```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns; sns.set()
from sklearn.metrics import confusion_matrix
%matplotlib inline
```

```
pip install lime
```

```
Collecting lime
  Downloading lime-0.2.0.1.tar.gz (275 kB)
     |████████████████████████████████| 275 kB 4.1 MB/s
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from li
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from li
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from lim
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: scikit-image>=0.12 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,>=4.3.0 in /usr/local/lib/python3.
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/l
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (
Building wheels for collected packages: lime
                                    up.py) ... done
                                    name=lime-0.2.0.1-py3-none-any.whl size=283857 sha256=a39
  Stored in directory: /root/.cache/pip/wheels/ca/cb/e5/ac701e12d365a08917bf4c6171c0961
Successfully built lime
Installing collected packages: lime
Successfully installed lime-0.2.0.1
```

Saved successfully!

```
from lime import lime_tabular
```

```
DDoS = pd.read_csv('/content/dataset_sdn.csv')
DDoS.head()
```

| | dt | switch | src | dst | pktcount | bytecount | dur | dur_nsec | tot_dur | f |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 11425 | 1 | 10.0.0.1 | 10.0.0.8 | 45304 | 48294064 | 100 | 716000000 | 1.010000e+11 | |
| **1** | 11605 | 1 | 10.0.0.1 | 10.0.0.8 | 126395 | 134737070 | 280 | 734000000 | 2.810000e+11 | |
| **2** | 11425 | 1 | 10.0.0.2 | 10.0.0.8 | 90333 | 96294978 | 200 | 744000000 | 2.010000e+11 | |
| **3** | 11425 | 1 | 10.0.0.2 | 10.0.0.8 | 90333 | 96294978 | 200 | 744000000 | 2.010000e+11 | |
| **4** | 11425 | 1 | 10.0.0.2 | 10.0.0.8 | 90333 | 96294978 | 200 | 744000000 | 2.010000e+11 | |

```
DDoS["rx_kbps"] = DDoS["rx_kbps"].fillna(DDoS["rx_kbps"].mean())
```

```
DDoS["tot_kbps"] = DDoS["tot_kbps"].fillna(DDoS["tot_kbps"].mean())


#Transformations of categorical features
DDoS['Protocol'] = DDoS['Protocol'].astype('category')
DDoS['src'] = DDoS['src'].astype('category')
DDoS['dst'] = DDoS['dst'].astype('category')

cat_columns = DDoS.select_dtypes(['category']).columns
DDoS[cat_columns] = DDoS[cat_columns].apply(lambda x: x.cat.codes)


from sklearn.model_selection import train_test_split
X = DDoS[['src','dst','dt', 'switch', 'pktcount', 'bytecount', 'dur', 'dur_nsec', 'tot_dur',
       'flows', 'packetins', 'pktperflow', 'byteperflow', 'pktrate',
       'Pairflow','Protocol' ,'port_no', 'tx_bytes', 'rx_bytes', 'tx_kbps', 'rx_kbps',
       'tot_kbps']]
y = DDoS['label']
```

Saved successfully!

```
train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling (or standardization)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


from sklearn.linear_model import LogisticRegression
model1=LogisticRegression()

#fit the model
model1.fit(X_train,y_train)

#predict
pred1 = model1.predict(X_test)

from sklearn.metrics import accuracy_score
```

```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
accuracy1 = accuracy_score(y_test, pred1)
print(' accuracy of Logistic regression = ', accuracy1)

precision1 = precision_score(y_test, pred1)
print(' precision of Logistic regression = ', precision1)

recall1 = recall_score(y_test, pred1)
print(' recall of Logistic regression = ',recall1 )

f11=f1_score(y_test, pred1, average='macro')
print('f1 score of Logistic regression = ',f11)
```

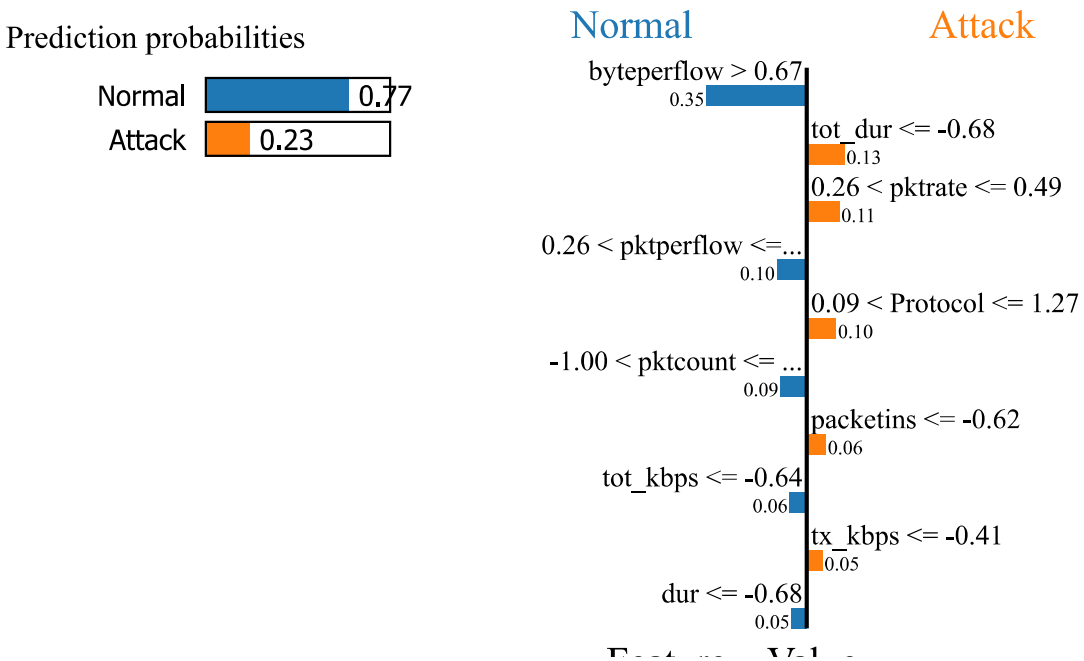
```
 accuracy of Logistic regression =  0.7699937706646222
 precision of Logistic regression =  0.7249025668592931
 recall of Logistic regression =  0.6620842027740272
f1 score of Logistic regression =  0.754257209178409
```

```
explainer = lime_tabular.LimeTabularExplainer(
 training_data=np.array(X_train),
 feature_names=DDoS.columns,
 class_names=['Normal', 'Attack'],
 mode='classification'
)



exp = explainer.explain_instance(
 data_row = X_test[0],
 predict_fn=model1.predict_proba
```

Saved successfully!

```
e)
```

```
exp = explainer.explain_instance(
 data_row = X_test[10000],
 predict_fn=model1.predict_proba
)
exp.show_in_notebook(show_table=True)
```




```
#ALGORITHM Gaussian NB
from sklearn.naive_bayes import GaussianNB
ML=GaussianNB()

#FIT DATA
```

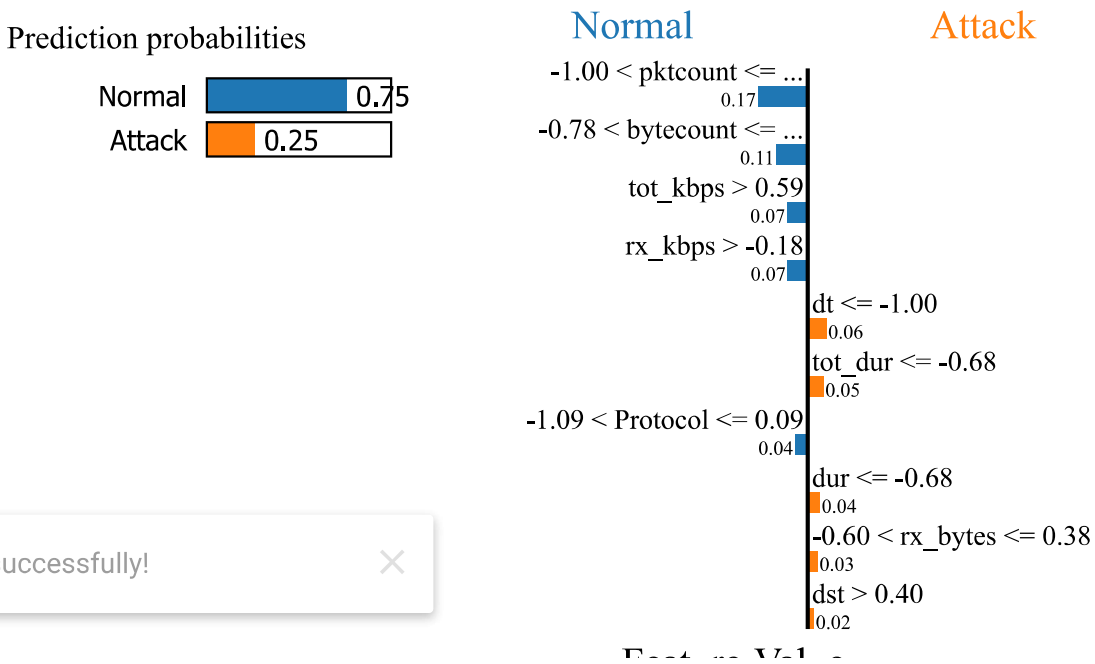
Saved successfully!

```
print("Prediction using GUASSIAN NB=",result)

# # Creating confusion matrix for evaluation
from sklearn.metrics import confusion_matrix
cm1 = confusion_matrix(y_test, result)

# # Print out confusion matrix and results
print("confusion matrix of Gaussian NB =", cm1)

accuracy = accuracy_score(y_test, result)
print('accuracy of Gaussian NB = ',accuracy)

precision = precision_score(y_test, result)
print('precision of Gaussian NB =',precision)

recall = recall_score(y_test, result)
print('recall of Gaussian NB =', recall)
```

```
f12=f1_score(y_test, result, average='macro')
print('f1 score of  Gaussian NB = ',f12)
```

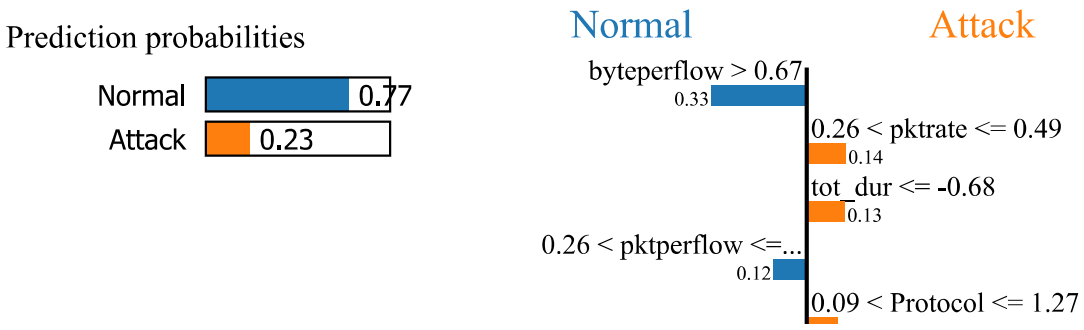
```
Prediction using GUASSIAN NB= [0 0 1 ... 0 0 1]
confusion matrix of Gaussian NB = [[8923 3799]
 [3245 4902]]
accuracy of Gaussian NB =  0.662465858450333
precision of Gaussian NB = 0.5633835191357315
recall of Gaussian NB = 0.6016938750460292
f1 score of  Gaussian NB =  0.6494518044638574
```

```
exp = explainer.explain_instance(
 data_row = X_test[0],
 predict_fn=ML.predict_proba
)
exp.show_in_notebook(show_table=True)
```




Saved successfully!

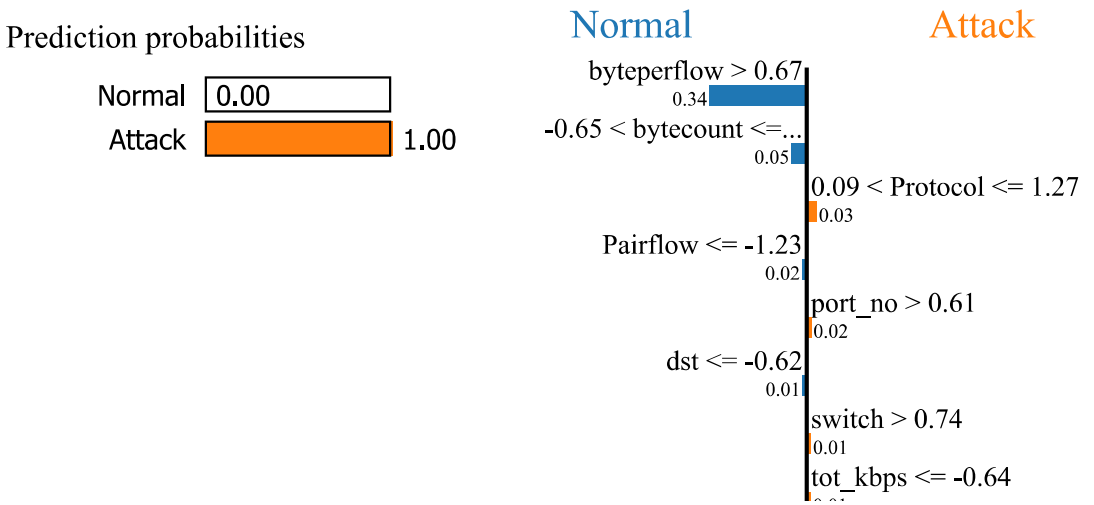
```
exp = explainer.explain_instance(
 data_row = X_test[10000],
 predict_fn=model1.predict_proba
)
exp.show_in_notebook(show_table=True)
```



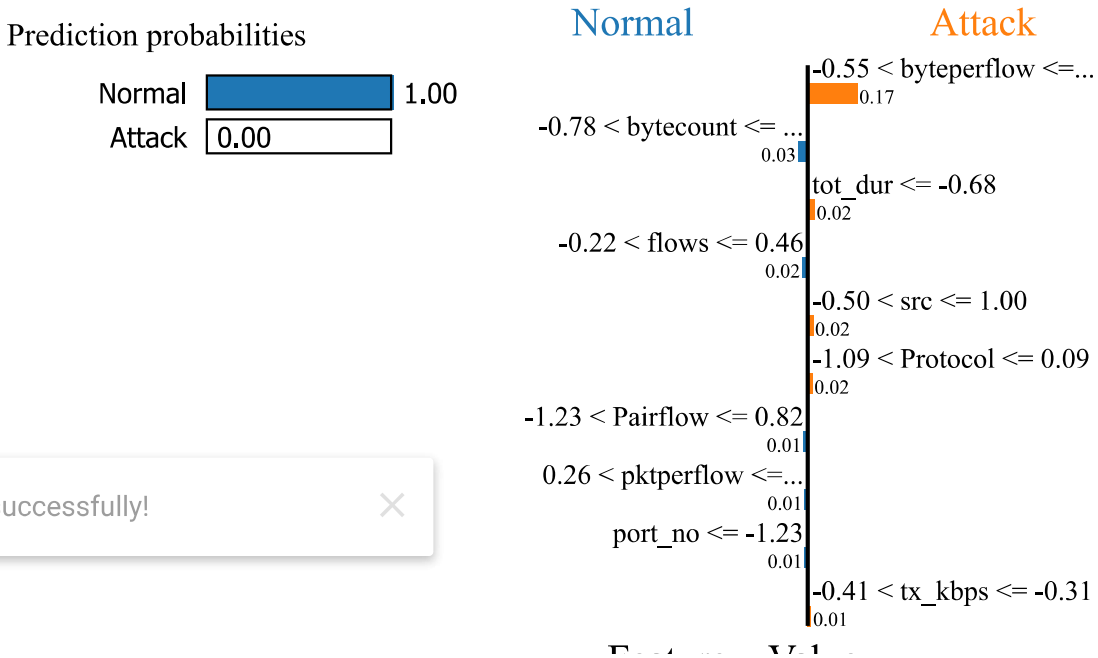

```
#algorithm(decision tree)
from sklearn.tree import DecisionTreeClassifier
Model3=DecisionTreeClassifier()
#fit data
Model3=Model3.fit(X_train,y_train)
#testing
result3=Model3.predict(X_test)

# # Creating confusion matrix for evaluation
cm2 = confusion_matrix(y_test, result3)
 # Print out confusion matrix and report
print('Confusion matrix of DT =' ,cm2)

accuracy3 = accuracy_score(y_test,result3)
print('accuracy of DT = ',accuracy3)

precision3 = precision_score(y_test, result3)
print('precision of DT =',precision3)

recall3 = recall_score(y_test, result3)
print('recall of DT =', recall3)

f14=f1_score(y_test, result3, average='macro')
```

Saved successfully!

```
Confusion matrix of DT = [[12722     0]
 [    0  8147]]
accuracy of DT =  1.0
precision of DT = 1.0
recall of DT = 1.0
f1 score of DT =  1.0


exp = explainer.explain_instance(
 data_row = X_test[10000],
 predict_fn=Model3.predict_proba
)
exp.show_in_notebook(show_table=True)
```




```
exp = explainer.explain_instance(
 data_row = X_test[0],
 predict_fn=Model3.predict_proba
)
exp.show_in_notebook(show_table=True)
```



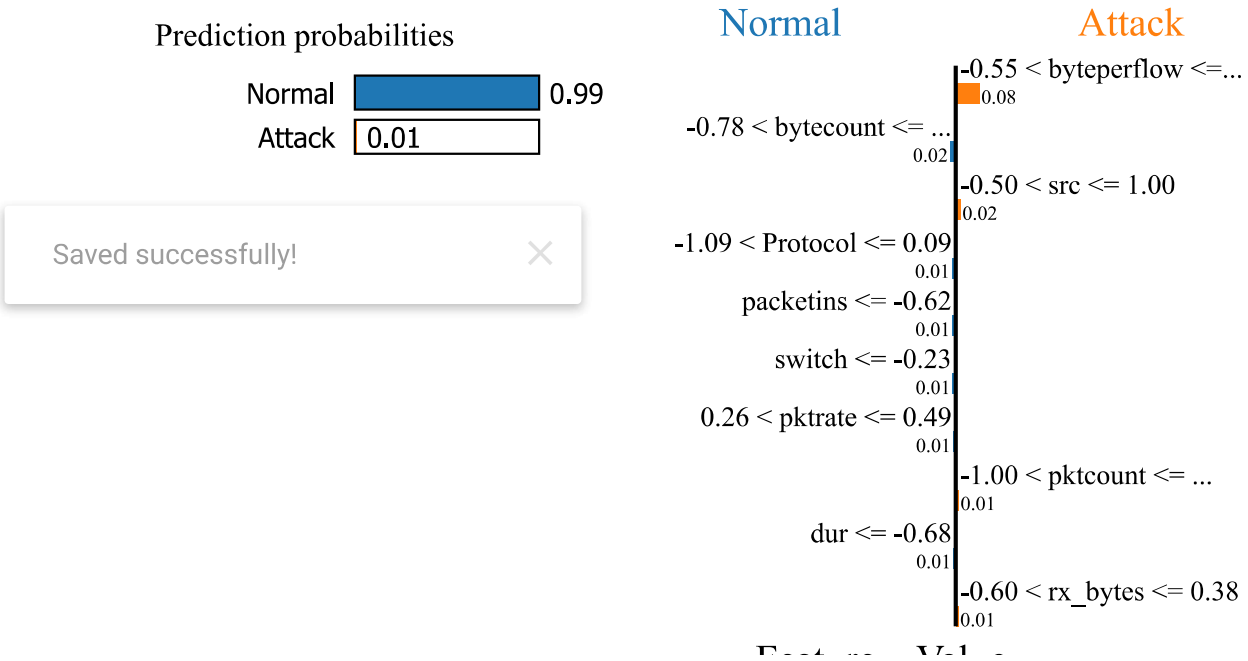

Saved successfully!

```
#Algorithm Randomforest

from sklearn.ensemble import RandomForestClassifier
Model4=RandomForestClassifier()
#fit data
Model4=Model4.fit(X_train,y_train)
#testing
result4=Model4.predict(X_test)

# # Creating confusion matrix for evaluation
cm4 = confusion_matrix(y_test, result4)
 # Print out confusion matrix and report
print('Confusion matrix of RF =' ,cm4)
```

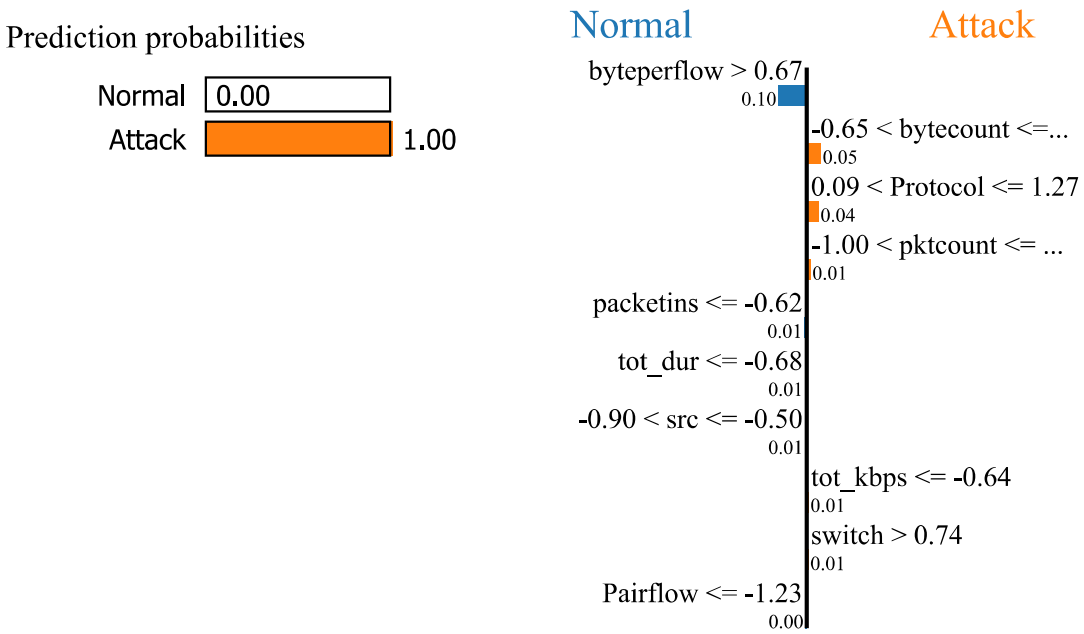
```
accuracy4 = accuracy_score(y_test,result4)
print('accuracy of RF = ',accuracy4)

precision4 = precision_score(y_test, result4)
print('precision of RF =',precision4)

recall4 = recall_score(y_test, result4)
print('recall of RF =', recall4)

f15=f1_score(y_test, result4, average='macro')
print('f1 score of RF = ',f15)
```

```
Confusion matrix of RF = [[12722     0]
 [    0  8147]]
accuracy of RF =  1.0
precision of RF = 1.0
recall of RF = 1.0
f1 score of RF =  1.0
```

```
exp = explainer.explain_instance(
 data_row = X_test[0],
 predict_fn=Model4.predict_proba
)
exp.show_in_notebook(show_table=True)
```




```
exp = explainer.explain_instance(
 data_row = X_test[10000],
 predict_fn=Model4.predict_proba
)
exp.show_in_notebook(show_table=True)
```



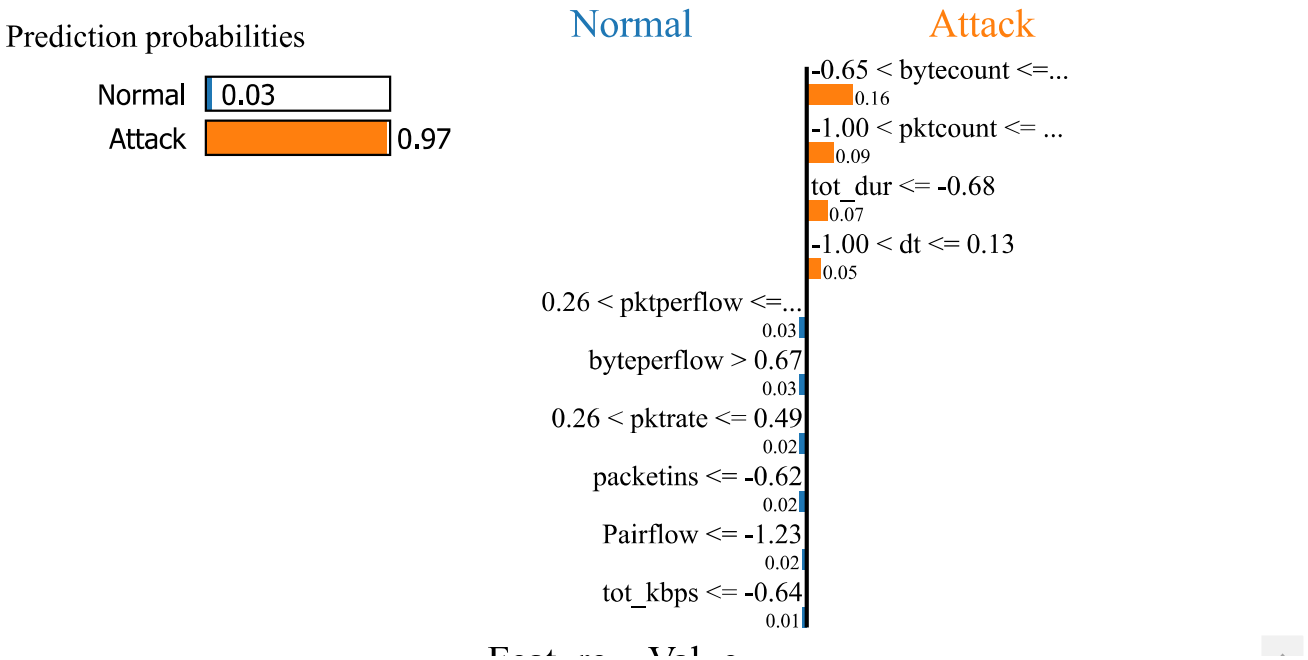

```
#algorithm Gradient Boosting
from sklearn.ensemble import GradientBoostingClassifier


Model5 = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,  max_depth=1, random

#testing
result5=Model5.predict(X_test)

# # Creating confusion matrix for evaluation
cm5 = confusion_matrix(y_test, result5)
 # Print out confusion matrix and report
print('Confusion matrix of Gradient Boosting =' ,cm5)

accuracy5 = accuracy_score(y_test, result5)
print('accuracy of Gradient Boosting = ',accuracy5)

precision5 = precision_score(y_test, result5)
print('precision of Gradient Boosting =',precision5)

recall5 = recall_score(y_test, result5)
print('recall of Gradient Boosting =', recall5)

f16=f1_score(y_test, result5, average='macro')
print('f1 score of Gradient Boosting = ',f16)
```

```
Confusion matrix of Gradient Boosting = [[12555   167]
 [   78  8069]]
accuracy of Gradient Boosting =  0.9882600987110067
precision of Gradient Boosting = 0.9797231665857212
recall of Gradient Boosting = 0.9904259236528784
f1 score of Gradient Boosting =  0.9876913427811314
```

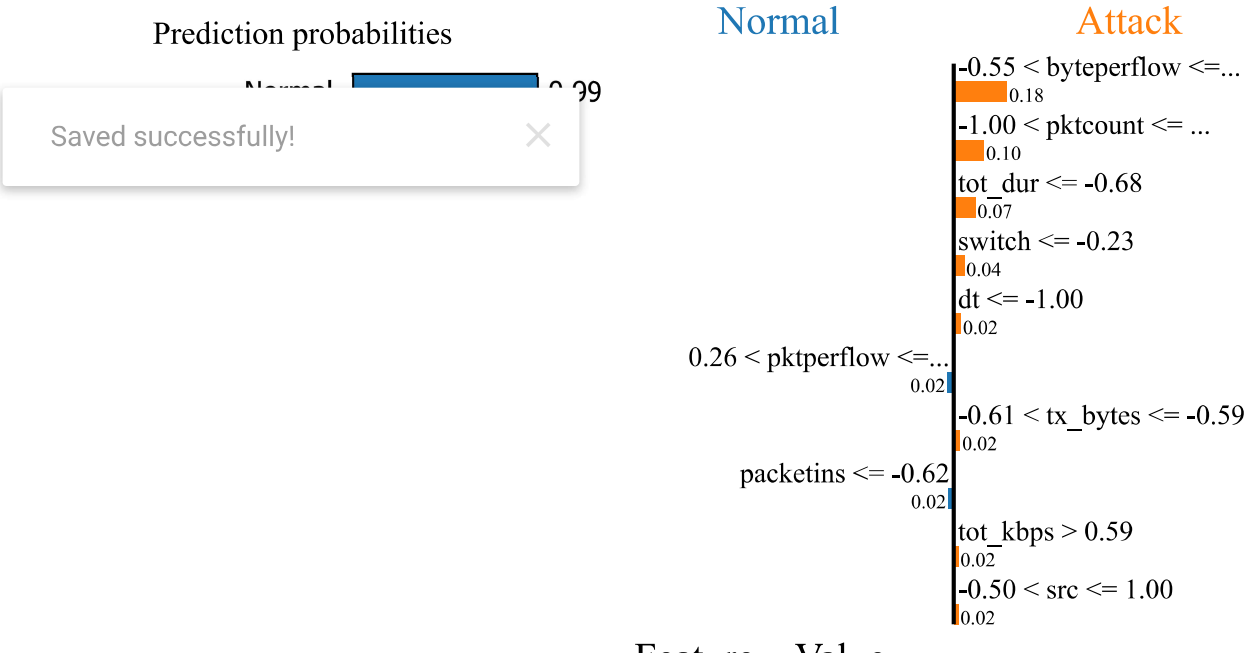
```
exp = explainer.explain_instance(
```

Saved successfully!

```
 data_row = X_test[10000],
 predict_fn=Model5.predict_proba
)
exp.show_in_notebook(show_table=True)
```




```
exp = explainer.explain_instance(
 data_row = X_test[0],
 predict_fn=Model5.predict_proba
)
exp.show_in_notebook(show_table=True)
```



✓ 3s completed at 1:09 PM

Saved successfully!