In [2]:
```
import numpy as np
import pandas as pd
pd.set_option('display.max_columns', None)
import seaborn as sns
import matplotlib.pyplot as plt
import os
from os import path
# import sklearn
from imblearn.over_sampling import SMOTE
import smote_variants as sv
```

Load data

In [3]:
```
classfication = pd.read_csv('classfication.csv',encoding='ISO-8859-1',index_col=0)
```

Variable

Inputs:

AGE_YRS: Age
SEX: Sex
V_ADMINBY: Vaccine Administered at
VAX_MANU: Vaccine Manufacturer
allergies_summary: Allergies to medications, food, or other products
Covid_involved: COVID-19 test result after taking vaccine
CUR_ILL: Bool value. Weather the vaccine recipient has illness currently
HISTORY_summary: Bool value. Pre-existing Conditions

Medication: Medication information from patients

Output:

Serious:Bool value (Yes, No)

Encoding

SEX: Male=0, Female=1

allergies_summary: None or Unknown=0, Allergies=1

Covid_involved: Unknown or Test Negative=0, Test Positive=1

CUR_ILL: No=0, Yes=1

HISTORY_summary: None or Unknown=0, Pre-existing Conditions=1

VAX_MANU: PFIZER\BIONTECH=0, MODERNA=1

Serious: No=0, Yes=1

Medication: No=0, Yes=1

In [4]:
```
final_data = classfication.drop(['VAERS_ID'], axis=1)
clean_data = classfication.copy()
final_data['SEX'] = clean_data['SEX'].apply(lambda x: 0 if x=='Male' else 1)
final_data['allergies_summary'] = clean_data['allergies_summary'].apply(lambda x: 0 if x=='None or Unknown' els
e 1)
final_data['Prior_Covid'] = clean_data['Prior_Covid'].apply(lambda x: 0 if x=='No Known Prior Covid' else 1)
final_data['current_ill'] = clean_data['current_ill'].apply(lambda x: 0 if x=='None or Unknown' else 1)
final_data['HISTORY_summary'] = clean_data['HISTORY_summary'].apply(lambda x: 0 if x=='None or Unknown' else 1)
final_data['vax_manu'] = clean_data['vax_manu'].apply(lambda x: 0 if x=='PFIZER\BIONTECH' else 1)
final_data['Outcome'] = clean_data['Outcome'].apply(lambda x: 0 if x=='Non-serious' else 1)
# for c in list_symptom:
#     final_data[c] = clean_data[c].apply(lambda x: 0 if x=='NO' else 1)
# # final_data[Serious].unique()
```

In [5]:
```
one_hot_V_ADMINBY = pd.get_dummies(final_data['V_ADMINBY'], prefix='V_ADMINBY').values
AGE_YRS = final_data['AGE_YRS'].values.reshape(-1,1)
SEX = final_data['SEX'].values.reshape(-1,1)
allergies_summary = final_data['allergies_summary'].values.reshape(-1,1)
Covid_involved = final_data['Prior_Covid'].values.reshape(-1,1)
CUR_ILL = final_data['current_ill'].values.reshape(-1,1)
HISTORY_summary = final_data['HISTORY_summary'].values.reshape(-1,1)
VAX_MANU = final_data['vax_manu'].values.reshape(-1,1)
medication = final_data.loc[:, 'Thyroid supplement': 'Pain medication/opioids'].values
Serious = final_data['Outcome'].values.reshape(-1,1)
```

In [281]:
```
X = np.concatenate((one_hot_V_ADMINBY,
                    AGE_YRS,
                    SEX,
                    allergies_summary,
                    Covid_involved,
                    CUR_ILL,
                    HISTORY_summary,
                    VAX_MANU,
                    medication), axis=1)
y = Serious
assert len(X) == len(y)
```

Useful metrics

A few metrics are utilized to evaluate the model:

False negatives and **false** positives are samples that were incorrectly classified

True negatives and **true** positives are samples that were correctly classified

Accuracy is the percentage of examples correctly classified > $\frac{True\ sample}{Total\ sample}$

Precision is the percentage of predicted positives that were correctly classified > $\frac{True\ positives}{True\ positives+False\ positives}$

Recall is the percentage of actual positives that were correctly classified > $\frac{True\ postives}{True\ postives+False\ negtives}$

AUC refers to the Area Under the Curve of a Receiver Operating Characteristic curve (ROC-AUC). This metric is equal to the probability that a classifier will rank a random positive sample higher than a random negative sample.

In [14]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, Dropout
```

In [15]:
```
def plot_metrics(history):
    metrics = ['accuracy', 'auc', 'precision', 'recall']
    colors = plt.rcParams['axes.prop_cycle'].by_key()['color']
    plt.figure(figsize=(15,10))
    for n, metric in enumerate(metrics):
        name = metric.replace("_"," ").capitalize()
        plt.subplot(2,2,n+1)
        plt.plot(history.epoch, history.history[metric], color="#374E55FF", label='Train')
        plt.plot(history.epoch, history.history['val_'+metric],
                 color="#B24745FF", linestyle="--", label='Val')
        plt.xlabel('Epoch')
        plt.ylabel(name)
        plt.legend()
#         plt.tight_layout()
def plot_cm(labels, predictions, p=0.5):
    cm = confusion_matrix(labels, predictions > p)
    plt.figure(figsize=(5,5))
    colormap = sns.diverging_palette(194, 1, s=44.1, l=48.4, as_cmap=True)
    sns.heatmap(cm, annot=True, fmt="d", cmap=colormap, annot_kws={"fontsize":15})
    plt.title('Confusion matrix @{:.2f}'.format(p), fontsize=15)
    plt.ylabel('Actual label', fontsize=15)
    plt.xlabel('Predicted label', fontsize=15)
    plt.xticks(fontsize=15)
    plt.yticks(fontsize=15)

    print('Legitimate Transactions Detected (True Negatives): ', cm[0][0])
    print('Legitimate Transactions Incorrectly Detected (False Positives): ', cm[0][1])
    print('Fraudulent Transactions Missed (False Negatives): ', cm[1][0])
    print('Fraudulent Transactions Detected (True Positives): ', cm[1][1])
    print('Total Fraudulent Transactions: ', np.sum(cm[1]))
```

In [211]:
```
# def create_model(node = 128,
#                  activation = 'relu',
#                  dropout_rate = 0,
#                  init_mode = 'uniform',
#                  optimizer='adam'):

# #     weight_constraint = 0
# #     optimizer = keras.optimizers.Adam
# #     lr = 0.01
# #     momemntum = 0
#     METRICS = [
#       keras.metrics.TruePositives(name='tp'),
#       keras.metrics.FalsePositives(name='fp'),
#       keras.metrics.TrueNegatives(name='tn'),
#       keras.metrics.FalseNegatives(name='fn'),
#       keras.metrics.BinaryAccuracy(name='accuracy'),
#       keras.metrics.Precision(name='precision'),
#       keras.metrics.Recall(name='recall'),
#       keras.metrics.AUC(name='auc')]
#     model = Sequential()
#     model.add(Dense(units = node, kernel_initializer = init_mode, activation = activation, input_dim = X_trai
n.shape[1]))
#     model.add(Dropout(dropout_rate))
#     model.add(Dense(units = node, kernel_initializer = init_mode, activation = activation))
#     model.add(Dropout(dropout_rate))
#     model.add(Dense(units = node, kernel_initializer = init_mode, activation = activation))
#     model.add(Dense(units = y_train.shape[1], kernel_initializer = init_mode, activation = 'sigmoid'))
#     model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=["accuracy"])
#     return model
```

In [212]:
```
# from sklearn.model_selection import GridSearchCV
# from keras.wrappers.scikit_learn import KerasClassifier
```

In [213]:
```
# estimator = KerasClassifier(build_fn=create_model, batch_size=128, epochs=200)
```

In [215]:
```
# kfold_splits = 3

# # epochs = 100
# # batch_size = 128
# param_grid = {
#     'node': [64,128,256],
#     'init_mode': ['normal'],
#     'optimizer': ['RMSprop', 'Adam', 'sgd'],
#     'dropout_rate': [, 0.8],
#     'epochs': [100],
#     'batch_size': [128]
# }
# grid = GridSearchCV(estimator=estimator, cv=kfold_splits, param_grid=param_grid, n_jobs=-1,verbose=1)
# grid_result = grid.fit(X, y)
```

In [320]:
```
# print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
# means = grid_result.cv_results_['mean_test_score']
# stds = grid_result.cv_results_['std_test_score']
# params = grid_result.cv_results_['params']
```

In [271]:
```
METRICS = [
      keras.metrics.TruePositives(name='tp'),
      keras.metrics.FalsePositives(name='fp'),
      keras.metrics.TrueNegatives(name='tn'),
      keras.metrics.FalseNegatives(name='fn'),
      keras.metrics.BinaryAccuracy(name='accuracy'),
      keras.metrics.Precision(name='precision'),
      keras.metrics.Recall(name='recall'),
      keras.metrics.AUC(name='auc'),
]
# METRICS = [keras.metrics.BinaryAccuracy(name='accuracy'),keras.metrics.AUC(name='auc')]
model = Sequential()

model.add(Dense(units = 128, kernel_initializer = 'uniform', activation = 'relu', input_dim = X_train.shape[1
]))
model.add(Dropout(0.80))
model.add(Dense(units = 256, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dropout(0.80))
model.add(Dense(units = 128, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))
model.summary()

model.compile(loss=keras.losses.BinaryCrossentropy(),
              optimizer='RMSprop',
              metrics=METRICS)
```

```
Model: "sequential_73"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_292 (Dense)            (None, 128)               4864
_________________________________________________________________
dropout_146 (Dropout)        (None, 128)               0
_________________________________________________________________
dense_293 (Dense)            (None, 256)               33024
_________________________________________________________________
dropout_147 (Dropout)        (None, 256)               0
_________________________________________________________________
dense_294 (Dense)            (None, 128)               32896
_________________________________________________________________
dense_295 (Dense)            (None, 1)                 129
=================================================================
Total params: 70,913
Trainable params: 70,913
Non-trainable params: 0
_________________________________________________________________
```

In [290]:
```
METRICS = [
      keras.metrics.TruePositives(name='tp'),
      keras.metrics.FalsePositives(name='fp'),
      keras.metrics.TrueNegatives(name='tn'),
      keras.metrics.FalseNegatives(name='fn'),
      keras.metrics.BinaryAccuracy(name='accuracy'),
      keras.metrics.Precision(name='precision'),
      keras.metrics.Recall(name='recall'),
      keras.metrics.AUC(name='auc'),
]
# METRICS = [keras.metrics.BinaryAccuracy(name='accuracy'),keras.metrics.AUC(name='auc')]
model = Sequential()

model.add(Dense(units = 128, kernel_initializer = 'uniform', activation = 'relu', input_dim = X_train.shape[1
]))
model.add(Dropout(0.8))
model.add(Dense(units = 256, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dropout(0.8))
model.add(Dense(units = 128, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))
model.summary()

model.compile(loss=keras.losses.BinaryCrossentropy(),
              optimizer='Adam',
              metrics=METRICS)
```

```
Model: "sequential_79"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_316 (Dense)            (None, 128)               4864
_________________________________________________________________
dropout_158 (Dropout)        (None, 128)               0
_________________________________________________________________
dense_317 (Dense)            (None, 256)               33024
_________________________________________________________________
dropout_159 (Dropout)        (None, 256)               0
_________________________________________________________________
dense_318 (Dense)            (None, 128)               32896
_________________________________________________________________
dense_319 (Dense)            (None, 1)                 129
=================================================================
Total params: 70,913
Trainable params: 70,913
Non-trainable params: 0
_________________________________________________________________
```

In [291]:
```
from sklearn.model_selection import train_test_split

oversample = SMOTE()
X_sample, y_sample = oversample.fit_resample(X, y)
X_train1, X_test, y_train1, y_test = train_test_split(X_sample, y_sample, test_size=0.20, random_state=3)
X_train, X_val, y_train, y_val = train_test_split(X_train1, y_train1, test_size=0.20, random_state=1)

batch_size = 128
num_epochs = 200

history = model.fit(X_train, y_train,
                        batch_size=batch_size,
                        epochs=num_epochs,
                        verbose=0,
                   validation_data=(X_val, y_val))
plot_metrics(history)
```

In [293]:
```
from sklearn.metrics import confusion_matrix

baseline_result = model.evaluate(X_val, y_val, batch_size=batch_size,verbose=0)
for name, value in zip(model.metrics_names, baseline_result):
    print(name, ': ', value)
test_predictions_baseline = model.predict(X_val)
plot_cm(y_val, test_predictions_baseline)
```

```
loss :  0.6530099511146545
tp :  470.0
fp :  301.0
tn :  541.0
fn :  315.0
accuracy :  0.6213890314102173
precision :  0.60959792137146
recall :  0.5987260937690735
auc :  0.6589081287384033
Legitimate Transactions Detected (True Negatives):  541
Legitimate Transactions Incorrectly Detected (False Positives):  301
Fraudulent Transactions Missed (False Negatives):  315
Fraudulent Transactions Detected (True Positives):  470
Total Fraudulent Transactions:  785
```