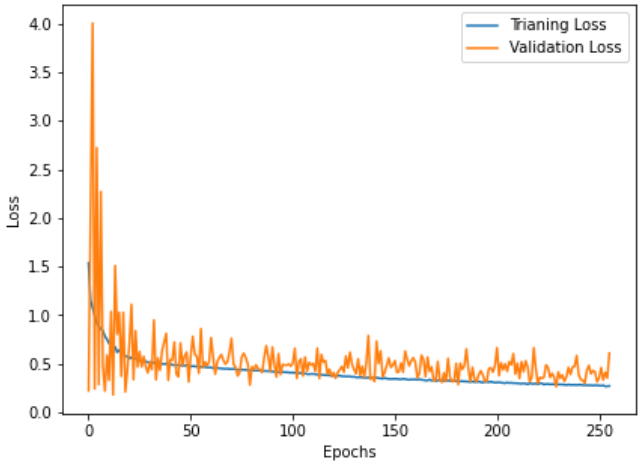
```
print("Recall Score: %.2f%%" % (recall_sigmoidALL * 100))
print("F1-score: %.4f" % (f1score_sigmoidALL * 100 ))
```

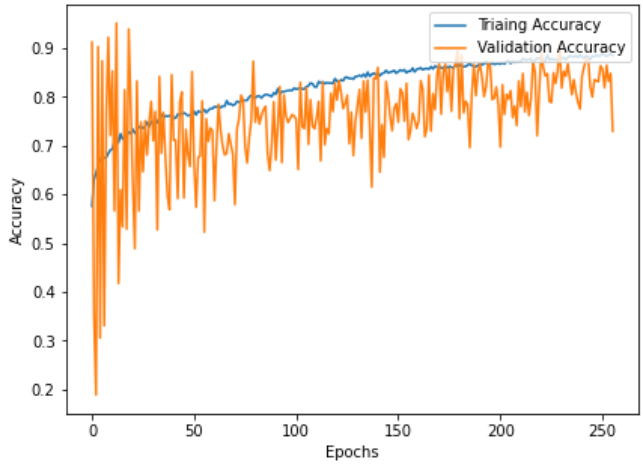
The Training Predictions:
Accuracy Score: 78.87%
Precision Score: 78.87%
Recall Score: 78.87%
F1-score: 78.8708

```
prediction_sigmoidALL = model_sigmoidALLn.predict(XVALID)
accuracy_sigmoidALL = accuracy_score(YVALID, prediction_sigmoidALL.round())
precision_sigmoidALL = precision_score(YVALID, prediction_sigmoidALL.round(), average='micro'
recall_sigmoidALL = recall_score(YVALID, prediction_sigmoidALL.round(), average='micro')
f1score_sigmoidALL = f1_score(YVALID, prediction_sigmoidALL.round(), average='micro')

print(color.BOLD + "The Validation Predictions:" + color.END)
print("Accuracy Score: %.2f%%" % (accuracy_sigmoidALL * 100))
print("Precision Score: %.2f%%" % (precision_sigmoidALL * 100))
print("Recall Score: %.2f%%" % (recall_sigmoidALL * 100))
print("F1-score: %.4f" % (f1score_sigmoidALL * 100 ))
```

The Validation Predictions:
Accuracy Score: 78.00%
Precision Score: 78.00%
Recall Score: 78.00%
F1-score: 78.0041

```
logistic_learning_curve(learningCURVEoverfit)
```



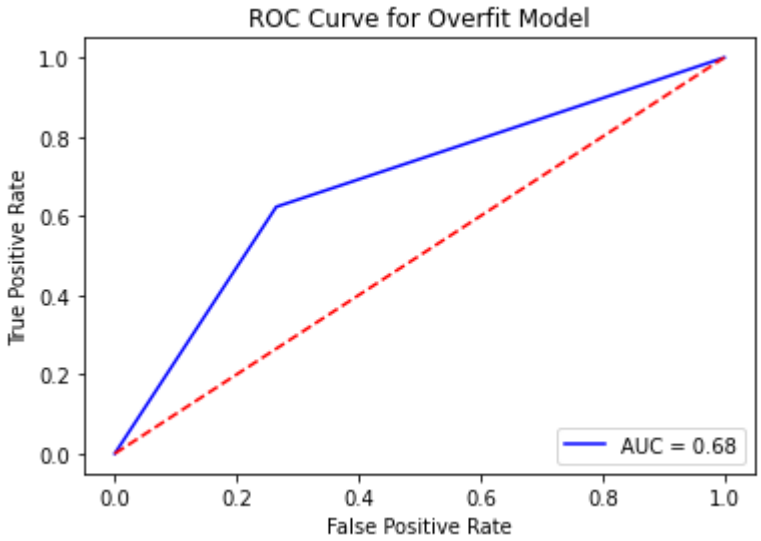



```
ypredictionsoverfit = modelEvaluation(XVALID, YVALID, model_overfit)
```

```
                precision    recall  f1-score   support
```

```
                0       0.98      0.74      0.84      1404
                1       0.10      0.62      0.18        69

         accuracy                           0.73      1473
        macro avg       0.54      0.68      0.51      1473
     weighted avg       0.93      0.73      0.81      1473
```

```
rocauc_graph(YVALID, ypredictionsoverfit, 'ROC Curve for Overfit Model')
```




```
prediction_overfit = model_overfit.predict(XTRAIN)
accuracy_overfit = accuracy_score(YTRAIN, prediction_overfit.round())
precision_overfit = precision_score(YTRAIN, prediction_overfit.round(), average='micro')
recall_overfit = recall_score(YTRAIN, prediction_overfit.round(), average='micro')
f1score_overfit = f1_score(YTRAIN, prediction_overfit.round(), average='micro')

print(color.BOLD + "The Training Predictions:" + color.END)
print("Accuracy Score: %.2f%%" % (accuracy_overfit * 100))
print("Precision Score: %.2f%%" % (precision_overfit * 100))
print("Recall Score: %.2f%%" % (recall_overfit * 100))
print("F1-score: %.4f" % (f1score_overfit * 100 ))
```

```
The Training Predictions:
Accuracy Score: 75.29%
Precision Score: 75.29%
Recall Score: 75.29%
F1-score: 75.2910
```

```
prediction_overfit = model_overfit.predict(XVALID)
accuracy_overfit = accuracy_score(YVALID, prediction_overfit.round())
precision_overfit = precision_score(YVALID, prediction_overfit.round(), average='micro')
recall_overfit = recall_score(YVALID, prediction_overfit.round(), average='micro')
f1score_overfit = f1_score(YVALID, prediction_overfit.round(), average='micro')

print(color.BOLD + "The Validaiton Predictions:" + color.END)
```

```
print("Accuracy Score: %.2f%%" % (accuracy_overfit * 100))
print("Precision Score: %.2f%%" % (precision_overfit * 100))
print("Recall Score: %.2f%%" % (recall_overfit * 100))
print("F1-score: %.4f" % (f1score_overfit * 100 ))
```

```
The Validaiton Predictions:
Accuracy Score: 72.98%
Precision Score: 72.98%
Recall Score: 72.98%
F1-score: 72.9803
```

✓ 0s completed at 10:52 PM