In [60]:

```
import pandas as pd
import numpy as np
from sklearn.metrics import classification_report as report
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ParameterGrid
from sklearn.metrics import  confusion_matrix as matrix
from sklearn.metrics import average_precision_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from matplotlib import pyplot
from sklearn.linear_model import LogisticRegression
RANDOM_STATE = 42
np.random.seed(seed=RANDOM_STATE)
```

Dataset1:http

load training/testing dataset

In [61]:

```
http_train=pd.read_csv('data/http_train.csv',sep=',')
http_test=pd.read_csv('data/http_test.csv',sep=',')

#train_raw_data = http_train.drop(http_train.index[0])
train_data = http_train.drop(http_train.columns[-1],axis='columns')
train_label = http_train.iloc[:,-1]

#test_raw_data = http_test.drop(http_test.index[0])
test_data = http_test.drop(http_test.columns[-1],axis='columns')
test_label = http_test.iloc[:,-1]

http_train["3"].value_counts()

#train_label.head()
```

Out[61]:

```
0    452229
1      1769
Name: 3, dtype: int64
```

train a model

In [62]:

```
lg_model=LogisticRegression(random_state=0, multi_class='ovr')
# Create regularization penalty space
penalty = ['l1', 'l2']
# Create regularization hyperparameter space
C = np.logspace(0, 4, 10)
# Create hyperparameter options
hyperparameters = dict(C=C, penalty=penalty)
#lg_model = GridSearchCV(lg_model, hyperparameters,scoring='average_precision', cv=5
lg_model.fit(train_data,train_label)
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-p
ackages/sklearn/linear_model/logistic.py:432: FutureWarning: Default s
olver will be changed to 'lbfgs' in 0.22. Specify a solver to silence
this warning.
  FutureWarning)
```

Out[62]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept
=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='ovr', n_jobs=None, penalty='l2', rando
m_state=0,
                   solver='warn', tol=0.0001, verbose=0, warm_start=Fa
lse)
```

make predict

In [63]:

```
result_model=lg_model.predict(test_data)
report_model=report(test_label,result_model,digits=5)
print(report_model)
```

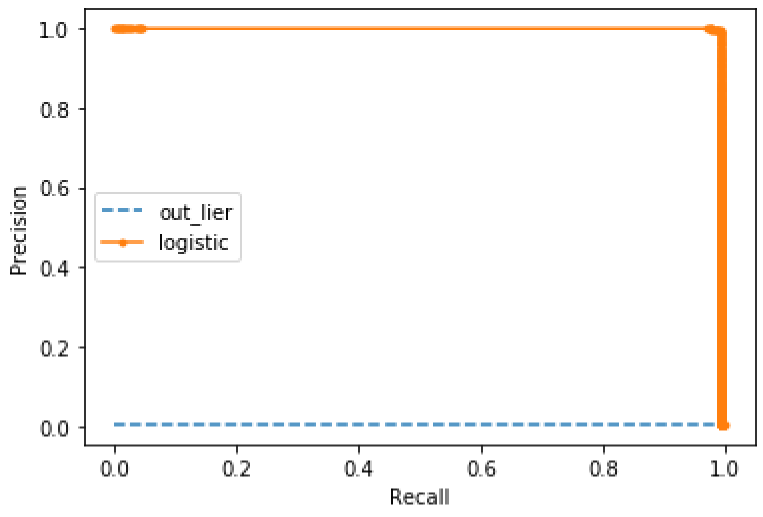
```
              precision    recall  f1-score   support

           0    0.99997   0.99998   0.99998    113058
           1    0.99546   0.99321   0.99434       442

    accuracy                        0.99996    113500
   macro avg    0.99772   0.99660   0.99716    113500
weighted avg    0.99996   0.99996   0.99996    113500
```

Plot

In [64]:

```
# predict probabilities
lr_probs = lg_model.predict_proba(test_data)
# keep probabilities for the positive outcome only
lr_probs = lr_probs[:, 1]
# predict class values
result_model = lg_model.predict(test_data)
lr_precision, lr_recall, _ = precision_recall_curve(test_label, lr_probs)
lr_f1, lr_auc = f1_score(test_label, result_model), auc(lr_recall, lr_precision)
# summarize scores
print('Logistic: f1=%.3f auc=%.3f' % (lr_f1, lr_auc))
# plot the precision-recall curves
out_lier = len(test_label[test_label==1]) / len(test_label)
pyplot.plot([0, 1], [out_lier, out_lier], linestyle='--', label='out_lier')
pyplot.plot(lr_recall, lr_precision, marker='.', label='logistic')
# axis labels
pyplot.xlabel('Recall')
pyplot.ylabel('Precision')
# show the legend
pyplot.legend()
# show the plot
pyplot.show()
```

```
Logistic: f1=0.994 auc=0.993
```



Dataset2: Cardio

load training/testing dataset

In [65]:

```
cardio_train=pd.read_csv('data/cardio_train.csv',sep=',')
cardio_test=pd.read_csv('data/cardio_test.csv',sep=',')

#train_raw_data = http_train.drop(http_train.index[0])
train_data = cardio_train.drop(cardio_train.columns[-1],axis='columns')
train_label = cardio_train.iloc[:,-1]

#test_raw_data = http_test.drop(http_test.index[0])
test_data = cardio_test.drop(cardio_test.columns[-1],axis='columns')
test_label = cardio_test.iloc[:,-1]

#train_data.head()
cardio_train["21"].value_counts()
```

Out[65]:

```
0    1323
1     141
Name: 21, dtype: int64
```

train a model

In [66]:

```
lg_model=LogisticRegression(random_state=0,multi_class='ovr')
# Create regularization penalty space
penalty = ['l1', 'l2']
# Create regularization hyperparameter space
C = np.logspace(0, 4, 10)
# Create hyperparameter options
hyperparameters = dict(C=C, penalty=penalty)
# lg_model = GridSearchCV(lg_model, hyperparameters,scoring='average_precision', cv=
lg_model.fit(train_data,train_label)
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-p
ackages/sklearn/linear_model/logistic.py:432: FutureWarning: Default s
olver will be changed to 'lbfgs' in 0.22. Specify a solver to silence
this warning.
  FutureWarning)
```

Out[66]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept
=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='ovr', n_jobs=None, penalty='l2', rando
m_state=0,
                   solver='warn', tol=0.0001, verbose=0, warm_start=Fa
lse)
```

make predict

In [67]:

```
result_model=lg_model.predict(test_data)
report_model=report(test_label,result_model,digits=5)
print(report_model)
```

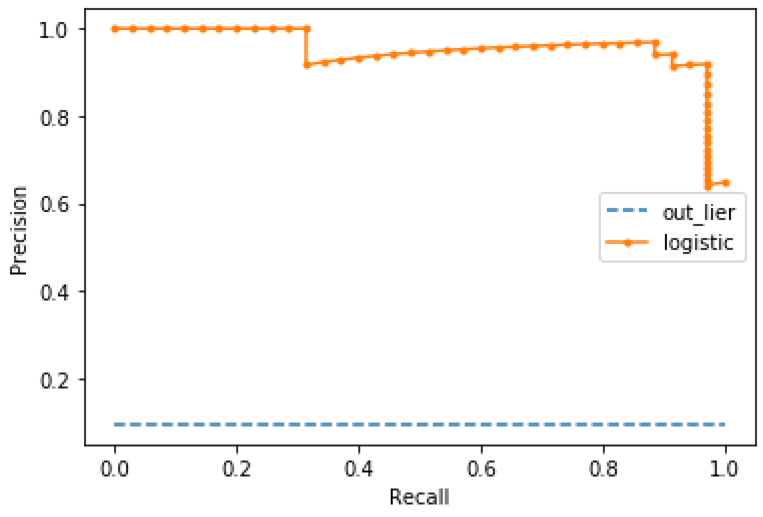
```
              precision    recall  f1-score   support

           0    0.99396   0.99096   0.99246       332
           1    0.91667   0.94286   0.92958        35

    accuracy                        0.98638       367
   macro avg    0.95531   0.96691   0.96102       367
weighted avg    0.98659   0.98638   0.98646       367
```

plot

In [68]:

```
# predict probabilities
lr_probs = lg_model.predict_proba(test_data)
# keep probabilities for the positive outcome only
lr_probs = lr_probs[:, 1]
# predict class values
result_model = lg_model.predict(test_data)
lr_precision, lr_recall, _ = precision_recall_curve(test_label, lr_probs)
lr_f1, lr_auc = f1_score(test_label, result_model), auc(lr_recall, lr_precision)
# summarize scores
print('Logistic: f1=%.3f auc=%.3f' % (lr_f1, lr_auc))
# plot the precision-recall curves
out_lier = len(test_label[test_label==1]) / len(test_label)
pyplot.plot([0, 1], [out_lier, out_lier], linestyle='--', label='out_lier')
pyplot.plot(lr_recall, lr_precision, marker='.', label='logistic')
# axis labels
pyplot.xlabel('Recall')
pyplot.ylabel('Precision')
# show the legend
pyplot.legend()
# show the plot
pyplot.show()
```

```
Logistic: f1=0.930 auc=0.955
```



Dataset3: Credit Cards

load training/testing dataset

In [69]:

```
creditCard_train=pd.read_csv('data/credit_train.csv',sep=',')
creditCard_test=pd.read_csv('data/credit_test.csv',sep=',')

#train_raw_data = http_train.drop(http_train.index[0])

train_data = creditCard_train.drop(creditCard_train.columns[[0,-1,-2]],axis=1)
train_label = creditCard_train.iloc[:,-1]

#test_raw_data = http_test.drop(http_test.index[0])
test_data = creditCard_test.drop(creditCard_test.columns[[0,-1,-2]],axis=1)
test_label = creditCard_test.iloc[:,-1]

#train_data.head()
train_label.head()
creditCard_train.head()
```

Out[69]:

	Time	V1	V2	V3	V4	V5	V6	V7	V
0	161919.0	1.946747	-0.752526	-1.355130	-0.661630	1.502822	4.024933	-1.479661	1.13988
1	124477.0	2.035149	-0.048880	-3.058693	0.247945	2.943487	3.298697	-0.002192	0.67478
2	41191.0	-0.991920	0.603193	0.711976	-0.992425	-0.825838	1.956261	-2.212603	-5.03752
3	132624.0	2.285718	-1.500239	-0.747565	-1.668119	-1.394143	-0.350339	-1.427984	0.0100
4	59359.0	-0.448747	-1.011440	0.115903	-3.454854	0.715771	-0.147490	0.504347	-0.1138

5 rows × 31 columns

make predict

In [70]:

```
lg_model=LogisticRegression(random_state=0, multi_class='ovr')
# Create regularization penalty space
penalty = ['l1', 'l2']
# Create regularization hyperparameter space
C = np.logspace(0, 4, 10)
# Create hyperparameter options
hyperparameters = dict(C=C, penalty=penalty)
#lg_model = GridSearchCV(lg_model, hyperparameters, scoring='average_precision', cv=
lg_model.fit(train_data,train_label)
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-p
ackages/sklearn/linear_model/logistic.py:432: FutureWarning: Default s
olver will be changed to 'lbfgs' in 0.22. Specify a solver to silence
this warning.
  FutureWarning)
```

Out[70]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept
=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='ovr', n_jobs=None, penalty='l2', rando
m_state=0,
                   solver='warn', tol=0.0001, verbose=0, warm_start=Fa
lse)
```

make predict

In [71]:

```
result_model=lg_model.predict(test_data)
report_model=report(test_label,result_model,digits=5)
print(report_model)
```

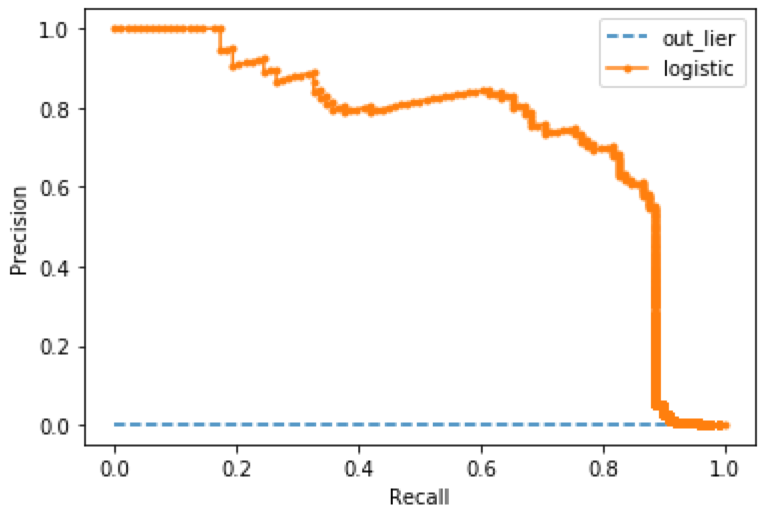
```
              precision    recall  f1-score   support

           0    0.99938   0.99977   0.99958     56864
           1    0.82895   0.64286   0.72414        98

    accuracy                        0.99916     56962
   macro avg    0.91417   0.82131   0.86186     56962
weighted avg    0.99909   0.99916   0.99910     56962
```

Plot

In [72]:

```
# predict probabilities
lr_probs = lg_model.predict_proba(test_data)
# keep probabilities for the positive outcome only
lr_probs = lr_probs[:, 1]
# predict class values
result_model = lg_model.predict(test_data)
lr_precision, lr_recall, _ = precision_recall_curve(test_label, lr_probs)
lr_f1, lr_auc = f1_score(test_label, result_model), auc(lr_recall, lr_precision)
# summarize scores
print('Logistic: f1=%.3f auc=%.3f' % (lr_f1, lr_auc))
# plot the precision-recall curves
out_lier = len(test_label[test_label==1]) / len(test_label)
pyplot.plot([0, 1], [out_lier, out_lier], linestyle='--', label='out_lier')
pyplot.plot(lr_recall, lr_precision, marker='.', label='logistic')
# axis labels
pyplot.xlabel('Recall')
pyplot.ylabel('Precision')
# show the legend
pyplot.legend()
# show the plot
pyplot.show()
```

```
Logistic: f1=0.724 auc=0.744
```


out_lier
logistic
Precision
Recall