evaluate_scores

July 23, 2017

```
In [1]: import pandas as pd
        import numpy as np
        import warnings
        from scipy import stats
        from IPython.display import display, HTML
        from sklearn import metrics as me

        warnings.filterwarnings('ignore')
        pd.set_option("display.max_rows",20)
        pd.set_option('precision', 4)
        import pylab as pl
        pl.figure(figsize=(10, 10))

        %matplotlib inline

In [2]: import numpy as np
        import matplotlib.pyplot as plt
        import itertools

        def plot_confusion_matrix(cm, classes,
                                  normalize=False,
                                  append = "",
                                  cmap=plt.cm.Blues):
            """
            This function prints and plots the confusion matrix.
            Normalization can be applied by setting `normalize=True`.
            """
            title='Confusion matrix {}'.format(append)
            np.set_printoptions(precision=4)

            plt.imshow(cm, interpolation='nearest', cmap=cmap)
            plt.title(title)
            plt.colorbar()
            tick_marks = np.arange(len(classes))
            plt.xticks(tick_marks, classes, rotation=45)
            plt.yticks(tick_marks, classes)

            if normalize:
                cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
                #print("Normalized confusion matrix")
            else:
                #print('Confusion matrix, without normalization')
                pass

            #print(cm)

            label = [["\n True Negative", "\n False Positive "],
                     ["\n False Negative ", "\n True Positive"]
                    ]

            thresh = cm.max() / 2.
            for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

                plt.text(j, i, "{} {}".format(cm[i, j].round(4), label[i][j]),
                         horizontalalignment="center",
                         color="white" if cm[i, j] > thresh else "black")

            plt.tight_layout()
            plt.ylabel('True label')
            plt.xlabel('Predicted label')

        def plot(actual_value, pred_value, scenario):
            from sklearn.metrics import confusion_matrix
            append = '{}; \n Total Normal traffic:{}, \n Total Attack Traffic: {}'.format(scena
                                                                                    actual_valu
                                                                                   actual_value

            cm_2labels = confusion_matrix(y_pred = pred_value, y_true = actual_value)
            plt.figure(figsize=[6,6])
            plot_confusion_matrix(cm_2labels, ['Normal', 'Attack'], normalize = False, append =

In [9]: def evaluate_lstm(model, past_scores, predictions):
            return evaluate(model, past_scores, predictions, 'LSTM')

        lstm_result = []

        def evaluate(model, past_scores, predictions, model_type='AE'):
            all_scenarios = pd.DataFrame(columns=['Model','Scenarios', 'Number of Features', '


            def get_best_df(past_scores):
                psg = past_scores.sort_values(by='quality_score', ascending=False).groupby(by=
                df = psg.first().sort_values(by='quality_score', ascending=False)
                return df
            def get_median_df(past_scores):
                psg = past_scores.sort_values(by='quality_score', ascending=False).groupby(by=
                df = psg.nth(int(psg.size()[0]/2)).sort_values(by='quality_score', ascending=Fa
                return df
            def get_worst_df(past_scores):
                psg = past_scores.sort_values(by='quality_score', ascending=False).groupby(by=
                df = psg.last().sort_values(by='quality_score', ascending=False)
                return df

            def get_result(past_scores, which='best'):
                if which == 'best':
                    df = get_best_df(past_scores)
                elif which == 'median':
                    df = get_median_df(past_scores)
                elif which == 'worst':
                    df = get_worst_df(past_scores)

                #epoch_nof_hidden
                key = int(df.iloc[0]['key'])
                nof = df.iloc[0].name[0]
                hidden = df.iloc[0].name[1]

                return "{}_{}_{}".format(key, nof, hidden), nof, df

            def view_data(name, past_scores, which):
                _, _, df = get_result(past_scores, which)
                #display(name)
                #display(df)

                group_by = 'no_of_features'
                if(model_type == 'LSTM'):
                    group_by = 'hidden_layers'
                df1 = df.reset_index().sort_values(by='quality_score', ascending=False).groupby

                df1 = df1.first().loc[:,['quality_score', 'time_taken']]
                df1 = df1.rename(index={1:"One", 4:"10%", 8:"20%", 16:"40%", 42:"All"})
                df1.plot(secondary_y = 'time_taken', title=name)

            #display("Individual Results for each Scenario:-")
            view_data("   Results for {}: Best case".format(model),past_scores, 'best')
            view_data("   Results for {}: General case".format(model),past_scores,'median')
            view_data("   Results for {}: Worst case".format(model),past_scores,'worst')

            def get_score(y_true, y_pred):
                f1 = me.matthews_corrcoef(y_true, y_pred)
                pre = me.precision_score(y_true, y_pred)
                rec = me.recall_score(y_true, y_pred)
                acc = me.accuracy_score(y_true, y_pred)
                return {"Quality Score":f1, "Precision":pre, "Recall":rec, "Accuracy":acc}
            display("Combined Results from all Scenarios for {}".format(model))


            def accumulate_scenarios(predictions, past_scores, which='best'):
                key, nof, df = get_result(past_scores, which)
                y_true = predictions[key]["Actual"]
                y_pred = predictions[key]["Prediction"]
                y_true = y_true.dropna()
                y_pred = y_pred.dropna()

                scores = get_score(y_true, y_pred)
                scores.update({"Scenarios":scenario,"Number of Features":nof,"Model":model})
                plot(y_true, y_pred, scenario)
                return pd.DataFrame(scores, index=[1])

            scenario = "Best Result"
            all_scenarios = all_scenarios.append(accumulate_scenarios(predictions, past_scores
            scenario = "General Result"
            all_scenarios = all_scenarios.append(accumulate_scenarios(predictions, past_scores
            scenario = "Worst Result"
            all_scenarios = all_scenarios.append(accumulate_scenarios(predictions, past_scores


            display(all_scenarios.set_index(['Model','Scenarios','Number of Features']))

            return all_scenarios
```

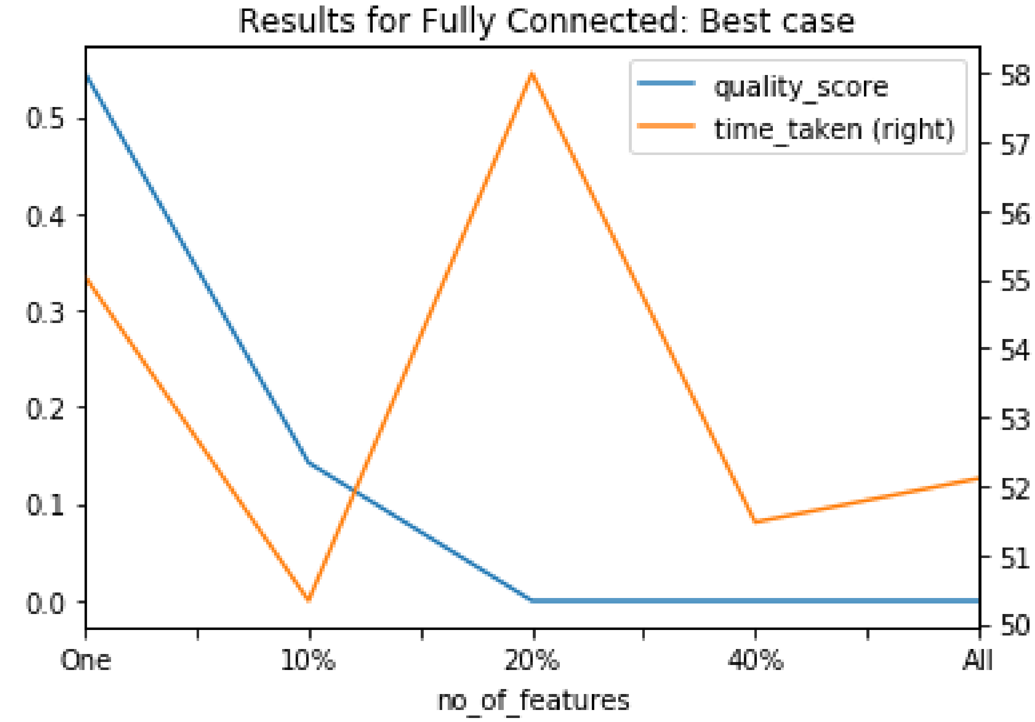
```
In [7]: past_scores = pd.read_pickle("dataset/scores/tf_dense_only_nsl_kdd_scores_all.pkl")
        predictions = pd.read_pickle("dataset/tf_dense_only_nsl_kdd_predictions.pkl")
```

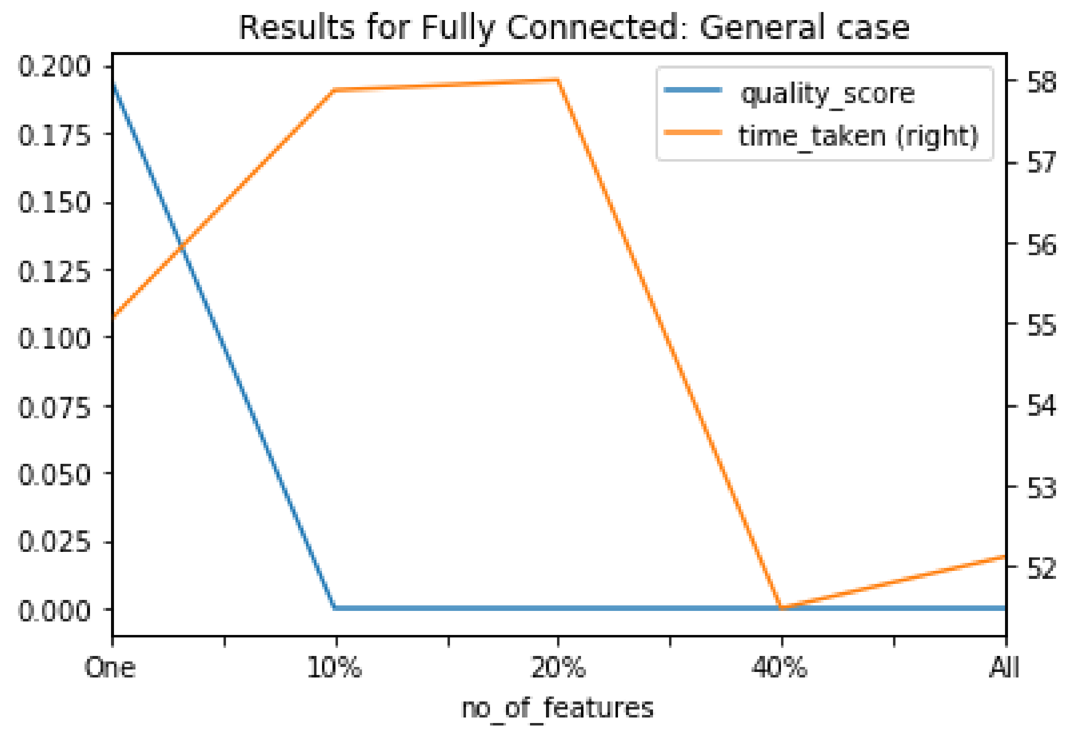
```
In [10]: all_scenarios_fcn = evaluate("Fully Connected", past_scores, predictions)
```

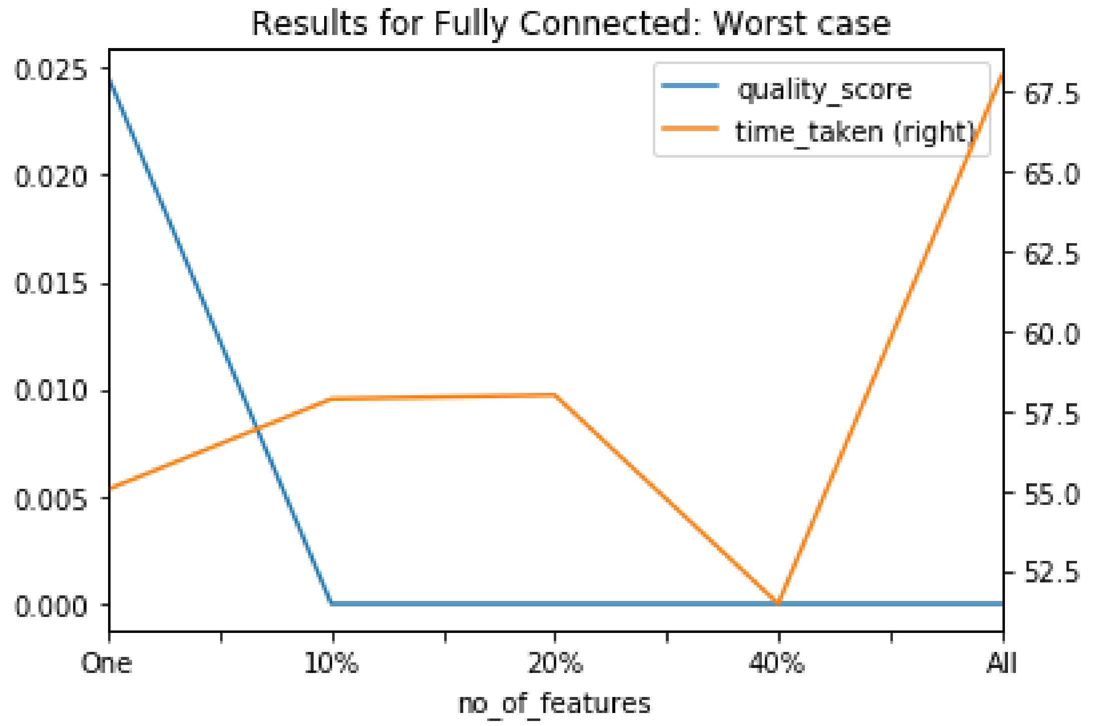
```
'Combined Results from all Scenarios for Fully Connected'
```

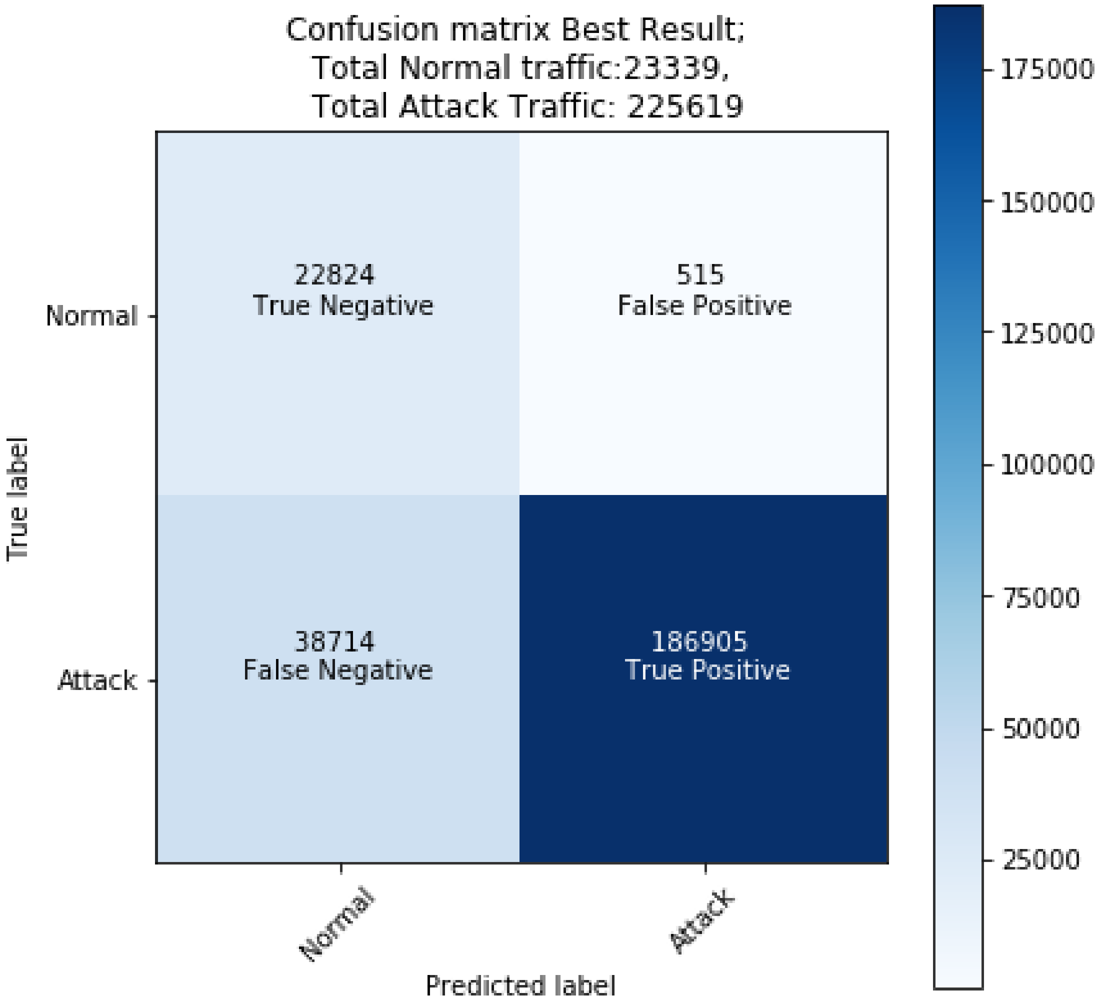
```
                                                   Accuracy  Precision  \
Model           Scenarios      Number of Features
Fully Connected Best Result    1                     0.8424     0.9973
                General Result 1                     0.7098     0.9899
                Worst Result   1                     0.6815     0.9922

                                                   Quality Score  Recall
Model           Scenarios      Number of Features
```

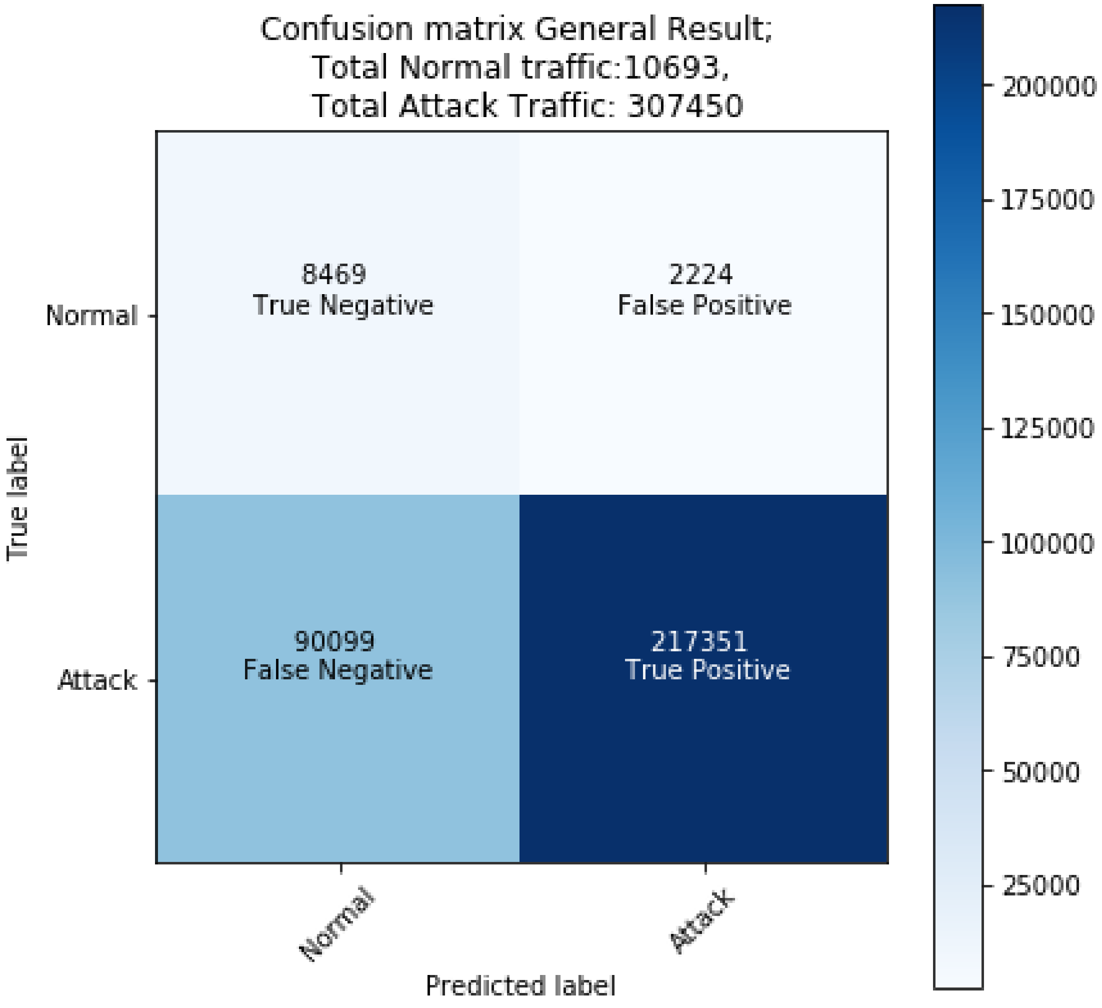
```
Fully Connected Best Result    1                          0.5448  0.8284
                General Result 1                          0.1945  0.7069
                Worst Result   1                          0.0246  0.6838
```

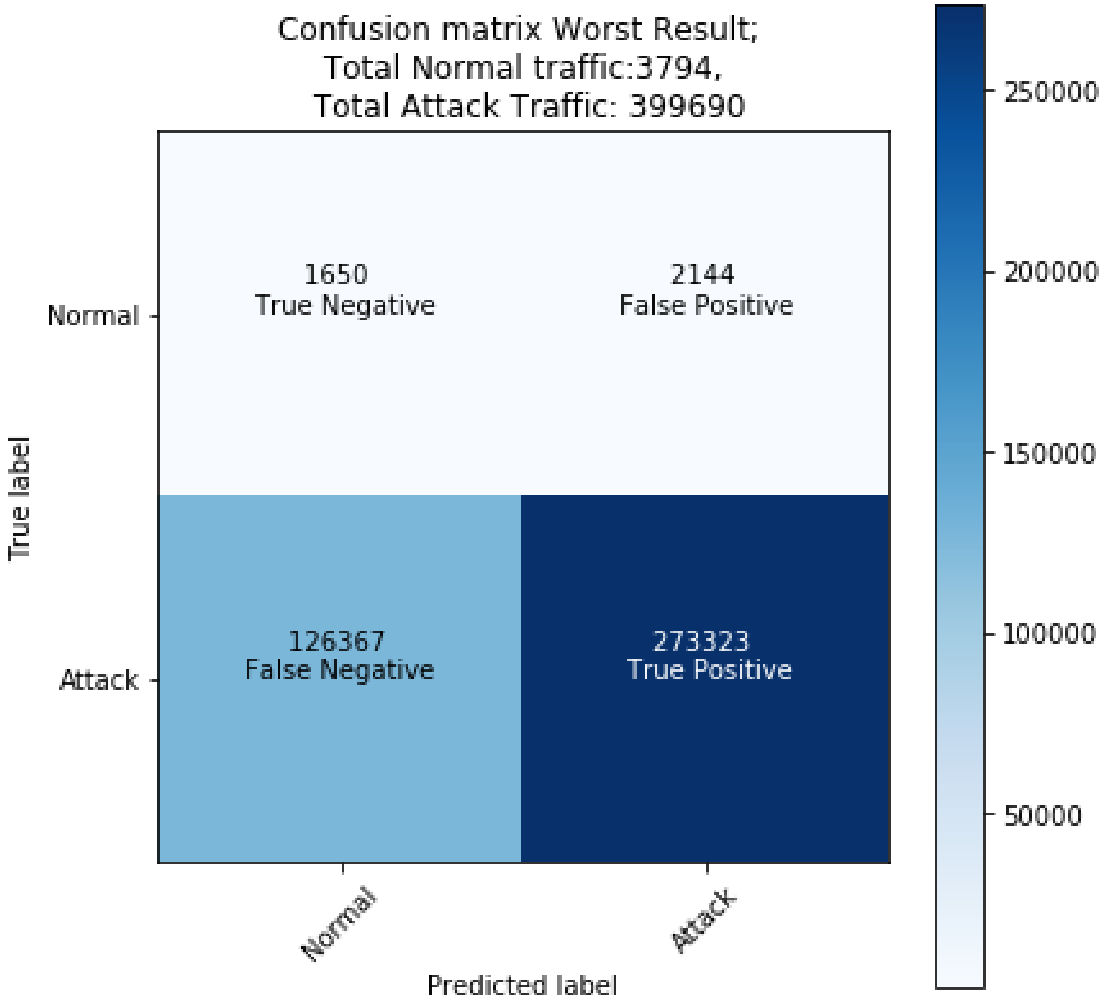



Results for Fully Connected: General case
quality_score
time_taken (right)
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
58
57
56
55
54
53
52
One
10%
20%
40%
All
no_of_features


Results for Fully Connected: Worst case
quality_score
time_taken (right)
0.025
0.020
0.015
0.010
0.005
0.000
67.5
65.0
62.5
60.0
57.5
55.0
52.5
One
10%
20%
40%
All
no_of_features


Confusion matrix Best Result;
Total Normal traffic:23339,
Total Attack Traffic: 225619
22824
True Negative
515
False Positive
38714
False Negative
186905
True Positive
Normal
Attack
True label
Normal
Attack
Predicted label
175000
150000
125000
100000
75000
50000
25000


Confusion matrix General Result;
Total Normal traffic:10693,
Total Attack Traffic: 307450
8469
True Negative
2224
False Positive
90099
False Negative
217351
True Positive
Normal
Attack
True label
Normal
Attack
Predicted label
200000
175000
150000
125000
100000
75000
50000
25000



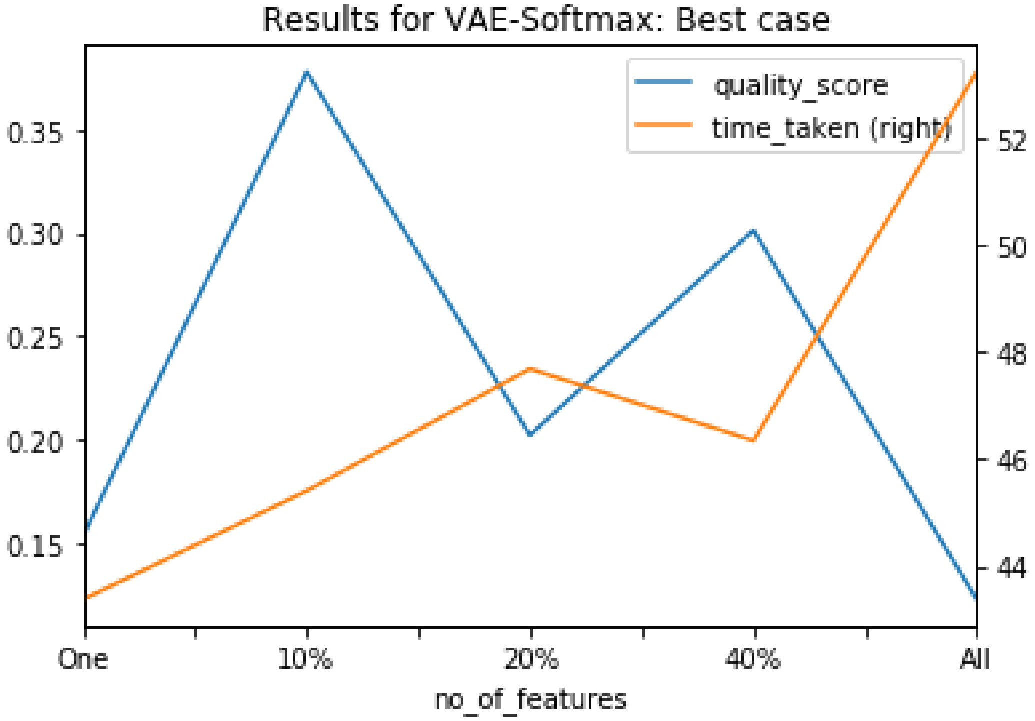
psg = past_scores.sort_values(by='f1_score', ascending=False).groupby(by=['no_of_features', 'hidden_layers']) df = psg.first().sort_values(by='f1_score', ascending=False).head(5) df1 = df.reset_index().sort_values(by='f1_score', ascending=False).groupby(by=['no_of_features']) df1 = df1.first().loc[:,['f1_score', 'time_taken']] df1 = df1.rename(index={1:"One", 4:"10%", 8:"20%", 16:"40%", 42:"All"}) df1.plot(secondary_y = 'time_taken', table=True)

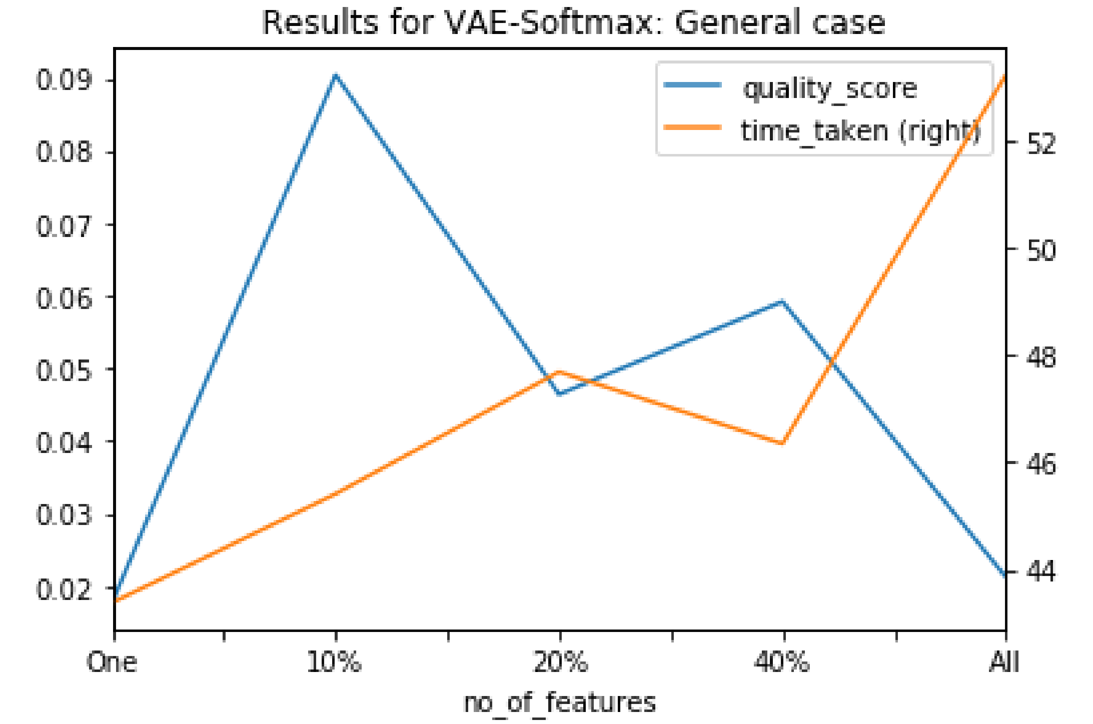
```
In [11]: past_scores = pd.read_pickle("dataset/scores/tf_vae_dense_trained_together_nsl_kdd_sco
         predictions = pd.read_pickle("dataset/tf_vae_dense_trained_together_nsl_kdd_predictio
```

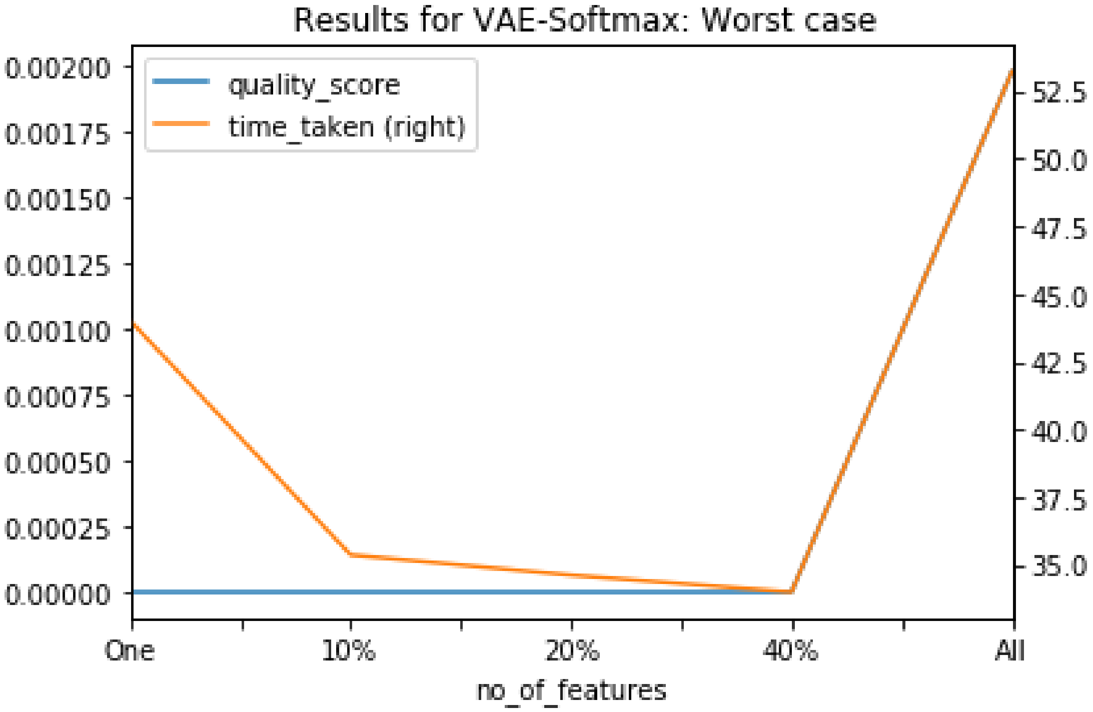
```
In [12]: all_scenarios_vae_sm = evaluate("VAE-Softmax", past_scores, predictions)
```

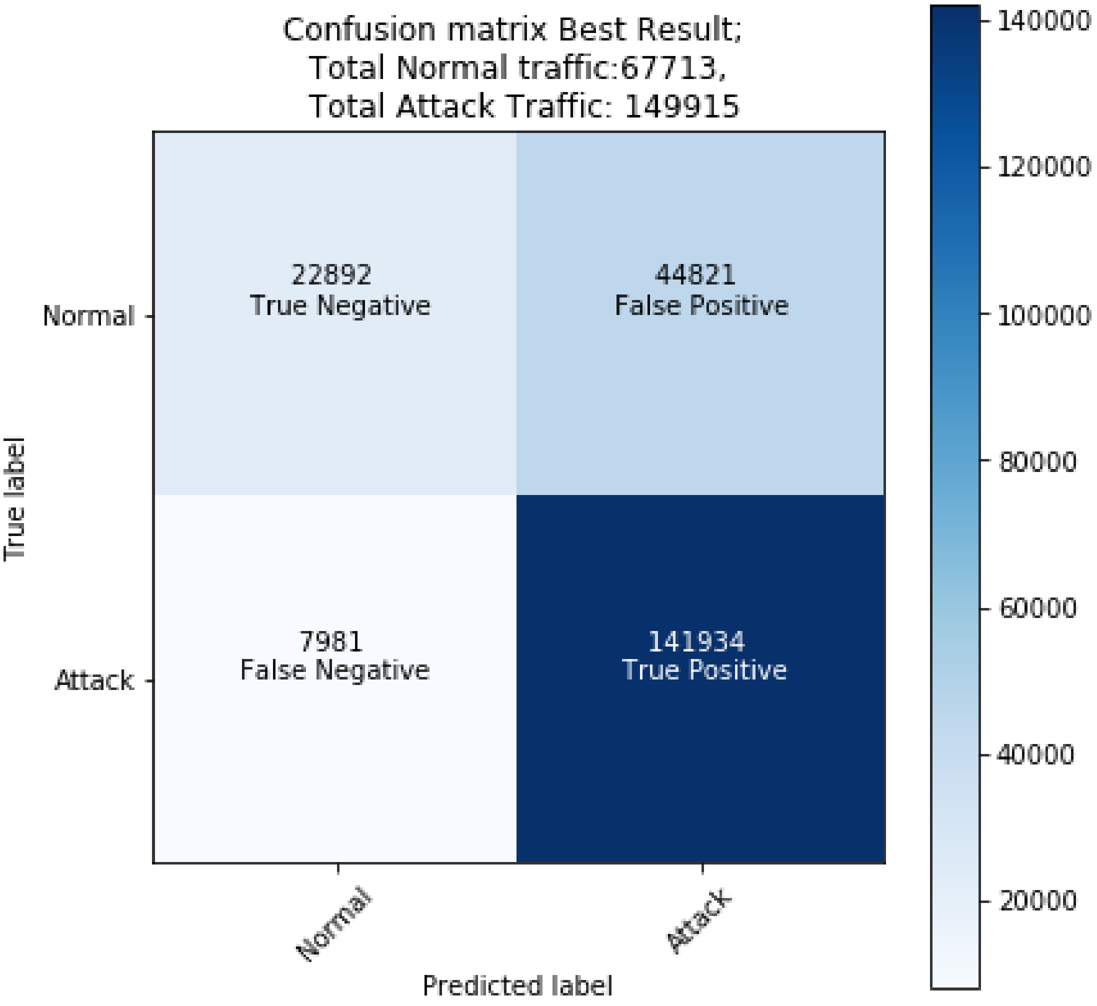
```
'Combined Results from all Scenarios for VAE-Softmax'


                                          Accuracy  Precision  \
Model       Scenarios     Number of Features
VAE-Softmax Best Result   4                   0.7574     0.7600
```

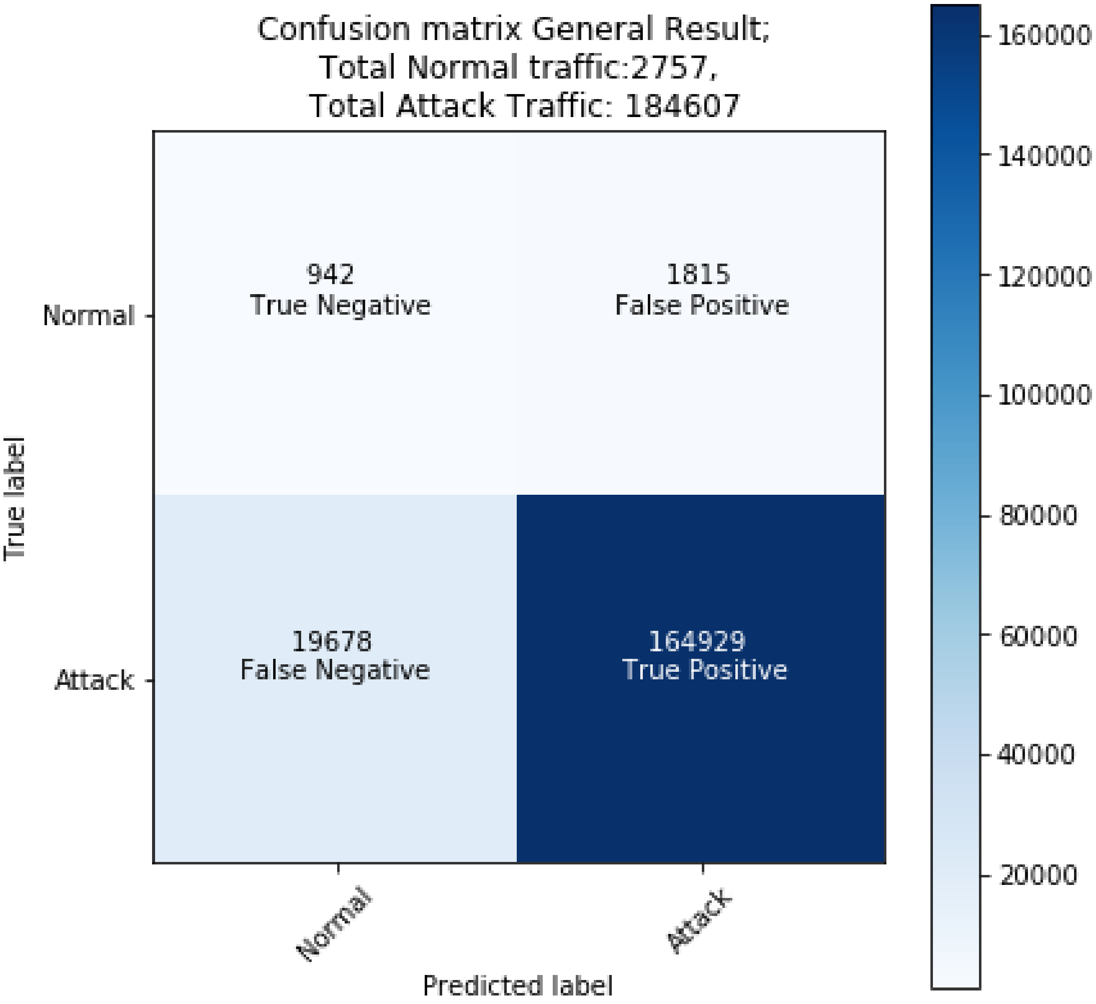
```
            General Result 4                     0.8853     0.9891
            Worst Result   42                    0.9723     0.9867

                                               Quality Score  Recall
Model       Scenarios      Number of Features
VAE-Softmax Best Result    4                          0.3779  0.9468
            General Result 4                          0.0904  0.8934
            Worst Result   42                         0.0020  0.9851
```

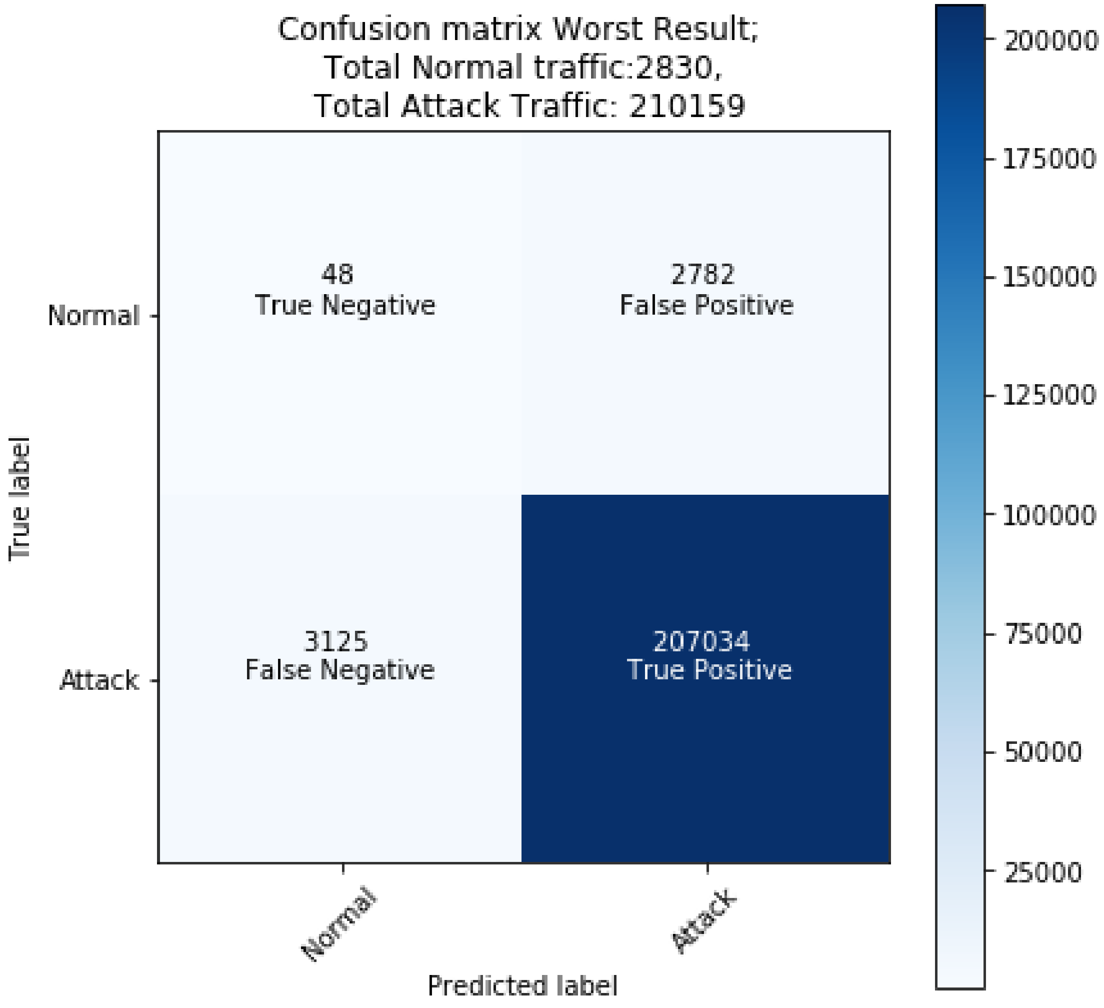



Results for VAE-Softmax: General case
quality_score
time_taken (right)
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
52
50
48
46
44
One
10%
20%
40%
All
no_of_features


Results for VAE-Softmax: Worst case
quality_score
time_taken (right)
0.00200
0.00175
0.00150
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
52.5
50.0
47.5
45.0
42.5
40.0
37.5
35.0
One
10%
20%
40%
All
no_of_features


Confusion matrix Best Result;
Total Normal traffic:67713,
Total Attack Traffic: 149915
22892
True Negative
44821
False Positive
7981
False Negative
141934
True Positive
Normal
Attack
True label
Normal
Attack
Predicted label
140000
120000
100000
80000
60000
40000
20000

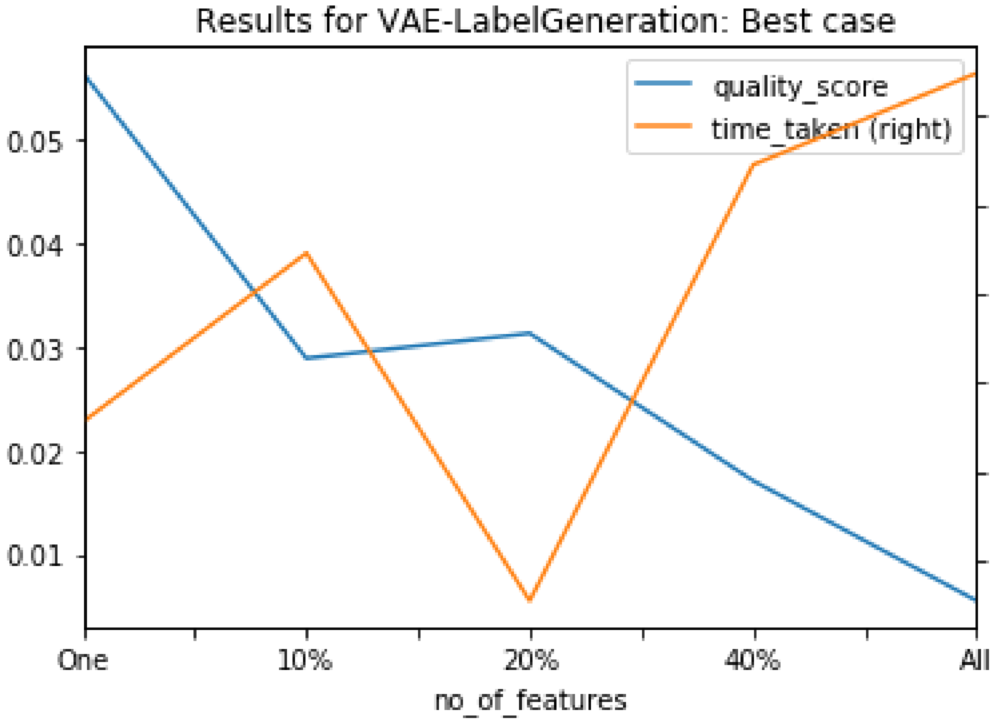

Confusion matrix General Result;
Total Normal traffic:2757,
Total Attack Traffic: 184607
942
True Negative
1815
False Positive
19678
False Negative
164929
True Positive
Normal
Attack
True label
Normal
Attack
Predicted label
160000
140000
120000
100000
80000
60000
40000
20000

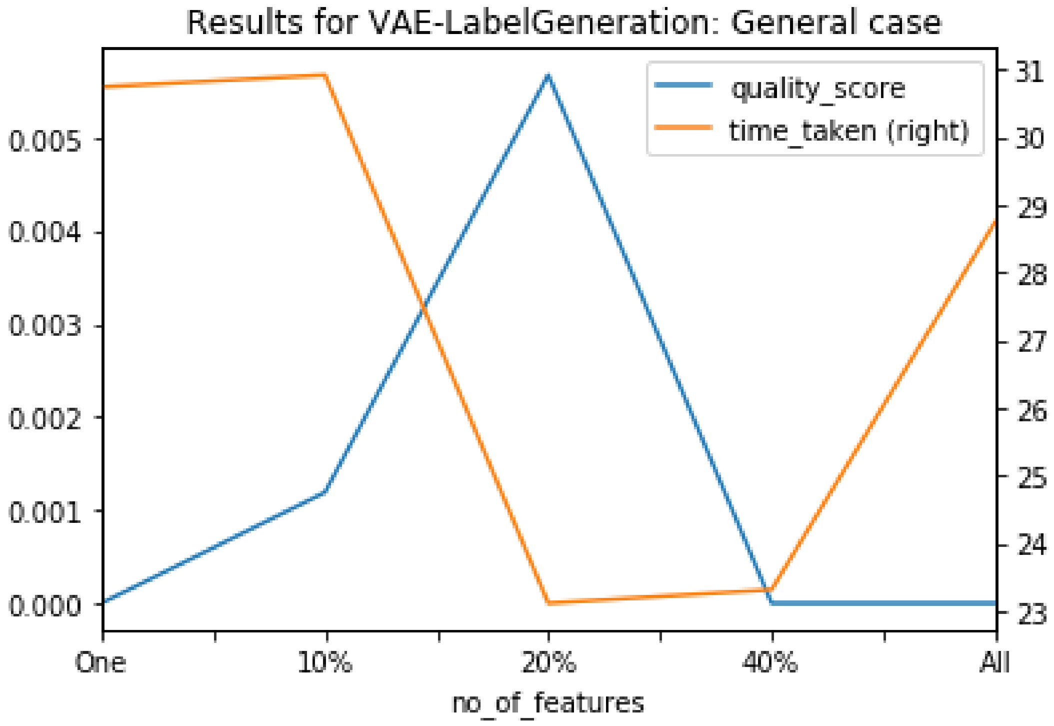


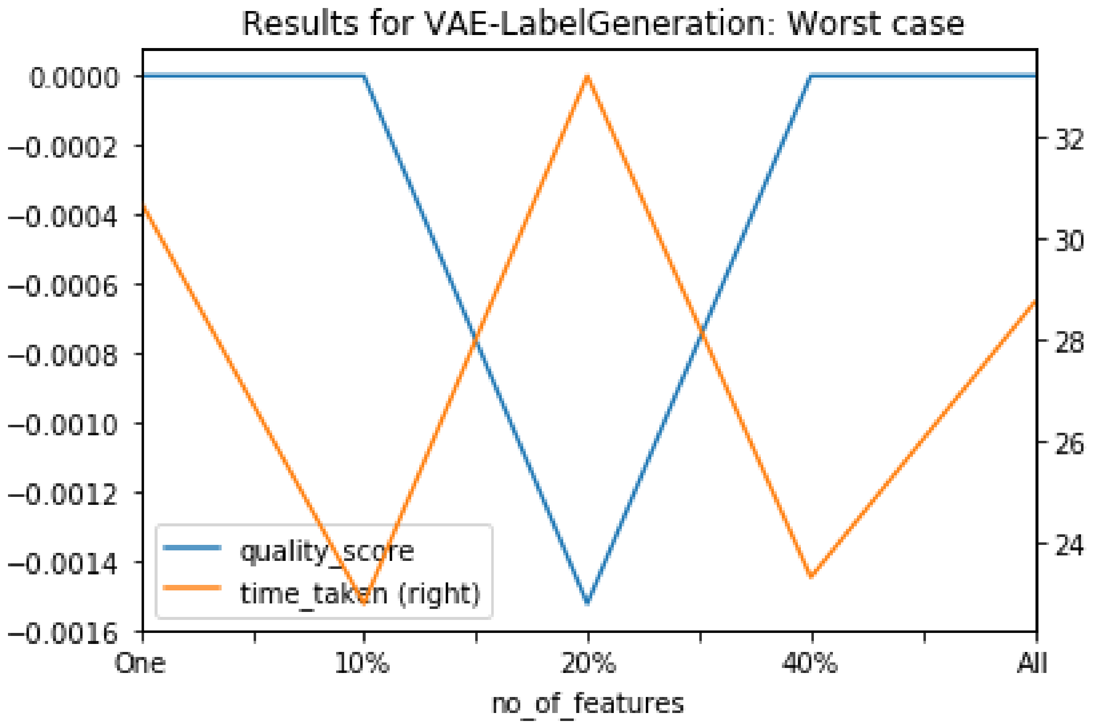
```
In [13]: past_scores = pd.read_pickle("dataset/scores/tf_vae_only_vae_loss_nsl_kdd_scores_all.
         predictions = pd.read_pickle("dataset/tf_vae_only_vae_loss_nsl_kdd_predictions.pkl")

In [14]: all_scenarios_vae = evaluate("VAE-LabelGeneration", past_scores, predictions)
```

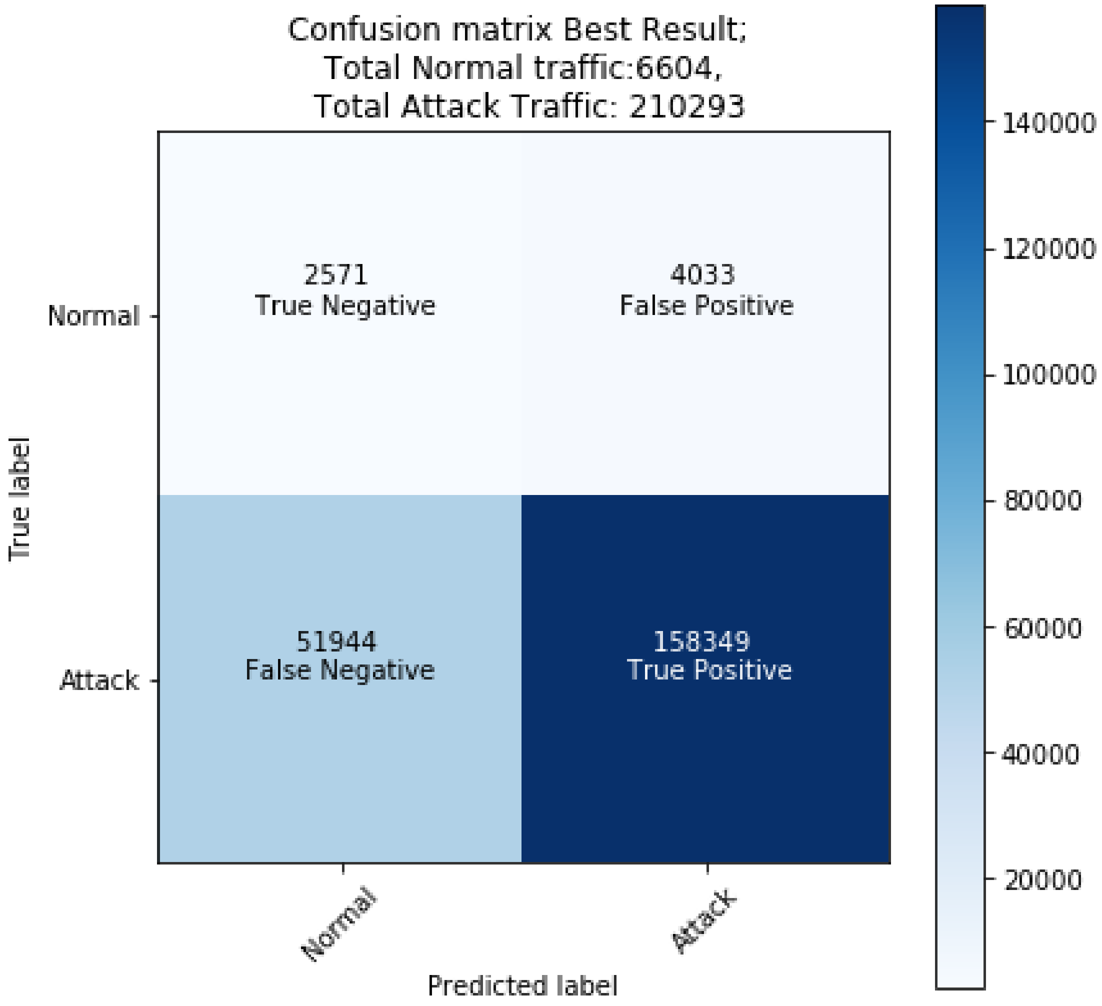
```
'Combined Results from all Scenarios for VAE-LabelGeneration'


                                                       Accuracy  Precision   \
Model               Scenarios      Number of Features
VAE-LabelGeneration Best Result    1                     0.7419     0.9752
                    General Result 8                     0.5578     0.9835
                    Worst Result   1                     0.9834     0.9834

                                                       Quality Score  Recall
Model               Scenarios      Number of Features
```

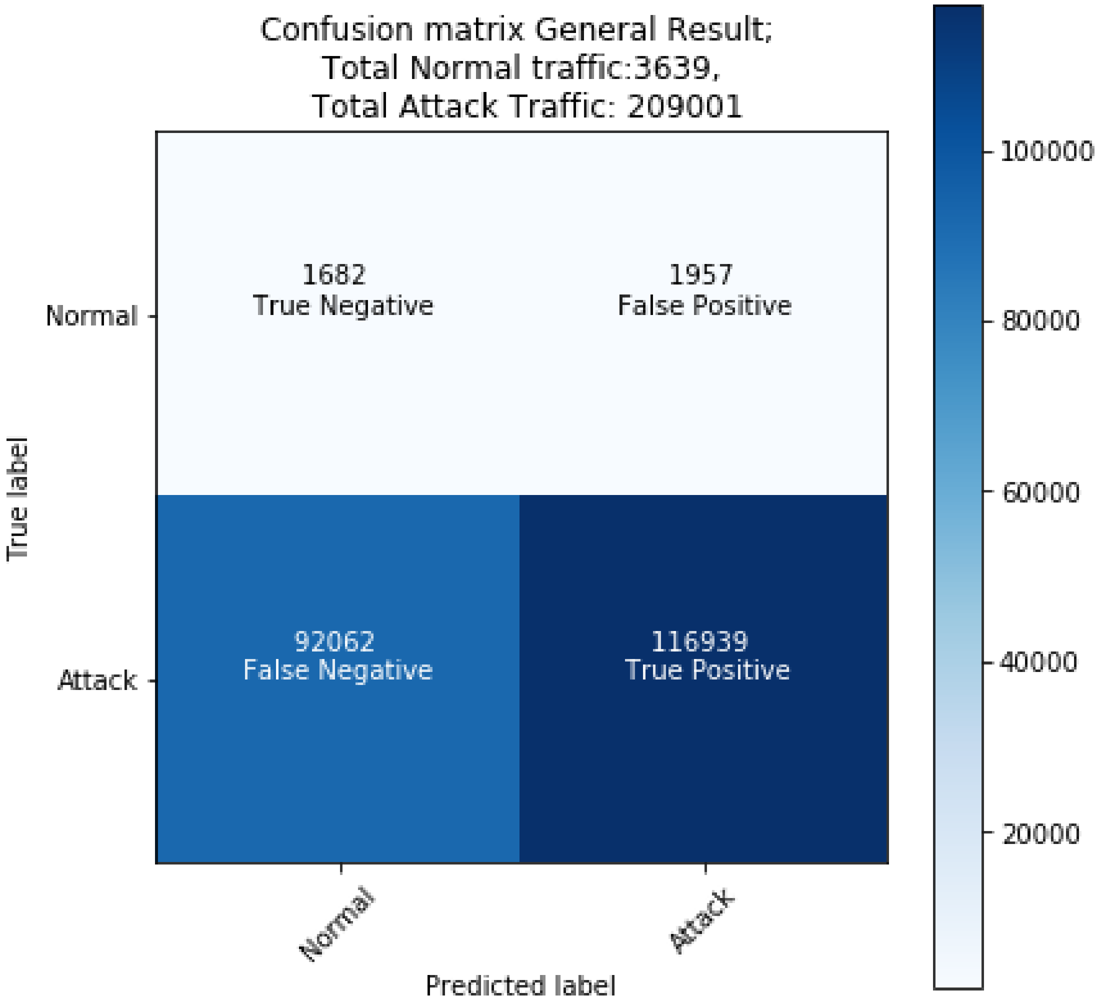
```
VAE-LabelGeneration Best Result    1                           0.0564  0.7530
                    General Result 8                           0.0057  0.5595
                    Worst Result   1                           0.0000  1.0000
```

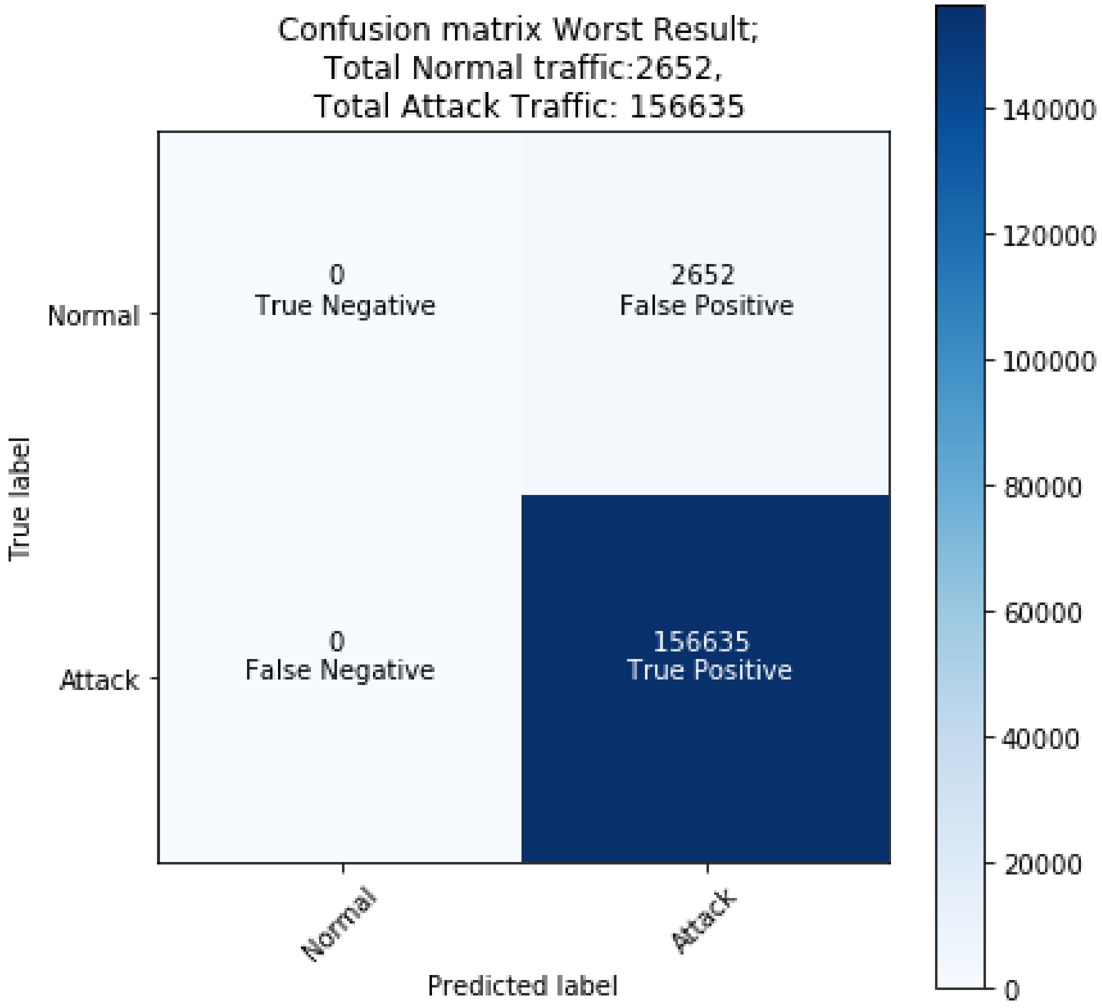



Results for VAE-LabelGeneration: General case
quality_score
time_taken (right)
0.000
0.001
0.002
0.003
0.004
0.005
23
24
25
26
27
28
29
30
31
One
10%
20%
40%
All
no_of_features


Results for VAE-LabelGeneration: Worst case
quality_score
time_taken (right)
0.0000
−0.0002
−0.0004
−0.0006
−0.0008
−0.0010
−0.0012
−0.0014
−0.0016
24
26
28
30
32
One
10%
20%
40%
All
no_of_features

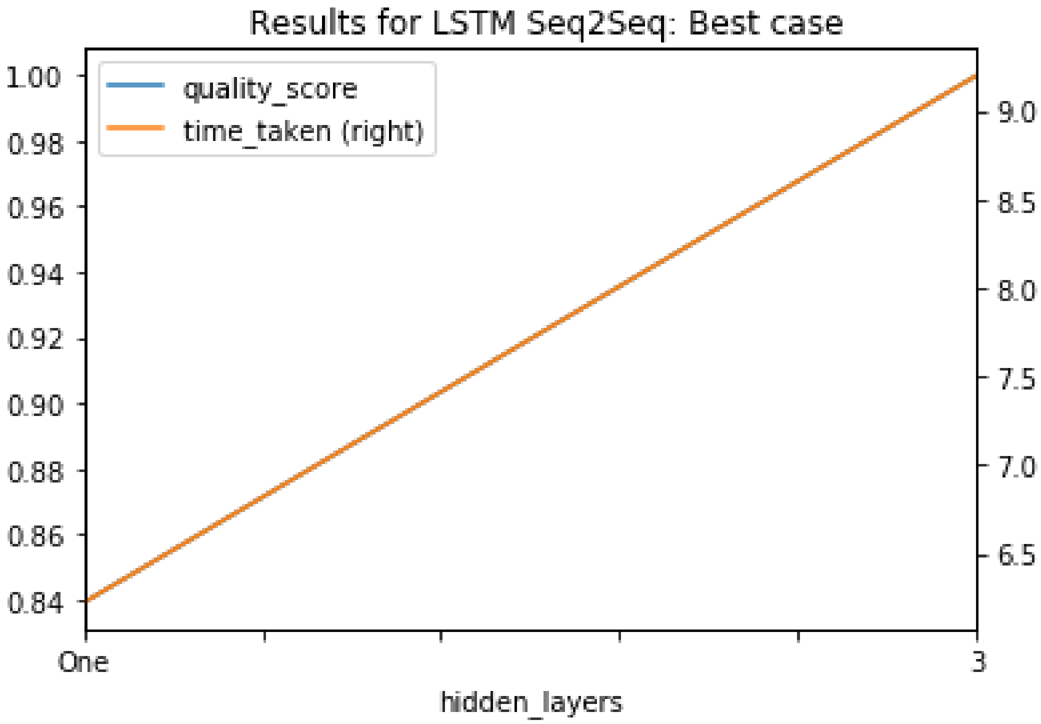

Confusion matrix Best Result;
Total Normal traffic:6604,
Total Attack Traffic: 210293
2571
True Negative
4033
False Positive
51944
False Negative
158349
True Positive
Normal
Attack
True label
Normal
Attack
Predicted label
140000
120000
100000
80000
60000
40000
20000

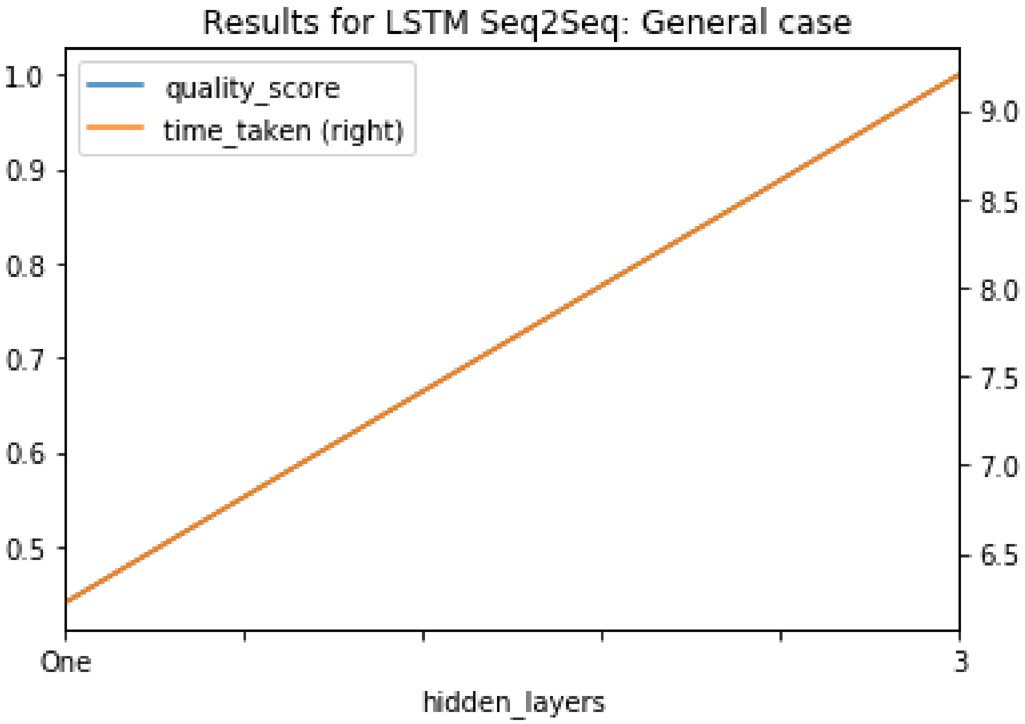

Confusion matrix General Result;
Total Normal traffic:3639,
Total Attack Traffic: 209001
1682
True Negative
1957
False Positive
92062
False Negative
116939
True Positive
Normal
Attack
True label
Normal
Attack
Predicted label
100000
80000
60000
40000
20000

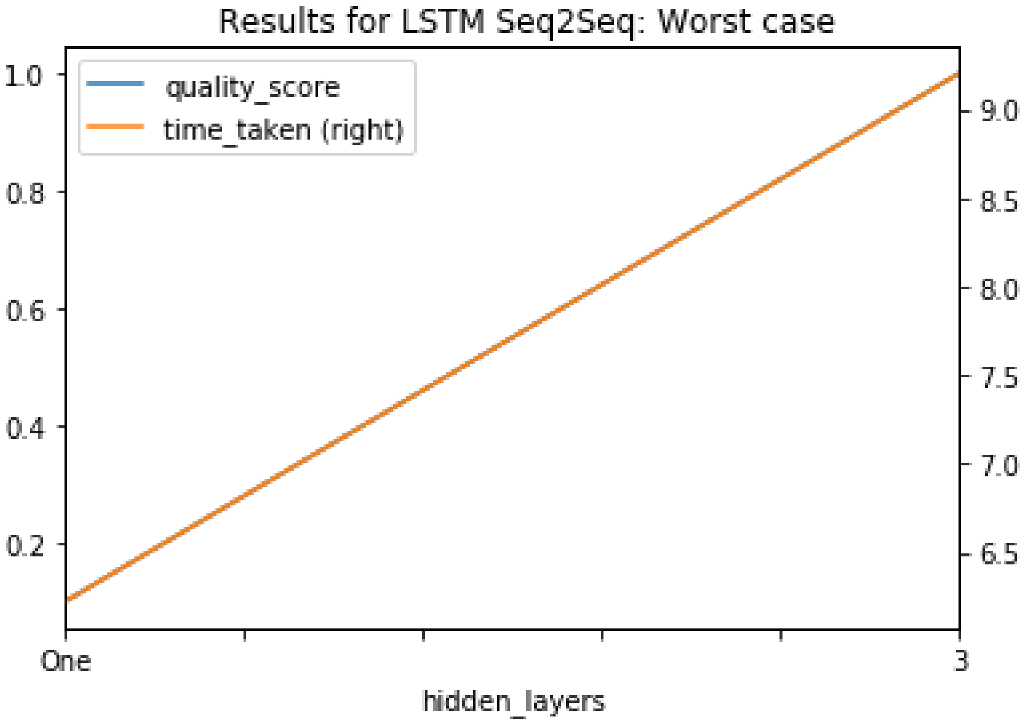


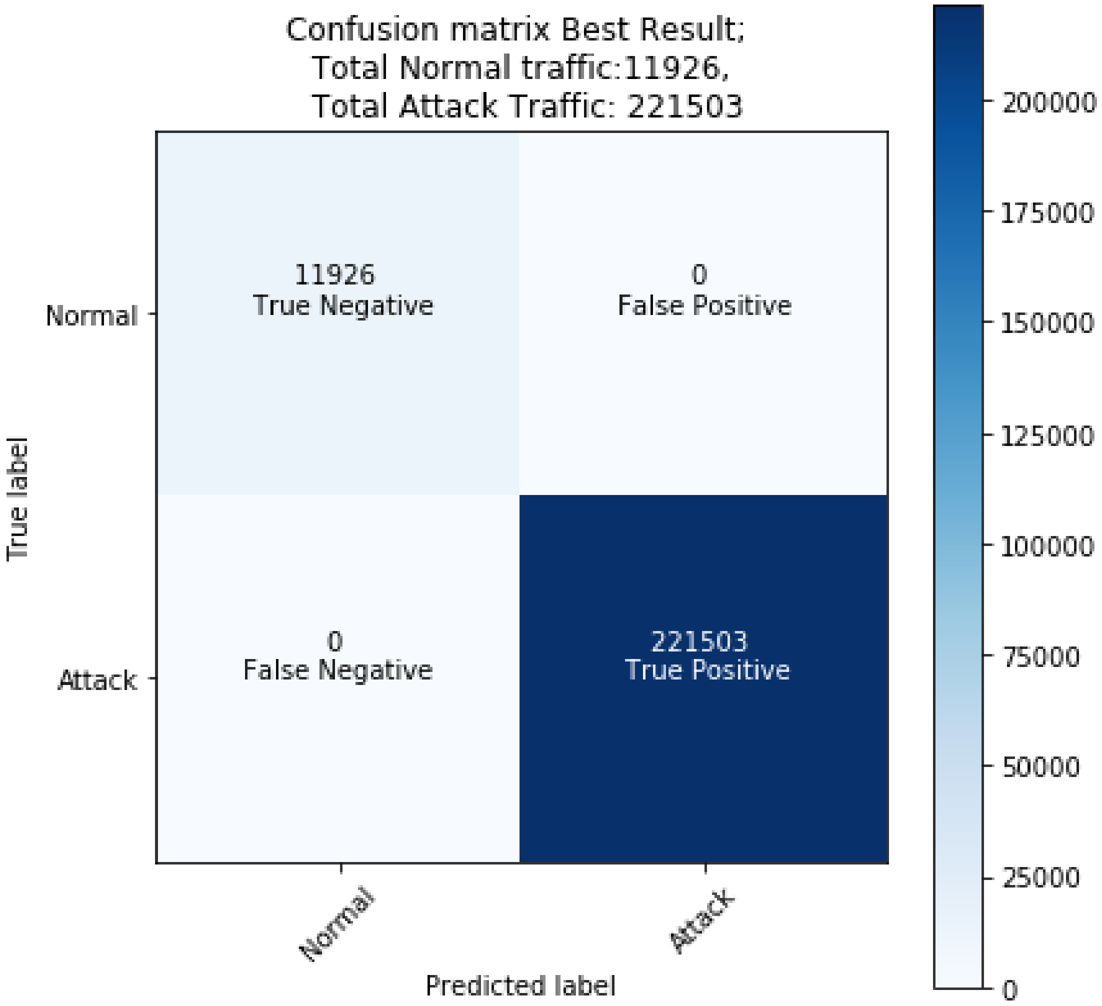
```
In [15]: past_scores = pd.read_pickle("dataset/scores/tf_lstm_nsl_kdd-orig_all.pkl")
         predictions = pd.read_pickle("dataset/tf_lstm_nsl_kdd_predictions.pkl")

In [16]: all_scenarios_lstm = evaluate_lstm("LSTM Seq2Seq", past_scores, predictions)

'Combined Results from all Scenarios for LSTM Seq2Seq'


                                                Accuracy  Precision  \
Model        Scenarios      Number of Features
LSTM Seq2Seq Best Result    1                        1.0        1.0
             General Result 1                        1.0        1.0
             Worst Result   1                        1.0        1.0

                                                Quality Score  Recall
Model        Scenarios      Number of Features
```

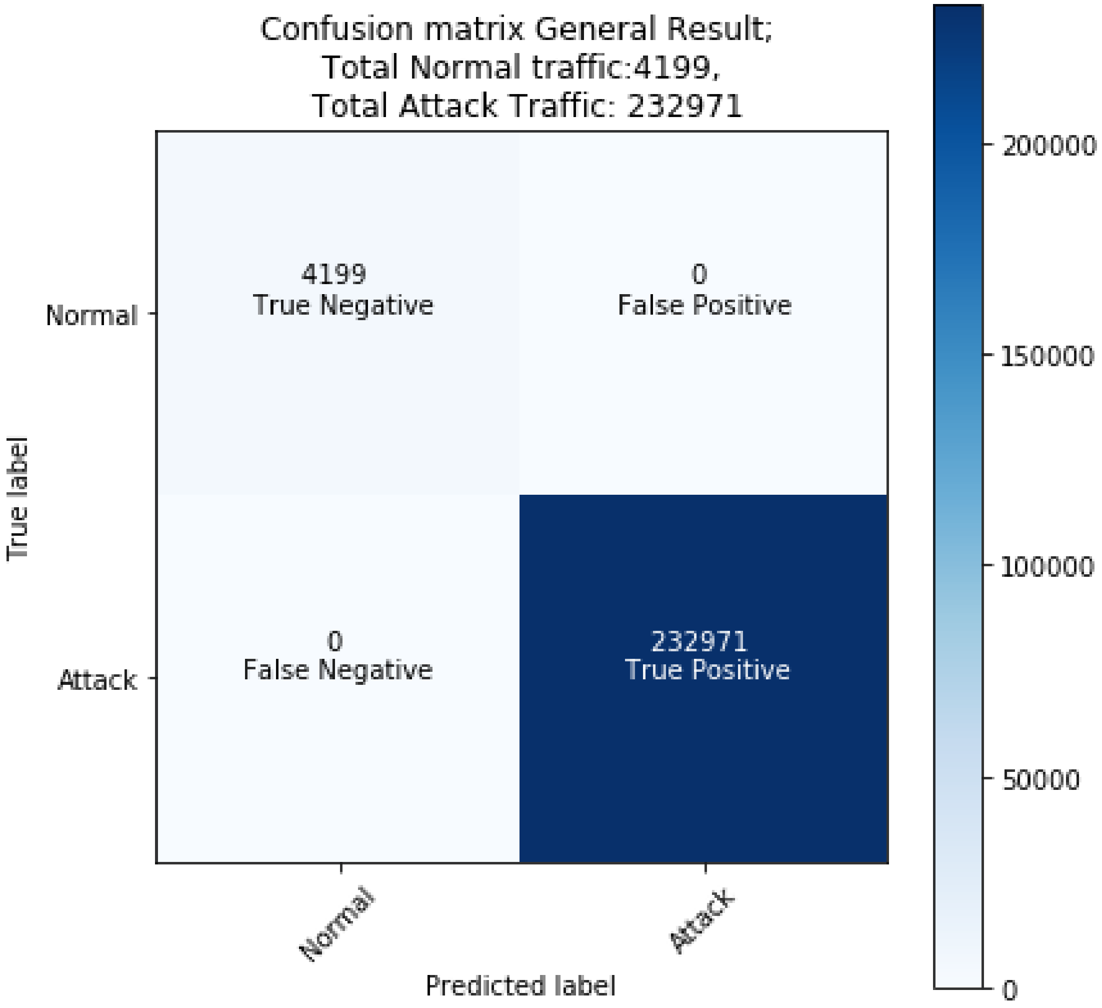
```
LSTM Seq2Seq Best Result    1                             1.0     1.0
             General Result 1                             1.0     1.0
             Worst Result   1                             1.0     1.0
```

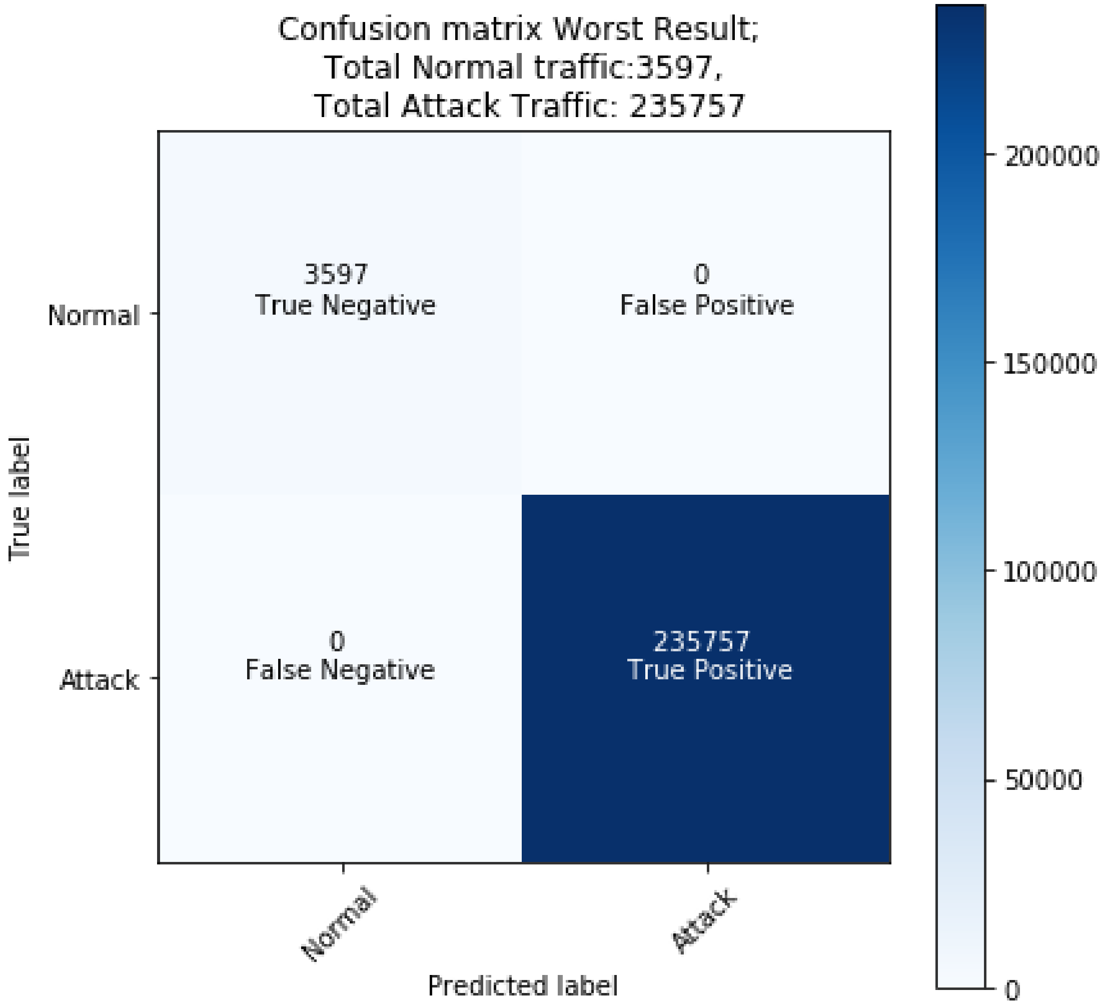



Results for LSTM Seq2Seq: General case
quality_score
time_taken (right)
1.0
0.9
0.8
0.7
0.6
0.5
9.0
8.5
8.0
7.5
7.0
6.5
One
3
hidden_layers


Results for LSTM Seq2Seq: Worst case
quality_score
time_taken (right)
1.0
0.8
0.6
0.4
0.2
9.0
8.5
8.0
7.5
7.0
6.5
One
3
hidden_layers


Confusion matrix Best Result;
Total Normal traffic:11926,
Total Attack Traffic: 221503
11926
True Negative
0
False Positive
0
False Negative
221503
True Positive
Normal
Attack
True label
Normal
Attack
Predicted label
200000
175000
150000
125000
100000
75000
50000
25000
0


Confusion matrix General Result;
Total Normal traffic:4199,
Total Attack Traffic: 232971
4199
True Negative
0
False Positive
0
False Negative
232971
True Positive
Normal
Attack
True label
Normal
Attack
Predicted label
200000
150000
100000
50000
0



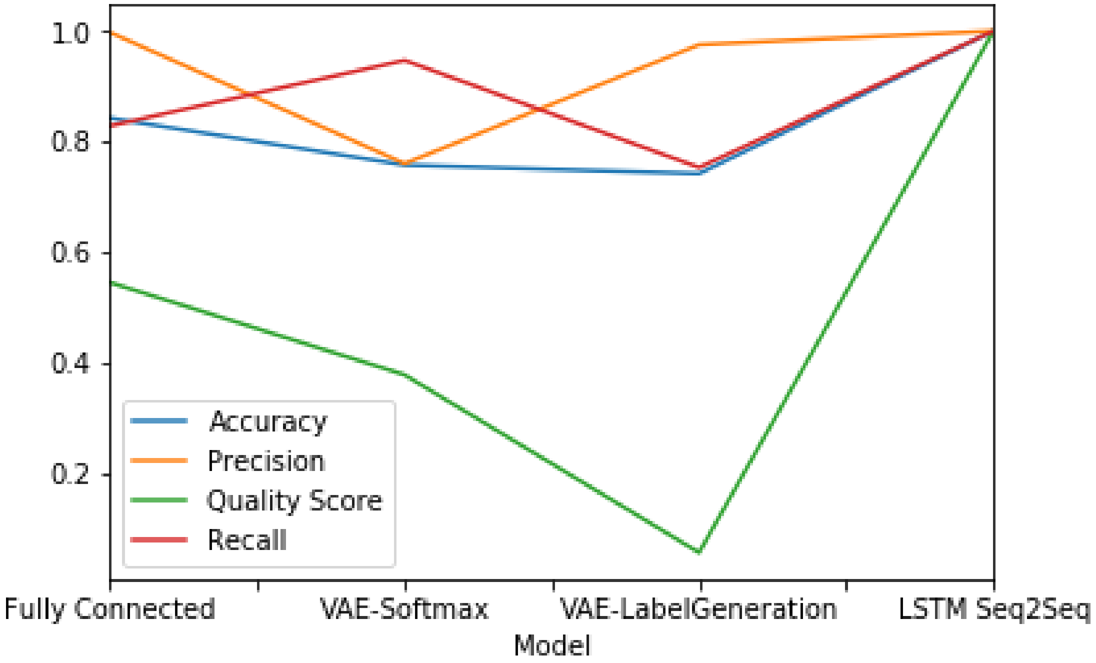
```
In [17]: all_scenarios = pd.concat([all_scenarios_fcn, all_scenarios_vae_sm, all_scenarios_vae

In [18]: all_scenarios_display = all_scenarios.set_index(['Model', 'Scenarios', 'Number of Fea
         #all_scenarios_display

In [19]: all_scenarios_best = all_scenarios.set_index(['Scenarios', 'Model']) #, 'Number of Fe
         all_scenarios_best.loc['Best Result']
```

```
Out[19]:                     Accuracy Number of Features  Precision  Quality Score  \
         Model
         Fully Connected       0.8424                  1     0.9973         0.5448
         VAE-Softmax           0.7574                  4     0.7600         0.3779
         VAE-LabelGeneration   0.7419                  1     0.9752         0.0564
         LSTM Seq2Seq          1.0000                  1     1.0000         1.0000

                               Recall
```

```
Model
Fully Connected      0.8284
VAE-Softmax          0.9468
VAE-LabelGeneration  0.7530
LSTM Seq2Seq         1.0000
```

```
In [20]: all_scenarios_best.loc['Best Result'].drop('Number of Features', axis = 1).plot()
```

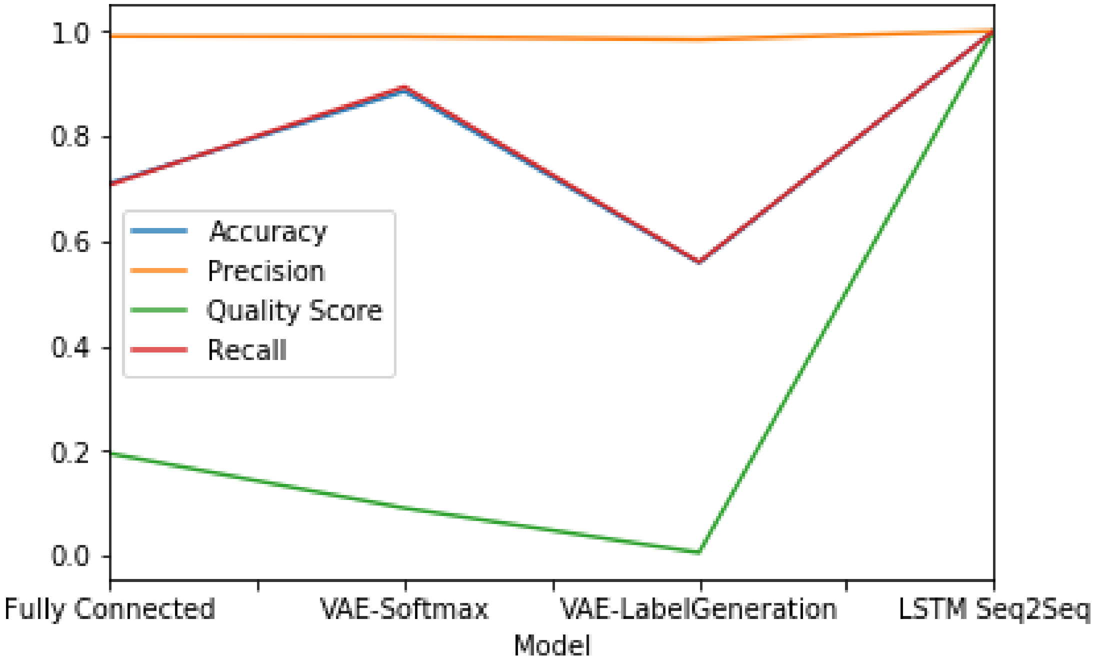
```
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7f35173a7780>
```



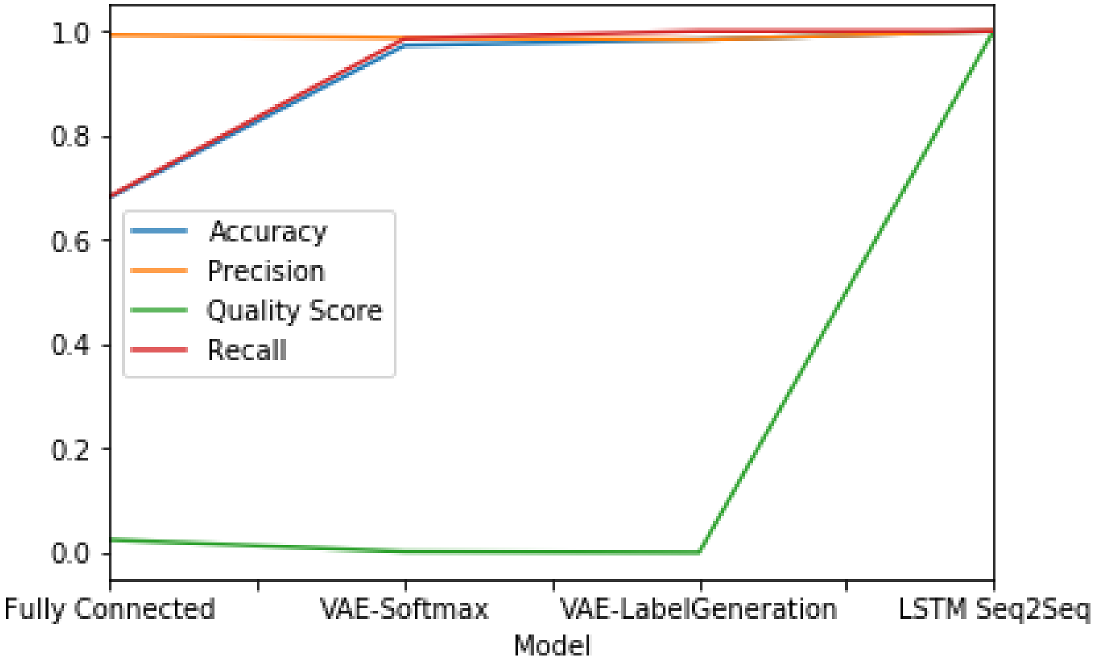
```
In [21]: all_scenarios_best.loc['General Result']
```

```
Out[21]:                      Accuracy  Number of Features  Precision  Quality Score  \
         Model
         Fully Connected        0.7098                   1     0.9899         0.1945
         VAE-Softmax            0.8853                   4     0.9891         0.0904
         VAE-LabelGeneration    0.5578                   8     0.9835         0.0057
         LSTM Seq2Seq           1.0000                   1     1.0000         1.0000

                              Recall
         Model
         Fully Connected      0.7069
         VAE-Softmax          0.8934
         VAE-LabelGeneration  0.5595
         LSTM Seq2Seq         1.0000
```

```
In [22]: all_scenarios_best.loc['General Result'].drop('Number of Features', axis = 1).plot()

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x7f35171e1b38>
```



```
In [23]: all_scenarios_best.loc['Worst Result']

Out[23]:                      Accuracy Number of Features  Precision  Quality Score  \
         Model
         Fully Connected        0.6815                  1     0.9922         0.0246
         VAE-Softmax            0.9723                 42     0.9867         0.0020
         VAE-LabelGeneration    0.9834                  1     0.9834         0.0000
         LSTM Seq2Seq           1.0000                  1     1.0000         1.0000

                             Recall
         Model
         Fully Connected     0.6838
         VAE-Softmax         0.9851
         VAE-LabelGeneration 1.0000
         LSTM Seq2Seq        1.0000

In [24]: all_scenarios_best.loc['Worst Result'].drop('Number of Features', axis = 1).plot()

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3517325c18>
```


1.0
0.8
0.6
0.4
0.2
0.0
Accuracy
Precision
Quality Score
Recall
Fully Connected
VAE-Softmax
VAE-LabelGeneration
LSTM Seq2Seq
Model

```
In [ ]:
```