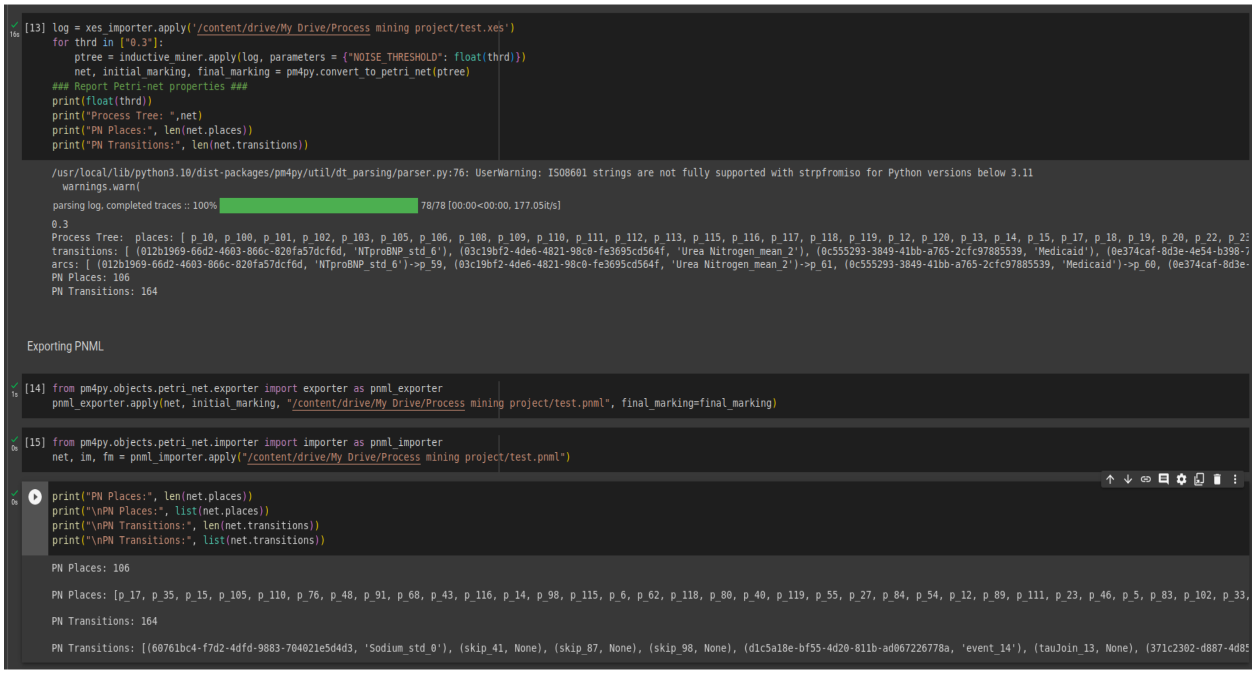
Juan Uriarte
CPSC 483-01 13657

Process Mining Project HW 1

Part 1

0.3 thrd



```
[13] log = xes_importer.apply('/content/drive/My Drive/Process mining project/test.xes')
     for thrd in ["0.3"]:
         ptree = inductive_miner.apply(log, parameters = {"NOISE_THRESHOLD": float(thrd)})
         net, initial_marking, final_marking = pm4py.convert_to_petri_net(ptree)
     ### Report Petri-net properties ###
     print(float(thrd))
     print("Process Tree: ",net)
     print("PN Places:", len(net.places))
     print("PN Transitions:", len(net.transitions))
```

```
/usr/local/lib/python3.10/dist-packages/pm4py/util/dt_parsing/parser.py:76: UserWarning: ISO8601 strings are not fully supported with strpfromiso for Python versions below 3.11
  warnings.warn(
parsing log, completed traces :: 100% 78/78 [00:00<00:00, 177.05it/s]
0.3
Process Tree:  places: [ p_10, p_100, p_101, p_102, p_103, p_105, p_106, p_108, p_109, p_110, p_111, p_112, p_113, p_115, p_116, p_117, p_118, p_119, p_12, p_120, p_13, p_14, p_15, p_17, p_18, p_19, p_20, p_22, p_23
transitions: [ (012b1969-66d2-4603-866c-820fa57dcf6d, 'NTproBNP_std_6'), (03c19bf2-4de6-4821-98c0-fe3695cd564f, 'Urea Nitrogen_mean_2'), (0c555293-3849-41bb-a765-2cfc97885539, 'Medicaid'), (0e374caf-8d3e-4e54-b398-7
arcs: [ (012b1969-66d2-4603-866c-820fa57dcf6d, 'NTproBNP_std_6')->p_59, (03c19bf2-4de6-4821-98c0-fe3695cd564f, 'Urea Nitrogen_mean_2')->p_61, (0c555293-3849-41bb-a765-2cfc97885539, 'Medicaid')->p_60, (0e374caf-8d3e-
PN Places: 106
PN Transitions: 164
```

Exporting PNML

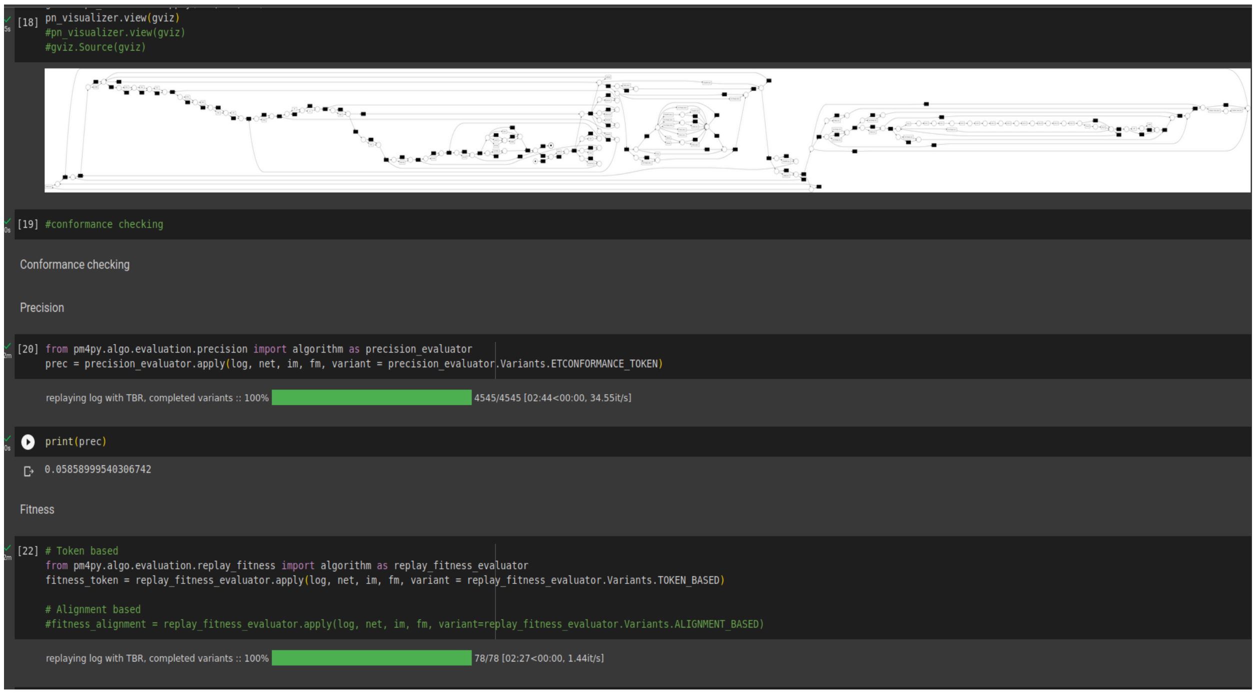
```
[14] from pm4py.objects.petri_net.exporter import exporter as pnml_exporter
     pnml_exporter.apply(net, initial_marking, "/content/drive/My Drive/Process mining project/test.pnml", final_marking=final_marking)
```

```
[15] from pm4py.objects.petri_net.importer import importer as pnml_importer
     net, im, fm = pnml_importer.apply("/content/drive/My Drive/Process mining project/test.pnml")
```

```
print("PN Places:", len(net.places))
print("\nPN Places:", list(net.places))
print("\nPN Transitions:", len(net.transitions))
print("\nPN Transitions:", list(net.transitions))
```

```
PN Places: 106

PN Places: [p_17, p_35, p_15, p_105, p_110, p_76, p_48, p_91, p_68, p_43, p_116, p_14, p_98, p_115, p_6, p_62, p_118, p_80, p_40, p_119, p_55, p_27, p_84, p_54, p_12, p_89, p_111, p_23, p_46, p_5, p_83, p_102, p_33,

PN Transitions: 164

PN Transitions: [(60761bc4-f7d2-4dfd-9883-704021e5d4d3, 'Sodium_std_0'), (skip_41, None), (skip_87, None), (skip_98, None), (d1c5a18e-bf55-4d20-811b-ad067226778a, 'event_14'), (tauJoin_13, None), (371c2302-d887-4d85
```

0.3 graph



```
[18] pn_visualizer.view(gviz)
     #pn_visualizer.view(gviz)
     #gviz.Source(gviz)
```

```
[19] #conformance checking
```

Conformance checking

Precision

```
[20] from pm4py.algo.evaluation.precision import algorithm as precision_evaluator
     prec = precision_evaluator.apply(log, net, im, fm, variant = precision_evaluator.Variants.ETCONFORMANCE_TOKEN)
```

```
replaying log with TBR, completed variants :: 100% 4545/4545 [02:44<00:00, 34.55it/s]
```

```
print(prec)
```

```
0.05858999540306742
```

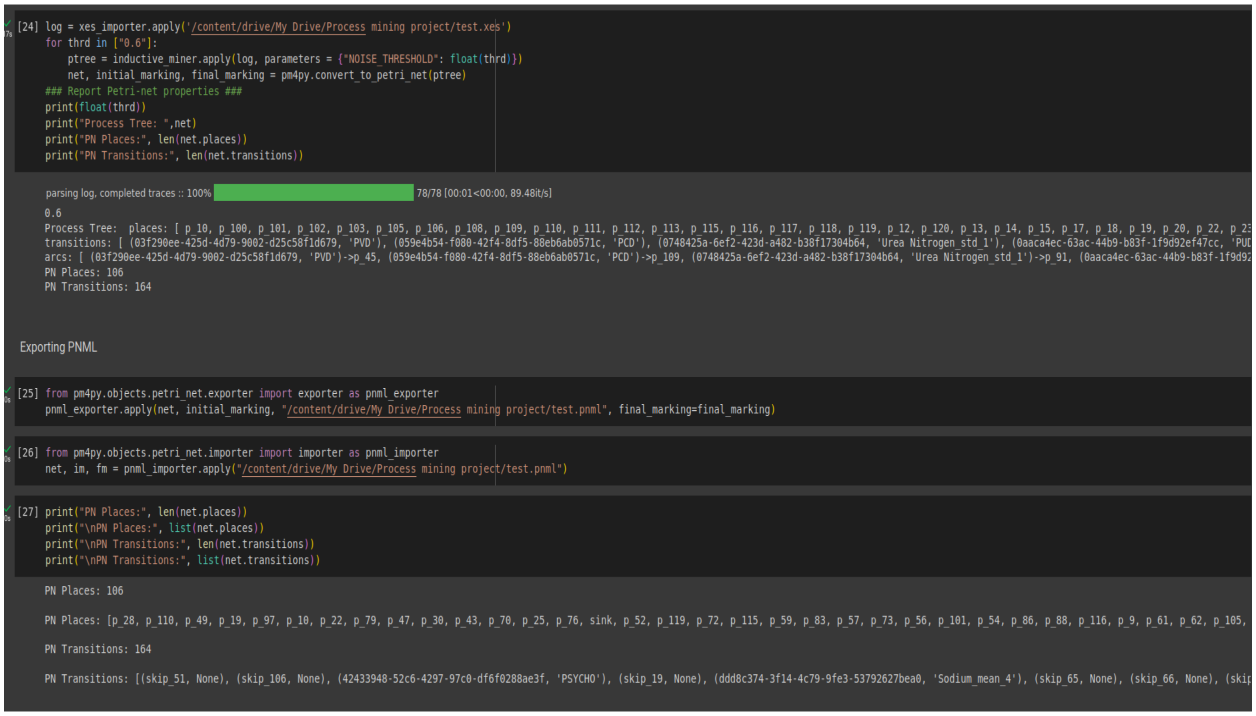
Fitness

```
[22] # Token based
     from pm4py.algo.evaluation.replay_fitness import algorithm as replay_fitness_evaluator
     fitness_token = replay_fitness_evaluator.apply(log, net, im, fm, variant = replay_fitness_evaluator.Variants.TOKEN_BASED)

     # Alignment based
     #fitness_alignment = replay_fitness_evaluator.apply(log, net, im, fm, variant=replay_fitness_evaluator.Variants.ALIGNMENT_BASED)
```

```
replaying log with TBR, completed variants :: 100% 78/78 [02:27<00:00, 1.44it/s]
```

Juan Uriarte
CPSC 483-01 13657

0.6 thrd



```
[24] log = xes_importer.apply('/content/drive/My Drive/Process mining project/test.xes')
     for thrd in ["0.6"]:
         ptree = inductive_miner.apply(log, parameters = {"NOISE_THRESHOLD": float(thrd)})
         net, initial_marking, final_marking = pm4py.convert_to_petri_net(ptree)
     ### Report Petri-net properties ###
     print(float(thrd))
     print("Process Tree: ",net)
     print("PN Places:", len(net.places))
     print("PN Transitions:", len(net.transitions))

     parsing log, completed traces :: 100% 78/78 [00:01<00:00, 89.48it/s]
     0.6
     Process Tree:  places: [ p_10, p_100, p_101, p_102, p_103, p_105, p_106, p_108, p_109, p_110, p_111, p_112, p_113, p_115, p_116, p_117, p_118, p_119, p_12, p_120, p_13, p_14, p_15, p_17, p_18, p_19, p_20, p_22, p_23
     transitions: [ (03f290ee-425d-4d79-9002-d25c58f1d679, 'PVD'), (059e4b54-f080-42f4-8df5-88eb6ab0571c, 'PCD'), (0748425a-6ef2-423d-a482-b38f17304b64, 'Urea Nitrogen_std_1'), (0aaca4ec-63ac-44b9-b83f-1f9d92ef47cc, 'PUD
     arcs: [ (03f290ee-425d-4d79-9002-d25c58f1d679, 'PVD')->p_45, (059e4b54-f080-42f4-8df5-88eb6ab0571c, 'PCD')->p_109, (0748425a-6ef2-423d-a482-b38f17304b64, 'Urea Nitrogen_std_1')->p_91, (0aaca4ec-63ac-44b9-b83f-1f9d92
     PN Places: 106
     PN Transitions: 164
```

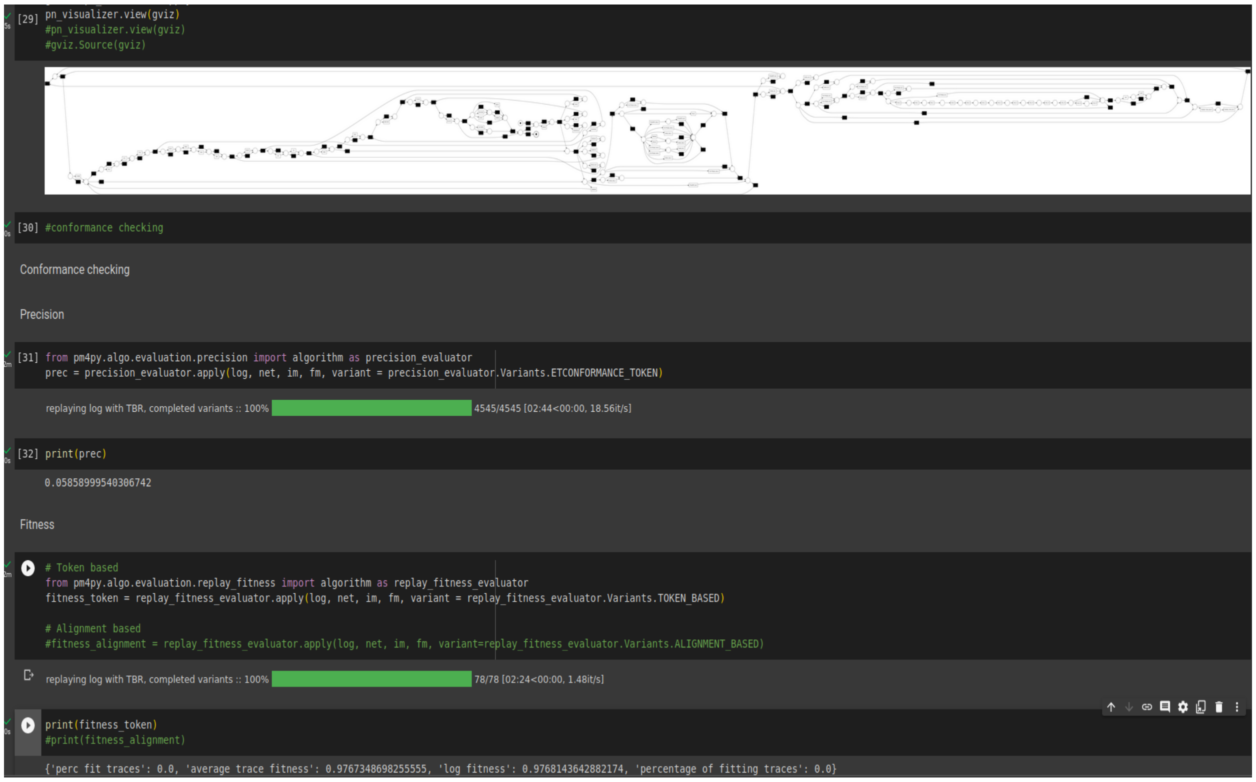
Exporting PNML

```
[25] from pm4py.objects.petri_net.exporter import exporter as pnml_exporter
     pnml_exporter.apply(net, initial_marking, "/content/drive/My Drive/Process mining project/test.pnml", final_marking=final_marking)
```

```
[26] from pm4py.objects.petri_net.importer import importer as pnml_importer
     net, im, fm = pnml_importer.apply("/content/drive/My Drive/Process mining project/test.pnml")
```

```
[27] print("PN Places:", len(net.places))
     print("\nPN Places:", list(net.places))
     print("\nPN Transitions:", len(net.transitions))
     print("\nPN Transitions:", list(net.transitions))

     PN Places: 106

     PN Places: [p_28, p_110, p_49, p_19, p_97, p_10, p_22, p_79, p_47, p_30, p_43, p_70, p_25, p_76, sink, p_52, p_119, p_72, p_115, p_59, p_83, p_57, p_73, p_56, p_101, p_54, p_86, p_88, p_116, p_9, p_61, p_62, p_105,

     PN Transitions: 164

     PN Transitions: [(skip_51, None), (skip_106, None), (42433948-52c6-4297-97c0-df6f0288ae3f, 'PSYCHO'), (skip_19, None), (ddd8c374-3f14-4c79-9fe3-53792627bea0, 'Sodium_mean_4'), (skip_65, None), (skip_66, None), (skip
```

0.6 graph



```
[29] pn_visualizer.view(gviz)
     #pn_visualizer.view(gviz)
     #gviz.Source(gviz)
```

```
[30] #conformance checking
```

Conformance checking

Precision

```
[31] from pm4py.algo.evaluation.precision import algorithm as precision_evaluator
     prec = precision_evaluator.apply(log, net, im, fm, variant = precision_evaluator.Variants.ETCONFORMANCE_TOKEN)

     replaying log with TBR, completed variants :: 100% 4545/4545 [02:44<00:00, 18.56it/s]
```

```
[32] print(prec)

     0.05858999540306742
```

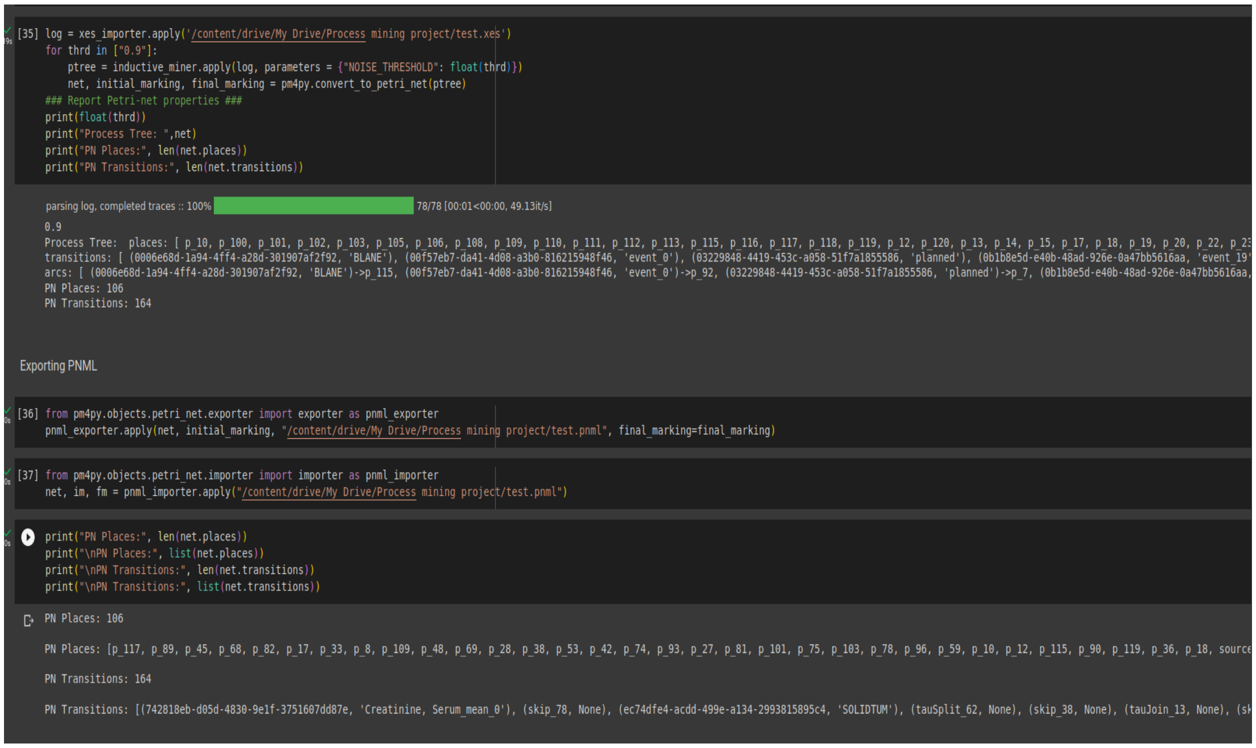
Fitness

```
# Token based
from pm4py.algo.evaluation.replay_fitness import algorithm as replay_fitness_evaluator
fitness_token = replay_fitness_evaluator.apply(log, net, im, fm, variant = replay_fitness_evaluator.Variants.TOKEN_BASED)

# Alignment based
#fitness_alignment = replay_fitness_evaluator.apply(log, net, im, fm, variant=replay_fitness_evaluator.Variants.ALIGNMENT_BASED)

replaying log with TBR, completed variants :: 100% 78/78 [02:24<00:00, 1.48it/s]
```

```
print(fitness_token)
#print(fitness_alignment)

{'perc fit traces': 0.0, 'average trace fitness': 0.9767348698255555, 'log fitness': 0.9768143642882174, 'percentage of fitting traces': 0.0}
```

Juan Uriarte
CPSC 483-01 13657

0.9 thrd



```
[35] log = xes_importer.apply('/content/drive/My Drive/Process mining project/test.xes')
     for thrd in ["0.9"]:
         ptree = inductive_miner.apply(log, parameters = {"NOISE_THRESHOLD": float(thrd)})
         net, initial_marking, final_marking = pm4py.convert_to_petri_net(ptree)
     ### Report Petri-net properties ###
     print(float(thrd))
     print("Process Tree: ",net)
     print("PN Places:", len(net.places))
     print("PN Transitions:", len(net.transitions))
```

```
parsing log, completed traces :: 100% 78/78 [00:01<00:00, 49.13it/s]
0.9
Process Tree:  places: [ p_10, p_100, p_101, p_102, p_103, p_105, p_106, p_108, p_109, p_110, p_111, p_112, p_113, p_115, p_116, p_117, p_118, p_119, p_12, p_120, p_13, p_14, p_15, p_17, p_18, p_19, p_20, p_22, p_23
transitions: [ (0006e68d-1a94-4ff4-a28d-301907af2f92, 'BLANE'), (00f57eb7-da41-4d08-a3b0-816215948f46, 'event_0'), (03229848-4419-453c-a058-51f7a1855586, 'planned'), (0b1b8e5d-e40b-48ad-926e-0a47bb5616aa, 'event_19'
arcs: [ (0006e68d-1a94-4ff4-a28d-301907af2f92, 'BLANE')->p_115, (00f57eb7-da41-4d08-a3b0-816215948f46, 'event_0')->p_92, (03229848-4419-453c-a058-51f7a1855586, 'planned')->p_7, (0b1b8e5d-e40b-48ad-926e-0a47bb5616aa,
PN Places: 106
PN Transitions: 164
```

Exporting PNML

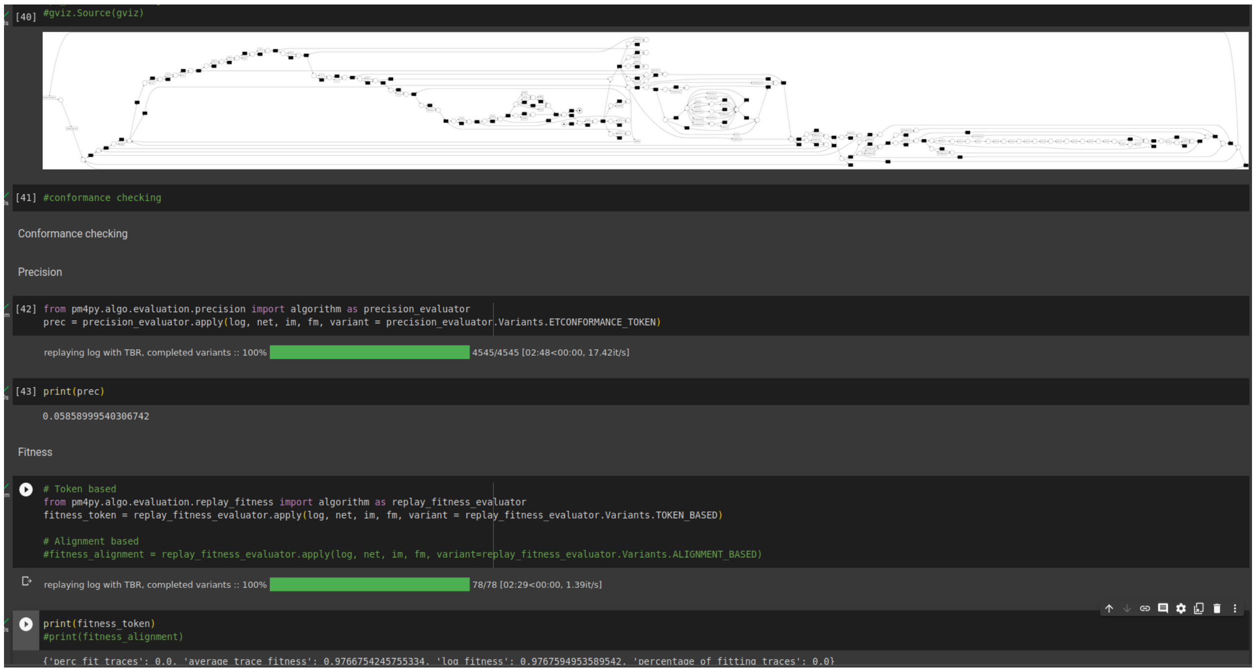
```
[36] from pm4py.objects.petri_net.exporter import exporter as pnml_exporter
     pnml_exporter.apply(net, initial_marking, "/content/drive/My Drive/Process mining project/test.pnml", final_marking=final_marking)
```

```
[37] from pm4py.objects.petri_net.importer import importer as pnml_importer
     net, im, fm = pnml_importer.apply("/content/drive/My Drive/Process mining project/test.pnml")
```

```
print("PN Places:", len(net.places))
print("\nPN Places:", list(net.places))
print("\nPN Transitions:", len(net.transitions))
print("\nPN Transitions:", list(net.transitions))
```

```
PN Places: 106

PN Places: [p_117, p_89, p_45, p_68, p_82, p_17, p_33, p_8, p_109, p_48, p_69, p_28, p_38, p_53, p_42, p_74, p_93, p_27, p_81, p_101, p_75, p_103, p_78, p_96, p_59, p_10, p_12, p_115, p_90, p_119, p_36, p_18, source

PN Transitions: 164

PN Transitions: [(742818eb-d05d-4830-9e1f-3751607dd87e, 'Creatinine, Serum_mean_0'), (skip_78, None), (ec74dfe4-acdd-499e-a134-2993815895c4, 'SOLIDTUM'), (tauSplit_62, None), (skip_38, None), (tauJoin_13, None), (sk
```

0.9 graph



```
[40] #gviz.Source(gviz)
```

```
[41] #conformance checking
```

Conformance checking

Precision

```
[42] from pm4py.algo.evaluation.precision import algorithm as precision_evaluator
     prec = precision_evaluator.apply(log, net, im, fm, variant = precision_evaluator.Variants.ETCONFORMANCE_TOKEN)
```

```
replaying log with TBR, completed variants :: 100% 4545/4545 [02:48<00:00, 17.42it/s]
```

```
[43] print(prec)
```

```
0.05858999540306742
```

Fitness

```
# Token based
from pm4py.algo.evaluation.replay_fitness import algorithm as replay_fitness_evaluator
fitness_token = replay_fitness_evaluator.apply(log, net, im, fm, variant = replay_fitness_evaluator.Variants.TOKEN_BASED)

# Alignment based
#fitness_alignment = replay_fitness_evaluator.apply(log, net, im, fm, variant=replay_fitness_evaluator.Variants.ALIGNMENT_BASED)
```

```
replaying log with TBR, completed variants :: 100% 78/78 [02:29<00:00, 1.39it/s]
```

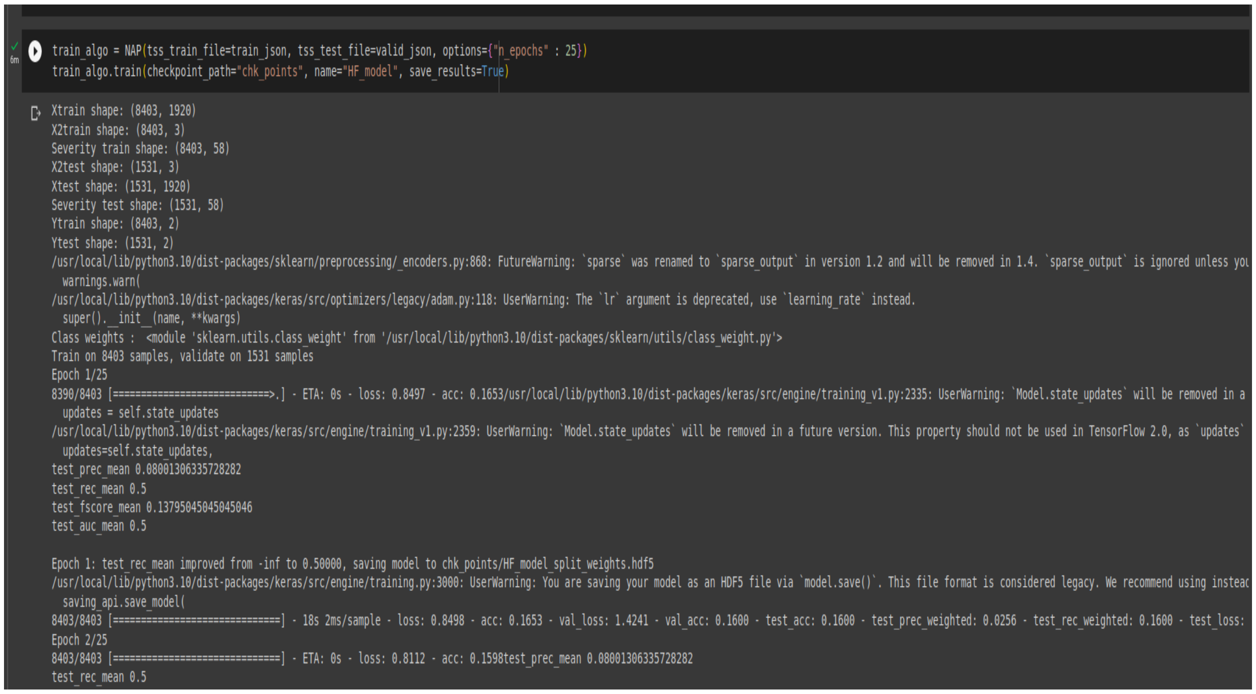
```
print(fitness_token)
#print(fitness_alignment)
```

```
{'perc_fit_traces': 0.0, 'average_trace_fitness': 0.9766754245755334, 'log_fitness': 0.9767594953589542, 'percentage_of_fitting_traces': 0.0}
```

Juan Uriarte
CPSC 483-01 13657

Part 2

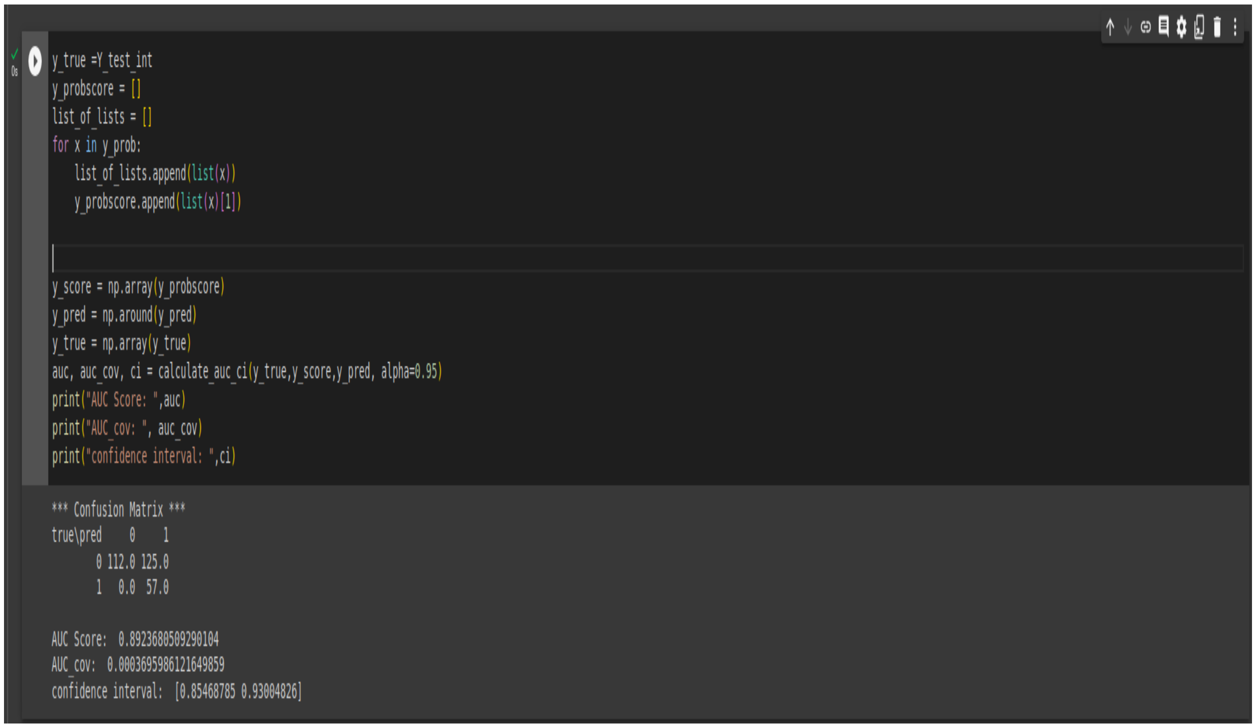
25 n_epochs input



```
train_algo = NAP(tss_train_file=train_json, tss_test_file=valid_json, options={"n_epochs" : 25})
train_algo.train(checkpoint_path="chk_points", name="HF_model", save_results=True)
```

```
Xtrain shape: (8403, 1920)
X2train shape: (8403, 3)
Severity train shape: (8403, 58)
X2test shape: (1531, 3)
Xtest shape: (1531, 1920)
Severity test shape: (1531, 58)
Ytrain shape: (8403, 2)
Ytest shape: (1531, 2)
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/keras/src/optimizers/legacy/adam.py:118: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
Class weights :  <module 'sklearn.utils.class_weight' from '/usr/local/lib/python3.10/dist-packages/sklearn/utils/class_weight.py'>
Train on 8403 samples, validate on 1531 samples
Epoch 1/25
8390/8403 [===========================>.] - ETA: 0s - loss: 0.8497 - acc: 0.1653/usr/local/lib/python3.10/dist-packages/keras/src/engine/training_v1.py:2335: UserWarning: `Model.state_updates` will be removed in a
  updates = self.state_updates
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training_v1.py:2359: UserWarning: `Model.state_updates` will be removed in a future version. This property should not be used in TensorFlow 2.0, as `updates`
  updates=self.state_updates,
test_prec_mean 0.08001306335728282
test_rec_mean 0.5
test_fscore_mean 0.13795045045045046
test_auc_mean 0.5

Epoch 1: test_rec_mean improved from -inf to 0.50000, saving model to chk_points/HF_model_split_weights.hdf5
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using insteac
  saving_api.save_model(
8403/8403 [==============================] - 18s 2ms/sample - loss: 0.8498 - acc: 0.1653 - val_loss: 1.4241 - val_acc: 0.1600 - test_acc: 0.1600 - test_prec_weighted: 0.0256 - test_rec_weighted: 0.1600 - test_loss:
Epoch 2/25
8403/8403 [==============================] - ETA: 0s - loss: 0.8112 - acc: 0.1598test_prec_mean 0.08001306335728282
test_rec_mean 0.5
```

25 n_epochs results

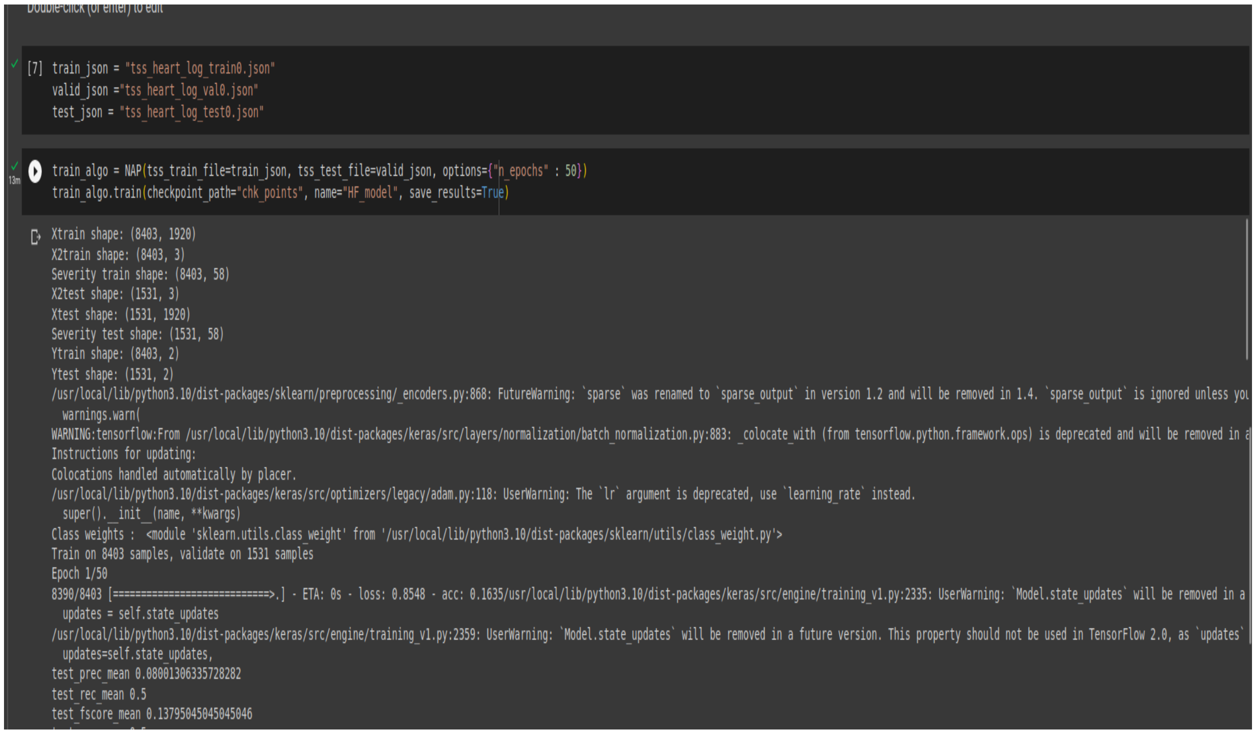


```
y_true =Y_test_int
y_probscore = []
list_of_lists = []
for x in y_prob:
    list_of_lists.append(list(x))
    y_probscore.append(list(x)[1])


y_score = np.array(y_probscore)
y_pred = np.around(y_pred)
y_true = np.array(y_true)
auc, auc_cov, ci = calculate_auc_ci(y_true,y_score,y_pred, alpha=0.95)
print("AUC Score: ",auc)
print("AUC cov: ", auc_cov)
print("confidence interval: ",ci)
```

```
*** Confusion Matrix ***
true\pred    0    1
       0 112.0 125.0
       1   0.0  57.0

AUC Score:  0.8923680509290104
AUC cov:  0.0003695986121649859
confidence interval:  [0.85468785 0.93004826]
```

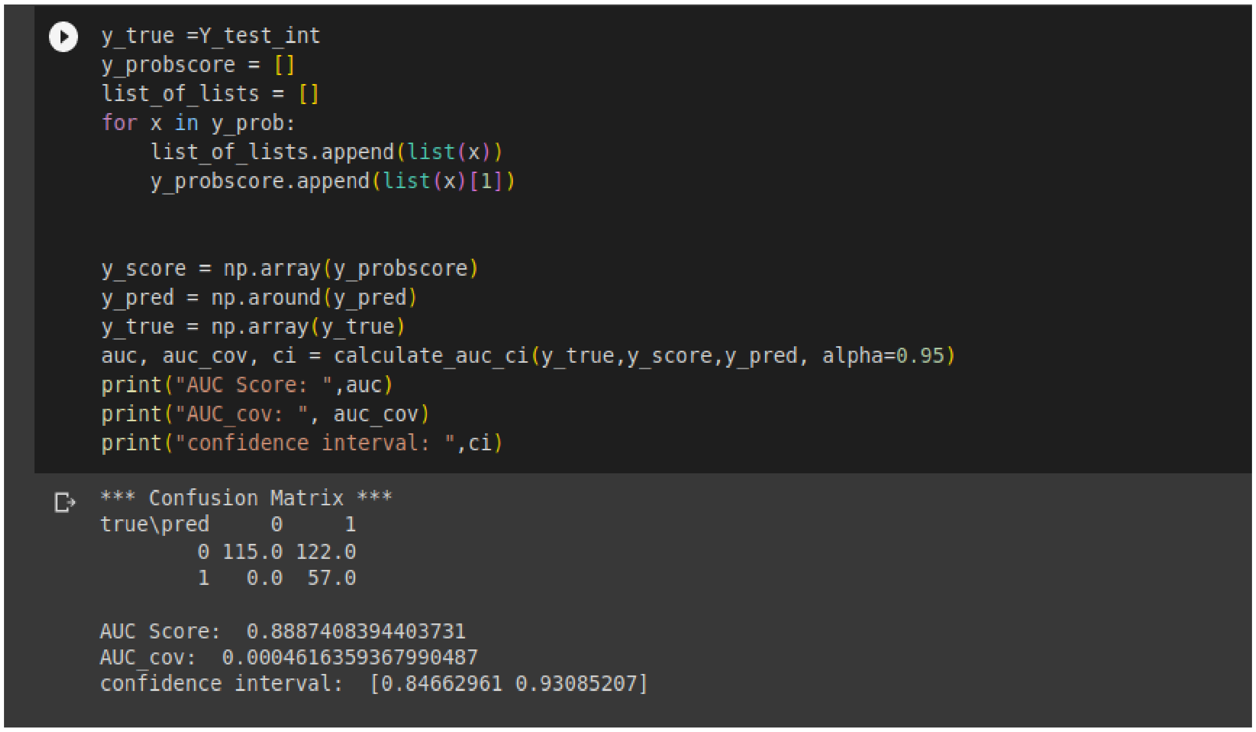
Juan Uriarte
CPSC 483-01 13657

50 n_epochs input



```
Double-click (or enter) to edit

[7] train_json = "tss_heart_log_train0.json"
    valid_json ="tss_heart_log_val0.json"
    test_json = "tss_heart_log_test0.json"

train_algo = NAP(tss_train_file=train_json, tss_test_file=valid_json, options={"n_epochs" : 50})
train_algo.train(checkpoint_path="chk_points", name="HF_model", save_results=True)

Xtrain shape: (8403, 1920)
X2train shape: (8403, 3)
Severity train shape: (8403, 58)
X2test shape: (1531, 3)
Xtest shape: (1531, 1920)
Severity test shape: (1531, 58)
Ytrain shape: (8403, 2)
Ytest shape: (1531, 2)
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you
  warnings.warn(
WARNING:tensorflow:From /usr/local/lib/python3.10/dist-packages/keras/src/layers/normalization/batch_normalization.py:883: _colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a
Instructions for updating:
Colocations handled automatically by placer.
/usr/local/lib/python3.10/dist-packages/keras/src/optimizers/legacy/adam.py:118: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
Class weights :  <module 'sklearn.utils.class_weight' from '/usr/local/lib/python3.10/dist-packages/sklearn/utils/class_weight.py'>
Train on 8403 samples, validate on 1531 samples
Epoch 1/50
8390/8403 [============================>.] - ETA: 0s - loss: 0.8548 - acc: 0.1635/usr/local/lib/python3.10/dist-packages/keras/src/engine/training_v1.py:2335: UserWarning: `Model.state_updates` will be removed in a
  updates = self.state_updates
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training_v1.py:2359: UserWarning: `Model.state_updates` will be removed in a future version. This property should not be used in TensorFlow 2.0, as `updates`
  updates=self.state_updates,
test_prec_mean 0.08001306335728282
test_rec_mean 0.5
test_fscore_mean 0.13795045045045046
```

50 n_epochs result

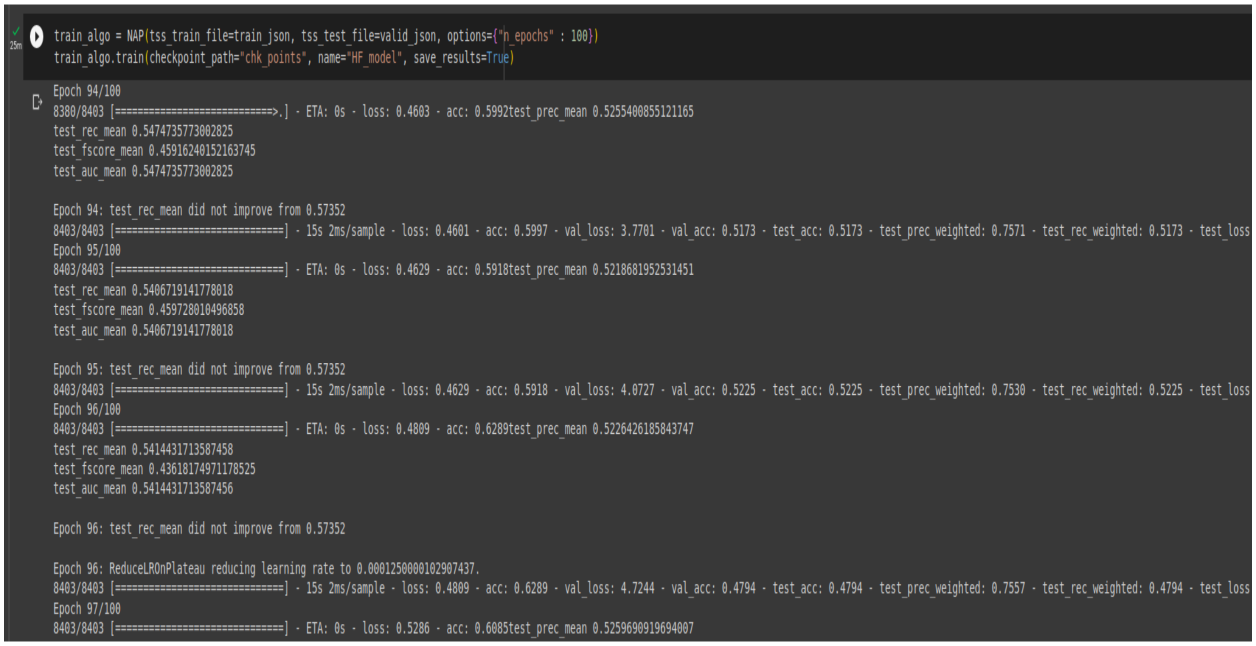


```
y_true =Y_test_int
y_probscore = []
list_of_lists = []
for x in y_prob:
    list_of_lists.append(list(x))
    y_probscore.append(list(x)[1])


y_score = np.array(y_probscore)
y_pred = np.around(y_pred)
y_true = np.array(y_true)
auc, auc_cov, ci = calculate_auc_ci(y_true,y_score,y_pred, alpha=0.95)
print("AUC Score: ",auc)
print("AUC_cov: ", auc_cov)
print("confidence interval: ",ci)
```

```
*** Confusion Matrix ***
true\pred     0     1
        0 115.0 122.0
        1   0.0  57.0

AUC Score:  0.8887408394403731
AUC_cov:  0.0004616359367990487
confidence interval:  [0.84662961 0.93085207]
```

Juan Uriarte
CPSC 483-01 13657

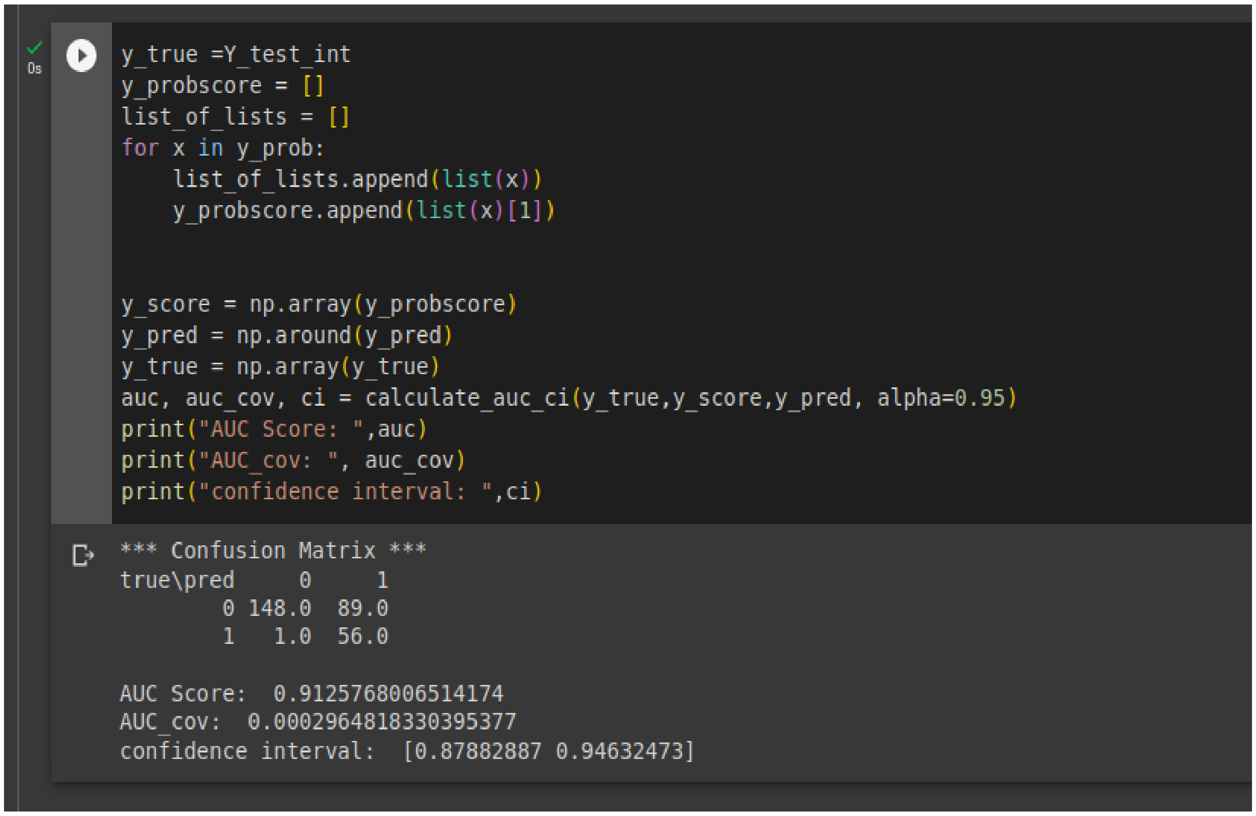
100 n_epochs input



```
train_algo = NAP(tss_train_file=train_json, tss_test_file=valid_json, options={"n_epochs" : 100})
train_algo.train(checkpoint_path="chk_points", name="HF_model", save_results=True)
```

```
Epoch 94/100
8380/8403 [===========================>.] - ETA: 0s - loss: 0.4603 - acc: 0.5992test_prec_mean 0.5255400855121165
test_rec_mean 0.5474735773002825
test_fscore_mean 0.45916240152163745
test_auc_mean 0.5474735773002825

Epoch 94: test_rec_mean did not improve from 0.57352
8403/8403 [==============================] - 15s 2ms/sample - loss: 0.4601 - acc: 0.5997 - val_loss: 3.7701 - val_acc: 0.5173 - test_acc: 0.5173 - test_prec_weighted: 0.7571 - test_rec_weighted: 0.5173 - test_loss
Epoch 95/100
8403/8403 [==============================] - ETA: 0s - loss: 0.4629 - acc: 0.5918test_prec_mean 0.5218681952531451
test_rec_mean 0.5406719141778018
test_fscore_mean 0.459728010496858
test_auc_mean 0.5406719141778018

Epoch 95: test_rec_mean did not improve from 0.57352
8403/8403 [==============================] - 15s 2ms/sample - loss: 0.4629 - acc: 0.5918 - val_loss: 4.0727 - val_acc: 0.5225 - test_acc: 0.5225 - test_prec_weighted: 0.7530 - test_rec_weighted: 0.5225 - test_loss
Epoch 96/100
8403/8403 [==============================] - ETA: 0s - loss: 0.4809 - acc: 0.6289test_prec_mean 0.5226426185843747
test_rec_mean 0.5414431713587458
test_fscore_mean 0.43618174971178525
test_auc_mean 0.5414431713587456

Epoch 96: test_rec_mean did not improve from 0.57352

Epoch 96: ReduceLROnPlateau reducing learning rate to 0.0001250000102907437.
8403/8403 [==============================] - 15s 2ms/sample - loss: 0.4809 - acc: 0.6289 - val_loss: 4.7244 - val_acc: 0.4794 - test_acc: 0.4794 - test_prec_weighted: 0.7557 - test_rec_weighted: 0.4794 - test_loss
Epoch 97/100
8403/8403 [==============================] - ETA: 0s - loss: 0.5286 - acc: 0.6085test_prec_mean 0.5259690919694007
```

100 n_epochs result



```
y_true =Y_test_int
y_probscore = []
list_of_lists = []
for x in y_prob:
    list_of_lists.append(list(x))
    y_probscore.append(list(x)[1])


y_score = np.array(y_probscore)
y_pred = np.around(y_pred)
y_true = np.array(y_true)
auc, auc_cov, ci = calculate_auc_ci(y_true,y_score,y_pred, alpha=0.95)
print("AUC Score: ",auc)
print("AUC_cov: ", auc_cov)
print("confidence interval: ",ci)
```

```
*** Confusion Matrix ***
true\pred     0     1
        0 148.0  89.0
        1   1.0  56.0

AUC Score:  0.9125768006514174
AUC_cov:  0.0002964818330395377
confidence interval:  [0.87882887 0.94632473]
```