Week4_13_ANN_model_adv

May 1, 2021

Artificial Neural Network (ANN)

Import Libraries

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

import warnings
warnings.filterwarnings("ignore")
```

Plot function

[2]:
```
import matplotlib.pyplot as plt
plt.style.use('ggplot')

def plot_history(history):
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    x = range(1, len(acc) + 1)

    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    plt.plot(x, acc, 'b', label='Training acc')
    plt.plot(x, val_acc, 'r', label='Validation acc')
    plt.title('Training and validation accuracy')
    plt.legend()
    plt.subplot(1, 2, 2)
    plt.plot(x, loss, 'b', label='Training loss')
    plt.plot(x, val_loss, 'r', label='Validation loss')
    plt.title('Training and validation loss')
    plt.legend()
```

Import dataset

[4]:
```
dataset = pd.read_csv('/Users/preethamvignesh/Downloads/ML_classwork/Week4/
↪Churn_Modelling.csv')
```

```
dataset.head()
```

```
[4]:    RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age  \
0          1    15634602  Hargrave          619    France  Female   42
1          2    15647311      Hill          608     Spain  Female   41
2          3    15619304      Onio          502    France  Female   42
3          4    15701354      Boni          699    France  Female   39
4          5    15737888  Mitchell          850     Spain  Female   43

   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

Separating Dependent and Independent variables

```
[5]: X = dataset.iloc[:, 3:13].values
y = dataset.iloc[:, 13].values
```

Encoding categorical data

```
[6]: # Encoding categorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer

#encoding 'Geography' to number
labelencoder_X1 = LabelEncoder()
X[:, 1] = labelencoder_X1.fit_transform(X[:, 1])

#encoding 'Gender' to numbers
labelencoder_X2 = LabelEncoder()
X[:, 2] = labelencoder_X2.fit_transform(X[:, 2])

#encoding 'Geography' to one hot
ct = ColumnTransformer([("Geography", OneHotEncoder(), [1])], remainder =␣
↪'passthrough')
# onehotencoder = OneHotEncoder(categorical_features = [1])
X = ct.fit_transform(X)
```

```
# get rid of dummy variable trap
X = X[:, 1:]
```

Feature scaling

[7]:
```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

[8]:
```
from sklearn.preprocessing import MinMaxScaler
scalar = MinMaxScaler()
X= scalar.fit_transform(X)
```

Split Train and Test set

[9]:
```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

Model preparation with layers

[10]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout

model = Sequential()
# Adding the input layer and the first hidden layer with dropout
model.add(Dense(units = 22, activation='relu', input_dim = 11)) # Input layer␣
 ↪and first hidden layer
# when we assing `input_dim = 11`, we actually creating input layer
# model.add(Dropout(p=0.1))

# Adding the second hidden layer with dropout
# doesn't need the input_dim params
# kernel_initializer updates weights
# activation function - rectifier
model.add(Dense(units = 22, activation='relu')) # Second hidden layer
# model.add(Dropout(p=0.1))

# Adding the output layer
# dependent variable with more than two categories (3), output_dim needs to␣
 ↪change (e.g. 3), activation function - sufmax
model.add(Dense(1, kernel_initializer = 'glorot_uniform', activation =␣
 ↪'sigmoid' ))
model.add(Dense(units = 1, activation='sigmoid')) # Output layer
```

```
# Compiling the ANN - applying Stochastic Gradient Descent to whole ANN
# Several different SGD algorithms
# mathematical details based on the loss function
# binary_crossentropy, categorical_cross_entropy
model.compile(loss = 'binary_crossentropy', optimizer='adam', metrics =␣
↪['accuracy'])
```

[11]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 22)                264
_________________________________________________________________
dense_1 (Dense)              (None, 22)                506
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 23
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 2
=================================================================
Total params: 795
Trainable params: 795
Non-trainable params: 0
_________________________________________________________________
```

Fitting Model

[12]:
```
# Fitting fully connnected NN to the Training set
model.fit(X_train, y_train, batch_size = 25, epochs = 100)
```

```
Epoch 1/100
320/320 [==============================] - 1s 1ms/step - loss: 0.5992 -
accuracy: 0.8032
Epoch 2/100
320/320 [==============================] - 0s 1ms/step - loss: 0.5179 -
accuracy: 0.8004
Epoch 3/100
320/320 [==============================] - 0s 1ms/step - loss: 0.4919 -
accuracy: 0.7993
Epoch 4/100
320/320 [==============================] - 0s 1ms/step - loss: 0.4759 -
accuracy: 0.7948
Epoch 5/100
320/320 [==============================] - 0s 1ms/step - loss: 0.4527 -
accuracy: 0.8041
Epoch 6/100
320/320 [==============================] - 0s 1ms/step - loss: 0.4598 -
```

```
accuracy: 0.7911
Epoch 7/100
320/320 [==============================] - 0s 1ms/step - loss: 0.4529 -
accuracy: 0.7939
Epoch 8/100
320/320 [==============================] - 0s 1ms/step - loss: 0.4435 -
accuracy: 0.7956
Epoch 9/100
320/320 [==============================] - 0s 1ms/step - loss: 0.4390 -
accuracy: 0.7974
Epoch 10/100
320/320 [==============================] - 0s 1ms/step - loss: 0.4362 -
accuracy: 0.7959
Epoch 11/100
320/320 [==============================] - 0s 1ms/step - loss: 0.4374 -
accuracy: 0.7902
Epoch 12/100
320/320 [==============================] - 0s 1ms/step - loss: 0.4280 -
accuracy: 0.7964
Epoch 13/100
320/320 [==============================] - 0s 1ms/step - loss: 0.4306 -
accuracy: 0.7952
Epoch 14/100
320/320 [==============================] - 0s 1ms/step - loss: 0.4272 -
accuracy: 0.7872
Epoch 15/100
320/320 [==============================] - 0s 1ms/step - loss: 0.4098 -
accuracy: 0.7976
Epoch 16/100
320/320 [==============================] - 0s 1ms/step - loss: 0.4018 -
accuracy: 0.7969
Epoch 17/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3913 -
accuracy: 0.8061
Epoch 18/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3961 -
accuracy: 0.7984
Epoch 19/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3982 -
accuracy: 0.7966
Epoch 20/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3908 -
accuracy: 0.7988
Epoch 21/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3918 -
accuracy: 0.7997
Epoch 22/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3927 -
```

```
accuracy: 0.8027
Epoch 23/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3772 -
accuracy: 0.8296
Epoch 24/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3925 -
accuracy: 0.8342
Epoch 25/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3920 -
accuracy: 0.8400
Epoch 26/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3903 -
accuracy: 0.8394
Epoch 27/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3780 -
accuracy: 0.8472
Epoch 28/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3775 -
accuracy: 0.8436
Epoch 29/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3850 -
accuracy: 0.8407
Epoch 30/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3776 -
accuracy: 0.8507
Epoch 31/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3599 -
accuracy: 0.8541
Epoch 32/100
320/320 [==============================] - 0s 990us/step - loss: 0.3759 -
accuracy: 0.8456
Epoch 33/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3721 -
accuracy: 0.8497
Epoch 34/100
320/320 [==============================] - 0s 988us/step - loss: 0.3622 -
accuracy: 0.8546
Epoch 35/100
320/320 [==============================] - 0s 992us/step - loss: 0.3669 -
accuracy: 0.8516
Epoch 36/100
320/320 [==============================] - 0s 993us/step - loss: 0.3654 -
accuracy: 0.8505
Epoch 37/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3652 -
accuracy: 0.8545
Epoch 38/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3710 -
```

```
accuracy: 0.8556
Epoch 39/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3528 -
accuracy: 0.8596
Epoch 40/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3583 -
accuracy: 0.8584
Epoch 41/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3665 -
accuracy: 0.8500
Epoch 42/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3591 -
accuracy: 0.8577
Epoch 43/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3595 -
accuracy: 0.8529
Epoch 44/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3643 -
accuracy: 0.8519
Epoch 45/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3679 -
accuracy: 0.8528
Epoch 46/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3550 -
accuracy: 0.8557
Epoch 47/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3583 -
accuracy: 0.8571
Epoch 48/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3573 -
accuracy: 0.8571
Epoch 49/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3625 -
accuracy: 0.8551
Epoch 50/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3491 -
accuracy: 0.8563
Epoch 51/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3481 -
accuracy: 0.8632
Epoch 52/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3372 -
accuracy: 0.8673
Epoch 53/100
320/320 [==============================] - 0s 995us/step - loss: 0.3644 -
accuracy: 0.8500
Epoch 54/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3409 -
```

```
accuracy: 0.8649
Epoch 55/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3614 -
accuracy: 0.8500
Epoch 56/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3575 -
accuracy: 0.8531
Epoch 57/100
320/320 [==============================] - 0s 973us/step - loss: 0.3417 -
accuracy: 0.8611
Epoch 58/100
320/320 [==============================] - 0s 988us/step - loss: 0.3550 -
accuracy: 0.8566
Epoch 59/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3476 -
accuracy: 0.8592
Epoch 60/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3516 -
accuracy: 0.8585
Epoch 61/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3456 -
accuracy: 0.8634
Epoch 62/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3531 -
accuracy: 0.8620
Epoch 63/100
320/320 [==============================] - 0s 973us/step - loss: 0.3429 -
accuracy: 0.8629
Epoch 64/100
320/320 [==============================] - 0s 976us/step - loss: 0.3503 -
accuracy: 0.8613
Epoch 65/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3554 -
accuracy: 0.8550
Epoch 66/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3440 -
accuracy: 0.8630
Epoch 67/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3391 -
accuracy: 0.8669
Epoch 68/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3490 -
accuracy: 0.8597
Epoch 69/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3372 -
accuracy: 0.8642
Epoch 70/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3378 -
```

```
accuracy: 0.8663
Epoch 71/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3364 -
accuracy: 0.8653
Epoch 72/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3410 -
accuracy: 0.8629
Epoch 73/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3388 -
accuracy: 0.8677
Epoch 74/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3317 -
accuracy: 0.8703
Epoch 75/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3468 -
accuracy: 0.8603
Epoch 76/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3413 -
accuracy: 0.8591
Epoch 77/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3429 -
accuracy: 0.8639
Epoch 78/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3400 -
accuracy: 0.8582
Epoch 79/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3330 -
accuracy: 0.8654
Epoch 80/100
320/320 [==============================] - 0s 996us/step - loss: 0.3348 -
accuracy: 0.8683
Epoch 81/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3425 -
accuracy: 0.8587
Epoch 82/100
320/320 [==============================] - 0s 976us/step - loss: 0.3413 -
accuracy: 0.8587
Epoch 83/100
320/320 [==============================] - 0s 985us/step - loss: 0.3376 -
accuracy: 0.8661
Epoch 84/100
320/320 [==============================] - 0s 971us/step - loss: 0.3427 -
accuracy: 0.8622
Epoch 85/100
320/320 [==============================] - 0s 986us/step - loss: 0.3329 -
accuracy: 0.8665
Epoch 86/100
320/320 [==============================] - 0s 980us/step - loss: 0.3393 -
```

```
accuracy: 0.8616
Epoch 87/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3317 -
accuracy: 0.8618
Epoch 88/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3383 -
accuracy: 0.8601
Epoch 89/100
320/320 [==============================] - 0s 980us/step - loss: 0.3366 -
accuracy: 0.8610
Epoch 90/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3264 -
accuracy: 0.8701
Epoch 91/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3351 -
accuracy: 0.8648
Epoch 92/100
320/320 [==============================] - 0s 995us/step - loss: 0.3344 -
accuracy: 0.8654
Epoch 93/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3285 -
accuracy: 0.8711
Epoch 94/100
320/320 [==============================] - 0s 979us/step - loss: 0.3390 -
accuracy: 0.8643
Epoch 95/100
320/320 [==============================] - 0s 974us/step - loss: 0.3320 -
accuracy: 0.8632
Epoch 96/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3306 -
accuracy: 0.8680
Epoch 97/100
320/320 [==============================] - 0s 982us/step - loss: 0.3366 -
accuracy: 0.8616
Epoch 98/100
320/320 [==============================] - 0s 972us/step - loss: 0.3455 -
accuracy: 0.8579
Epoch 99/100
320/320 [==============================] - 0s 1ms/step - loss: 0.3427 -
accuracy: 0.8569
Epoch 100/100
320/320 [==============================] - 0s 990us/step - loss: 0.3331 -
accuracy: 0.8653
```

```
[12]: <tensorflow.python.keras.callbacks.History at 0x135b660a0>
```

Model Evaluate

[13]:
```python
model.evaluate(X_test,y_test)
```

```
63/63 [==============================] - 0s 924us/step - loss: 0.3585 -
accuracy: 0.8470
```

[13]:
```
[0.3585488796234131, 0.847000002861023]
```

[14]:
```python
# Predicting on the Test set
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5)
```

[15]:
```python
# Get acurracy on Test set
from sklearn.metrics import confusion_matrix
cm_test = confusion_matrix(y_test, y_pred)
cm_test
```

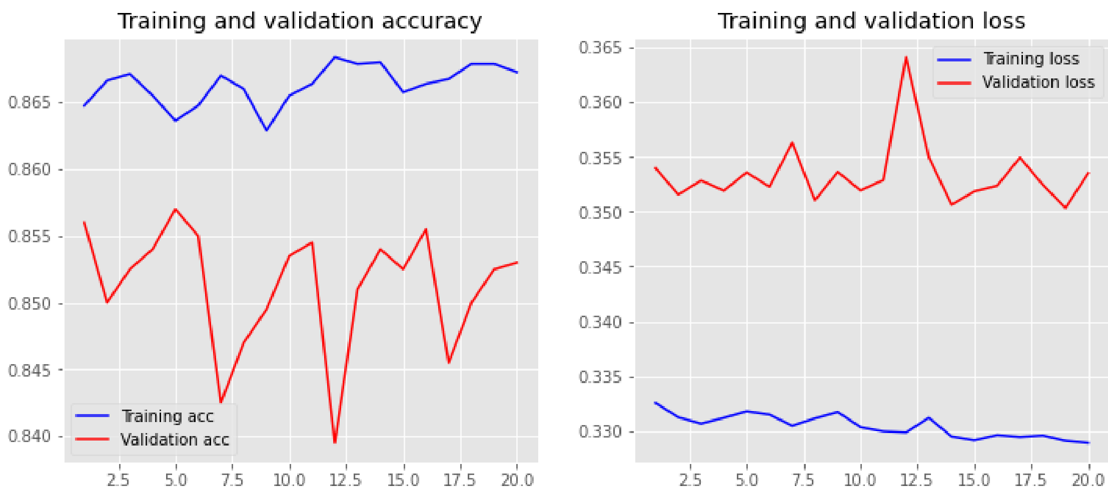
[15]:
```
array([[1476,  113],
       [ 193,  218]])
```

[16]:
```python
history = model.fit(X_train, y_train,
                    epochs=20,
                    verbose=True,
                    validation_data=(X_test, y_test),
                    batch_size=52)

loss, accuracy = model.evaluate(X_train, y_train, verbose=False)
print("Training Accuracy: {:.4f}".format(accuracy))
loss, accuracy = model.evaluate(X_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))
plot_history(history)
```

```
Epoch 1/20
154/154 [==============================] - 1s 5ms/step - loss: 0.3325 -
accuracy: 0.8648 - val_loss: 0.3540 - val_accuracy: 0.8560
Epoch 2/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3312 -
accuracy: 0.8666 - val_loss: 0.3516 - val_accuracy: 0.8500
Epoch 3/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3306 -
accuracy: 0.8671 - val_loss: 0.3529 - val_accuracy: 0.8525
Epoch 4/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3312 -
accuracy: 0.8655 - val_loss: 0.3519 - val_accuracy: 0.8540
Epoch 5/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3318 -
accuracy: 0.8636 - val_loss: 0.3536 - val_accuracy: 0.8570
Epoch 6/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3315 -
```

```
accuracy: 0.8648 - val_loss: 0.3523 - val_accuracy: 0.8550
Epoch 7/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3305 -
accuracy: 0.8670 - val_loss: 0.3563 - val_accuracy: 0.8425
Epoch 8/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3311 -
accuracy: 0.8660 - val_loss: 0.3511 - val_accuracy: 0.8470
Epoch 9/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3317 -
accuracy: 0.8629 - val_loss: 0.3537 - val_accuracy: 0.8495
Epoch 10/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3303 -
accuracy: 0.8655 - val_loss: 0.3520 - val_accuracy: 0.8535
Epoch 11/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3299 -
accuracy: 0.8664 - val_loss: 0.3529 - val_accuracy: 0.8545
Epoch 12/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3298 -
accuracy: 0.8684 - val_loss: 0.3641 - val_accuracy: 0.8395
Epoch 13/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3312 -
accuracy: 0.8679 - val_loss: 0.3550 - val_accuracy: 0.8510
Epoch 14/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3295 -
accuracy: 0.8680 - val_loss: 0.3506 - val_accuracy: 0.8540
Epoch 15/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3291 -
accuracy: 0.8658 - val_loss: 0.3519 - val_accuracy: 0.8525
Epoch 16/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3296 -
accuracy: 0.8664 - val_loss: 0.3524 - val_accuracy: 0.8555
Epoch 17/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3294 -
accuracy: 0.8668 - val_loss: 0.3550 - val_accuracy: 0.8455
Epoch 18/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3296 -
accuracy: 0.8679 - val_loss: 0.3525 - val_accuracy: 0.8500
Epoch 19/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3291 -
accuracy: 0.8679 - val_loss: 0.3504 - val_accuracy: 0.8525
Epoch 20/20
154/154 [==============================] - 0s 2ms/step - loss: 0.3289 -
accuracy: 0.8673 - val_loss: 0.3535 - val_accuracy: 0.8530
Training Accuracy: 0.8705
Testing Accuracy:  0.8530
```



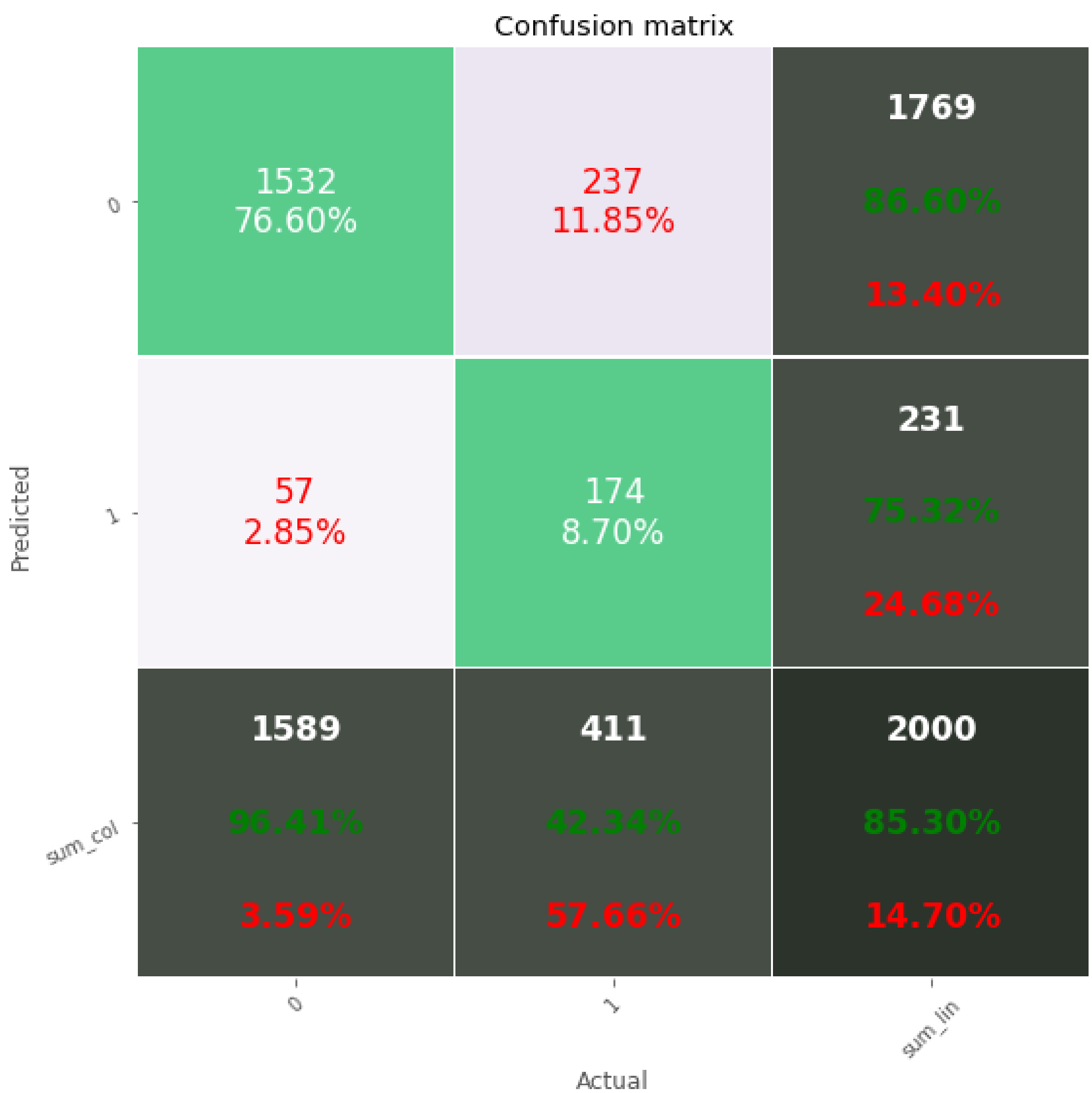
[17]:
```
y_pred = model.predict(X_test)
y_pred = y_pred > 0.5

con_res = confusion_matrix(y_test, y_pred)
print("Confusion matrix:")
print(confusion_matrix(y_test,y_pred))
print("Accuracy: {:.2f}%".format(accuracy_score(y_test, y_pred)*100))
```

```
Confusion matrix:
[[1532   57]
 [ 237  174]]
Accuracy: 85.30%
```

Metrix: Confusion Matrix

[19]:
```
%run -i '/Users/preethamvignesh/Desktop/Work/ML_EIT/
 ↪confusion_matrix_different_ways.py'
df_confmatrx = pd.DataFrame(con_res, range(2),range(2))
df_confmatrx
cmap = 'PuRd'
confusion_matrix_dfrntway(df_confmatrx, cmap=cmap,fz=17)
```



Predicting single observation

```
[18]: #  Predicting a single new observation
new_prediction = model.predict(sc.transform(np.array([[0, 0, 600, 1, 40, 3,␣
 ↪60000, 2, 1, 1, 50000]])))
new_prediction = (new_prediction > 0.5)
print(new_prediction)
```

```
[[False]]
```

```
[ ]:
```

```
[ ]:
```