Credit card fraud detection

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
dataset = pd.read_csv('creditcard.csv')
dataset
```

	Time	V1	V2	V3	V4	V5	V6	V
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.23959
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.07880
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.79146
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.23760
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.59294
...	...	...	...	...	...	...	...	.
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.91821
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.02433
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.29682
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.68618
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.57700

284807 rows × 31 columns

```
import pandas as pd
import numpy as np

dataset.dropna()
```

	Time	V1	V2	V3	V4	V5	V6	V
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.23959
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.07880
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.79146
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.23760
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.59294
...	...	...	...	...	...	...	...	.
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.91821
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.02433
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.29682
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.68618

```
print('Fraud:', len(dataset[dataset['Class'] == 1]))
print('Non-Fraud:', len(dataset[dataset['Class'] == 0]))
```

```
Fraud: 492
Non-Fraud: 284315
```

```
X = dataset.iloc[:, :-1].values
y = dataset['Class'].values
print(X.shape)
print(y.shape)
```

```
(284807, 30)
(284807,)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Log Regression

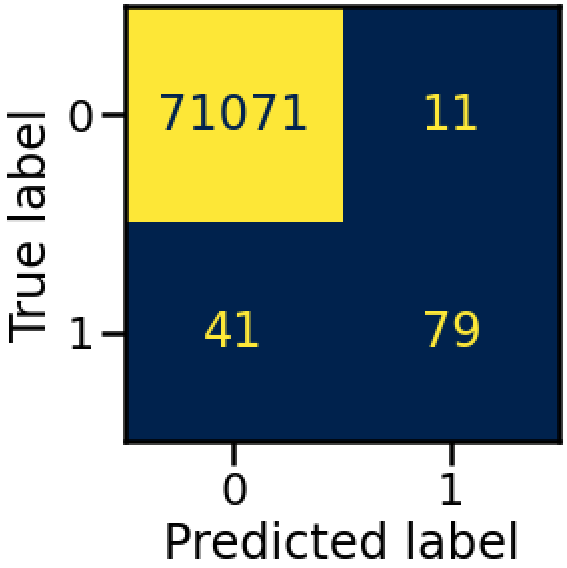
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(random_state=0)
lr.fit(X_train, y_train)
```

```
     LogisticRegression(random_state=0)
```

```
y_pred_lr = lr.predict(X_test)
```

```
from sklearn.metrics import plot_confusion_matrix
sns.set_context("poster")
disp = plot_confusion_matrix(lr, X_test, y_test, cmap='cividis', colorbar=False)
```

```
     /usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: F
       warnings.warn(msg, category=FutureWarning)
```

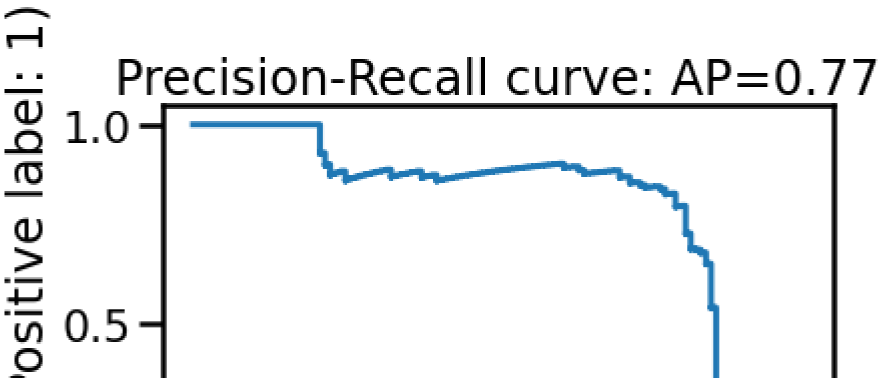


```
from sklearn.metrics import average_precision_score
y_score = lr.decision_function(X_test)
avg_precision = average_precision_score(y_test, y_score)
print('Average precision-recall score:', avg_precision)
```

```
     Average precision-recall score: 0.77211117725839
```

```
from sklearn.metrics import plot_precision_recall_curve
disp = plot_precision_recall_curve(lr, X_test, y_test)
disp.ax_.set_title('Precision-Recall curve: ' 'AP={0:0.2f}'.format(avg_precision))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: F
  warnings.warn(msg, category=FutureWarning)
Text(0.5, 1.0, 'Precision-Recall curve: AP=0.77')
```



```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
acc = accuracy_score(y_test, y_pred_lr)
prec = precision_score(y_test, y_pred_lr)
rec = recall_score(y_test, y_pred_lr)
f1 = f1_score(y_test, y_pred_lr)
print('accuracy:', acc)
print('precision:', prec)
print('recall:', rec)
print('f1_score:', f1)
```

```
accuracy: 0.9992696834358585
precision: 0.8777777777777778
recall: 0.6583333333333333
f1_score: 0.7523809523809525
```

```
results = pd.DataFrame([['LogisticsRegression', rec, prec, f1]],
               columns = ['Model', 'Recall', 'Precision', 'F1 Score'])
```

Oversampling

```
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=0)
X_train_resampledOS, y_train_resampledOS = ros.fit_resample(X_train, y_train)
```

```
from sklearn.linear_model import LogisticRegression
lr_resampledOS = LogisticRegression(random_state=0)
lr_resampledOS.fit(X_train_resampledOS, y_train_resampledOS)
```

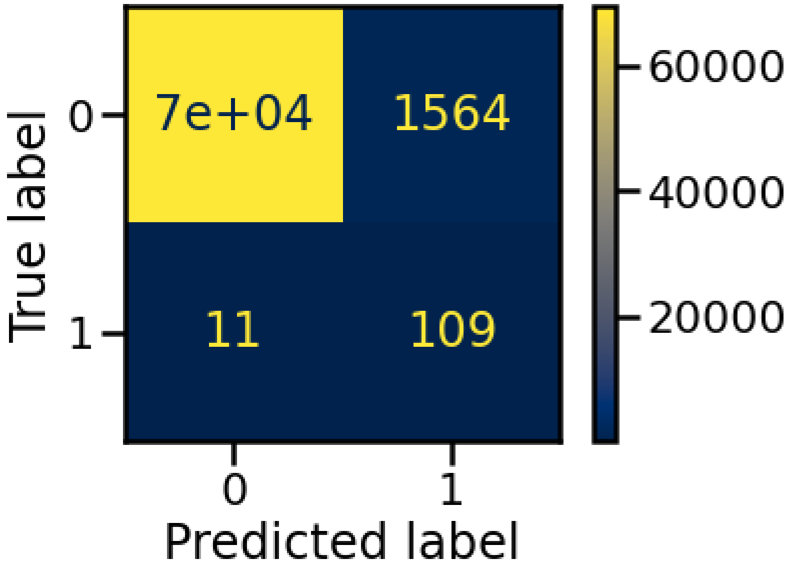
```
LogisticRegression(random_state=0)
```

```
y_pred_lrOS = lr_resampledOS.predict(X_test)
```

```
from sklearn.metrics import plot_confusion_matrix
sns.set_context("poster")
```

```
disp = plot_confusion_matrix(lr_resampledOS, X_test, y_test, cmap='cividis', colorbar=True)
```

```
    /usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: F
      warnings.warn(msg, category=FutureWarning)
```



```
from sklearn.metrics import average_precision_score
y_score = lr_resampledOS.decision_function(X_test)
avg_precision = average_precision_score(y_test, y_score)
print('Average precision-recall score:', avg_precision)
```

```
    Average precision-recall score: 0.7595760417231494
```

```
from sklearn.metrics import plot_precision_recall_curve
disp = plot_precision_recall_curve(lr_resampledOS, X_test, y_test)
disp.ax_.set_title('Precision-Recall curve: ' 'AP={0:0.2f}'.format(avg_precision))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: F
  warnings.warn(msg, category=FutureWarning)
Text(0.5, 1.0, 'Precision-Recall curve: AP=0.76')
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
acc = accuracy_score(y_test, y_pred_lrOS)
prec = precision_score(y_test, y_pred_lrOS)
rec = recall_score(y_test, y_pred_lrOS)
f1 = f1_score(y_test, y_pred_lrOS)
print('accuracy:', acc)
print('precision:', prec)
print('recall:', rec)
print('f1_score:', f1)
```

```
accuracy: 0.9778798348361001
precision: 0.06515242080095636
recall: 0.9083333333333333
f1_score: 0.12158393753485776
```

```
model_results = pd.DataFrame([['OverSampledLogisticsRegression', rec, prec, f1]],
               columns = ['Model', 'Recall', 'Precision', 'F1 Score'])
results = results.append(model_results, ignore_index = True)
```

Undersampling

```
from imblearn.under_sampling import RandomUnderSampler
rus = RandomUnderSampler(random_state=0)
X_train_resampledUS, y_train_resampledUS = rus.fit_resample(X_train, y_train)
```

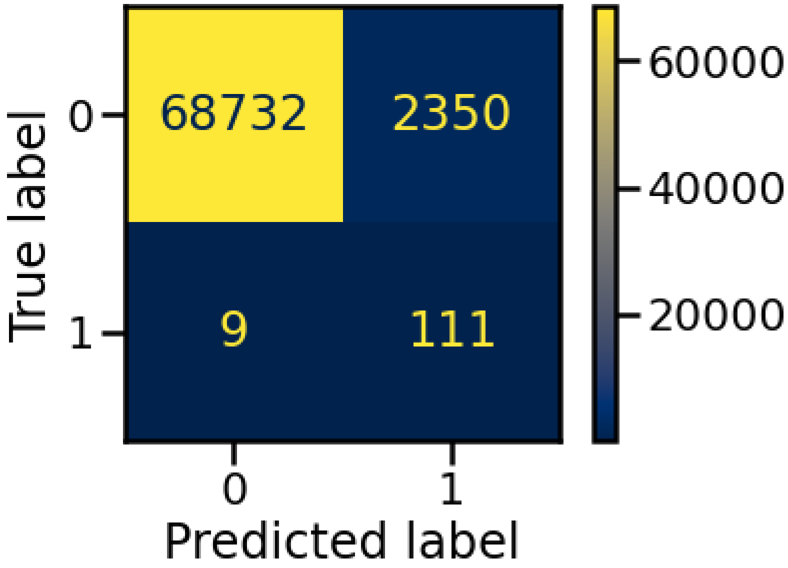
```
from sklearn.linear_model import LogisticRegression
lr_resampledUS = LogisticRegression()
lr_resampledUS.fit(X_train_resampledUS, y_train_resampledUS)
```

```
LogisticRegression()
```

```
y_pred_lrUS = lr_resampledUS.predict(X_test)
```

```
from sklearn.metrics import plot_confusion_matrix
sns.set_context("poster")
disp = plot_confusion_matrix(lr_resampledUS, X_test, y_test, cmap='cividis', colorbar=True)
```

```
     /usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: F
       warnings.warn(msg, category=FutureWarning)
```



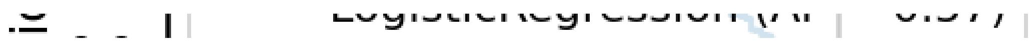
```
from sklearn.metrics import average_precision_score
y_score = lr_resampledUS.decision_function(X_test)
avg_precision = average_precision_score(y_test, y_score)
print('Average precision-recall score:', avg_precision)

     Average precision-recall score: 0.5965069727327362
```

```
from sklearn.metrics import plot_precision_recall_curve
disp = plot_precision_recall_curve(lr_resampledUS, X_test, y_test)
disp.ax_.set_title('Precision-Recall curve: ' 'AP={0:0.2f}'.format(avg_precision))
```

```
     /usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: F
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
acc = accuracy_score(y_test, y_pred_lrUS)
prec = precision_score(y_test, y_pred_lrUS)
rec = recall_score(y_test, y_pred_lrUS)
f1 = f1_score(y_test, y_pred_lrUS)
print('accuracy:', acc)
print('precision:', prec)
print('recall:', rec)
print('f1_score:', f1)
```

```
     accuracy: 0.9668689081767367
     precision: 0.04510361641609102
     recall: 0.925
     f1_score: 0.08601317318868655
```



```
model_results = pd.DataFrame([['UnderSampledLogisticsRegression', rec, prec, f1]],
               columns = ['Model', 'Recall', 'Precision', 'F1 Score'])
results = results.append(model_results, ignore_index = True)
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(criterion = 'entropy', random_state = 0)
rfc.fit(X_train, y_train)
```

```
     RandomForestClassifier(criterion='entropy', random_state=0)
```

```
y_pred_rfc = rfc.predict(X_test)
```

```
from sklearn.metrics import plot_confusion_matrix
sns.set_context("poster")
disp = plot_confusion_matrix(rfc, X_test, y_test, cmap='cividis', colorbar=True)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: F
  warnings.warn(msg, category=FutureWarning)
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
acc = accuracy_score(y_test, y_pred_rfc)
prec = precision_score(y_test, y_pred_rfc)
rec = recall_score(y_test, y_pred_rfc)
f1 = f1_score(y_test, y_pred_rfc)
print('accuracy:', acc)
print('precision:', prec)
print('recall:', rec)
print('f1_score:', f1)
```

```
accuracy: 0.9995505744220669
precision: 0.94
recall: 0.7833333333333333
f1_score: 0.8545454545454546
```

```
model_results = pd.DataFrame([['RandomForest', rec, prec, f1]],
               columns = ['Model', 'Recall', 'Precision', 'F1 Score'])
results = results.append(model_results, ignore_index = True)
```

Random Forest: Oversampling

```
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=0)
X_train_resampledOS, y_train_resampledOS = ros.fit_resample(X_train, y_train)
```

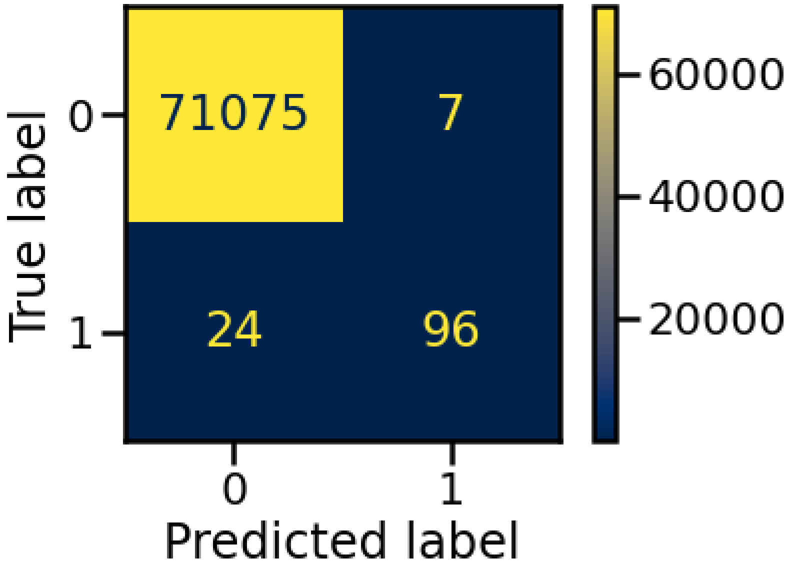
```
from sklearn.ensemble import RandomForestClassifier
rfc_resampledOS = RandomForestClassifier(criterion = 'entropy', random_state = 0)
rfc_resampledOS.fit(X_train_resampledOS, y_train_resampledOS)
```

```
RandomForestClassifier(criterion='entropy', random_state=0)
```

```
y_pred_rfcOS = rfc_resampledOS.predict(X_test)
```

```
from sklearn.metrics import plot_confusion_matrix
sns.set_context("poster")
disp = plot_confusion_matrix(rfc_resampledOS, X_test, y_test, cmap='cividis', colorbar=True)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: F
  warnings.warn(msg, category=FutureWarning)
```



```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
acc = accuracy_score(y_test, y_pred_rfcOS)
prec = precision_score(y_test, y_pred_rfcOS)
rec = recall_score(y_test, y_pred_rfcOS)
f1 = f1_score(y_test, y_pred_rfcOS)
print('accuracy:', acc)
print('precision:', prec)
print('recall:', rec)
print('f1_score:', f1)
```

```
accuracy: 0.9995646189713772
precision: 0.9320388349514563
recall: 0.8
f1_score: 0.8609865470852018
```

```
model_results = pd.DataFrame([['OverSampledRandomForest', rec, prec, f1]],
               columns = ['Model', 'Recall', 'Precision', 'F1 Score'])
results = results.append(model_results, ignore_index = True)
```

Random Forst (Undersampling)

```
from imblearn.under_sampling import RandomUnderSampler
rus = RandomUnderSampler(random_state=0)
X_train_resampledUS, y_train_resampledUS = rus.fit_resample(X_train, y_train)
```

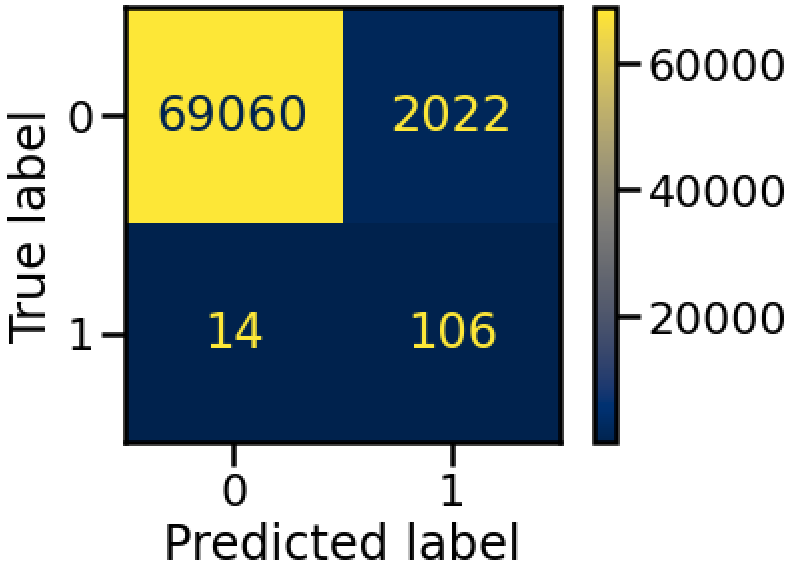
```
from sklearn.ensemble import RandomForestClassifier
rfc_resampledUS = RandomForestClassifier(criterion = 'entropy', random_state=0)
rfc_resampledUS.fit(X_train_resampledUS, y_train_resampledUS)
```

```
RandomForestClassifier(criterion='entropy', random_state=0)
```

```
y_pred_rfcUS = rfc_resampledUS.predict(X_test)
```

```
from sklearn.metrics import plot_confusion_matrix
sns.set_context("poster")
disp = plot_confusion_matrix(rfc_resampledUS, X_test, y_test, cmap='cividis', colorbar=True)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: F
  warnings.warn(msg, category=FutureWarning)
```



```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
acc = accuracy_score(y_test, y_pred_rfcUS)
prec = precision_score(y_test, y_pred_rfcUS)
rec = recall_score(y_test, y_pred_rfcUS)
f1 = f1_score(y_test, y_pred_rfcUS)
print('accuracy:', acc)
print('precision:', prec)
print('recall:', rec)
print('f1_score:', f1)
```

```
accuracy: 0.9714052976039999
precision: 0.04981203007518797
recall: 0.8833333333333333
f1_score: 0.09430604982206404
```

```
model_results = pd.DataFrame([['UnderSampledRandomForest', rec, prec, f1]],
               columns = ['Model', 'Recall', 'Precision', 'F1 Score'])
results = results.append(model_results, ignore_index = True)
```

Nueral Network

```
from keras.models import Sequential
ann = Sequential()
```

```
ann.add(Dense(units=16, kernel_initializer='uniform', activation='relu', input_dim=30))
ann.add(Dense(units=16, kernel_initializer='uniform', activation='relu'))
ann.add(Dense(units=16, kernel_initializer='uniform', activation='relu'))
ann.add(Dense(units=16, kernel_initializer='uniform', activation='relu'))
ann.add(Dense(units=1, kernel_initializer='uniform', activation='sigmoid'))
```

```
ann.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

```
ann.fit(X_train, y_train, batch_size = 10, epochs = 5)
```

```
Epoch 1/5
21361/21361 [==============================] - 39s 2ms/step - loss: 0.0071 - accuracy: 0
Epoch 2/5
21361/21361 [==============================] - 37s 2ms/step - loss: 0.0036 - accuracy: 0
Epoch 3/5
21361/21361 [==============================] - 37s 2ms/step - loss: 0.0034 - accuracy: 0
Epoch 4/5
21361/21361 [==============================] - 37s 2ms/step - loss: 0.0032 - accuracy: 0
Epoch 5/5
21361/21361 [==============================] - 37s 2ms/step - loss: 0.0030 - accuracy: 0
<keras.callbacks.History at 0x7f361420ead0>
```

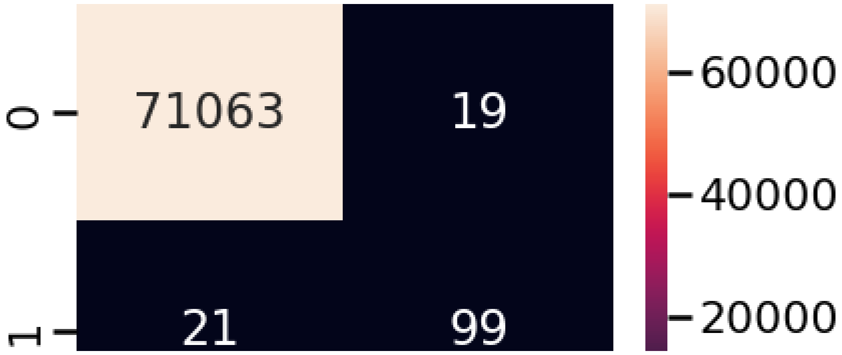
```
ann.evaluate(X_test, y_test)
```

```
2226/2226 [==============================] - 3s 1ms/step - loss: 0.0033 - accuracy: 0.9
[0.003334356937557459, 0.9994382262229919]
```

```
y_pred_nn = ann.predict(X_test)
y_pred_nn = (y_pred_nn > 0.5)
```

```
from sklearn.metrics import confusion_matrix
sns.set_context("poster")
cm = confusion_matrix(y_test, y_pred_nn)
sns.heatmap(cm, annot=True, fmt='g')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3614234f90>
```



```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
acc = accuracy_score(y_test, y_pred_nn)
prec = precision_score(y_test, y_pred_nn)
rec = recall_score(y_test, y_pred_nn)
f1 = f1_score(y_test, y_pred_nn)
print('accuracy:', acc)
print('precision:', prec)
print('recall:', rec)
print('f1_score:', f1)
```

```
accuracy: 0.9994382180275835
precision: 0.8389830508474576
recall: 0.825
f1_score: 0.8319327731092436
```

```
model_results = pd.DataFrame([['NeuralNetwork', rec, prec, f1]],
               columns = ['Model', 'Recall', 'Precision', 'F1 Score'])
results = results.append(model_results, ignore_index = True)
```

```
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=0)
X_train_resampledOS, y_train_resampledOS = ros.fit_resample(X_train, y_train)
```

```
from keras.models import Sequential
ann = Sequential()
```

```
from keras.layers import Dense
ann.add(Dense(units=16, kernel_initializer='uniform', activation='relu', input_dim=30))
ann.add(Dense(units=16, kernel_initializer='uniform', activation='relu'))
ann.add(Dense(units=16, kernel_initializer='uniform', activation='relu'))
ann.add(Dense(units=16, kernel_initializer='uniform', activation='relu'))
ann.add(Dense(units=1, kernel_initializer='uniform', activation='sigmoid'))
```

```
ann.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

```
ann.fit(X_train_resampledOS, y_train_resampledOS, batch_size = 10, epochs = 5)
ann.fit(X_train_resampledOS, y_train_resampledOS, batch_size = 10, epochs = 5)
```

```
Epoch 1/5
42647/42647 [==============================] - 116s 3ms/step - loss: 0.0268 - accuracy:
Epoch 2/5
42647/42647 [==============================] - 82s 2ms/step - loss: 0.0092 - accuracy: 0
Epoch 3/5
42647/42647 [==============================] - 73s 2ms/step - loss: 0.0068 - accuracy: 0
Epoch 4/5
42647/42647 [==============================] - 73s 2ms/step - loss: 0.0055 - accuracy: 0
Epoch 5/5
42647/42647 [==============================] - 73s 2ms/step - loss: 0.0050 - accuracy: 0
<keras.callbacks.History at 0x7f36826a2510>
```

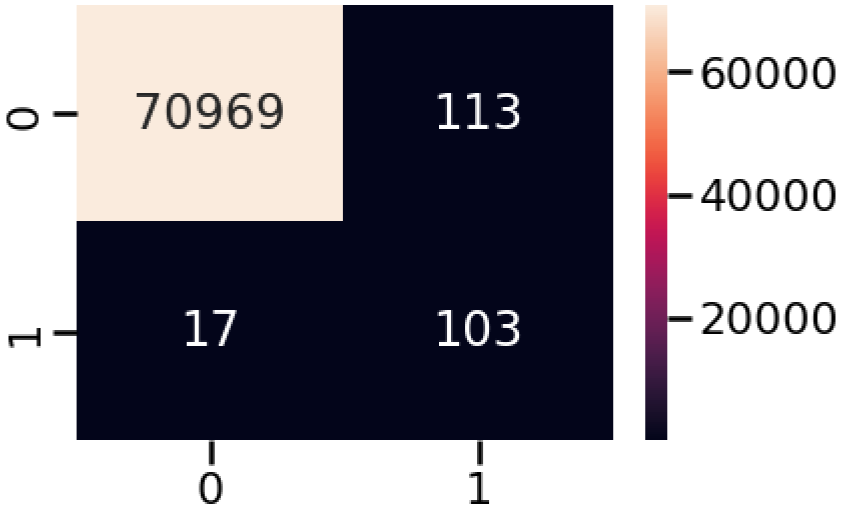
```
ann.evaluate(X_test, y_test)
```

```
2226/2226 [==============================] - 3s 1ms/step - loss: 0.0122 - accuracy: 0.99
[0.012222286313772202, 0.9981741905212402]
```

```
y_pred_nnOS = ann.predict(X_test)
y_pred_nnOS = (y_pred_nnOS > 0.5)
```

```
from sklearn.metrics import confusion_matrix
sns.set_context("poster")
cm = confusion_matrix(y_test, y_pred_nnOS)
sns.heatmap(cm, annot=True, fmt='g')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3614708910>
```



```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
acc = accuracy_score(y_test, y_pred_nnOS)
prec = precision_score(y_test, y_pred_nnOS)
rec = recall_score(y_test, y_pred_nnOS)
```

```
f1 = f1_score(y_test, y_pred_nnOS)
print('accuracy:', acc)
print('precision:', prec)
print('recall:', rec)
print('f1_score:', f1)
```

```
accuracy: 0.9981742085896463
precision: 0.47685185185185186
recall: 0.8583333333333333
f1_score: 0.6130952380952381
```

```
model_results = pd.DataFrame([['OverSampledNeuralNetwork', rec, prec, f1]],
               columns = ['Model', 'Recall', 'Precision', 'F1 Score'])
results = results.append(model_results, ignore_index = True)
```

Nueral Network: Under sampling

```
from imblearn.under_sampling import RandomUnderSampler
rus = RandomUnderSampler(random_state=0)
X_train_resampledUS, y_train_resampledUS = rus.fit_resample(X_train, y_train)
```

```
from keras.models import Sequential
ann = Sequential()
```

```
from keras.layers import Dense
ann.add(Dense(units=16, kernel_initializer='uniform', activation='relu', input_dim=30))
ann.add(Dense(units=16, kernel_initializer='uniform', activation='relu'))
ann.add(Dense(units=16, kernel_initializer='uniform', activation='relu'))
ann.add(Dense(units=16, kernel_initializer='uniform', activation='relu'))
ann.add(Dense(units=1, kernel_initializer='uniform', activation='sigmoid'))
```

```
ann.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

```
ann.fit(X_train_resampledUS, y_train_resampledUS, batch_size = 10, epochs = 5)
```

```
Epoch 1/5
75/75 [==============================] - 1s 2ms/step - loss: 0.6357 - accuracy: 0.7285
Epoch 2/5
75/75 [==============================] - 0s 2ms/step - loss: 0.3920 - accuracy: 0.9274
Epoch 3/5
75/75 [==============================] - 0s 2ms/step - loss: 0.2915 - accuracy: 0.9435
Epoch 4/5
75/75 [==============================] - 0s 2ms/step - loss: 0.1755 - accuracy: 0.9476
Epoch 5/5
```

```
    75/75 [==============================] - 0s 2ms/step - loss: 0.1441 - accuracy: 0.9543
    <keras.callbacks.History at 0x7f361469bf50>
```

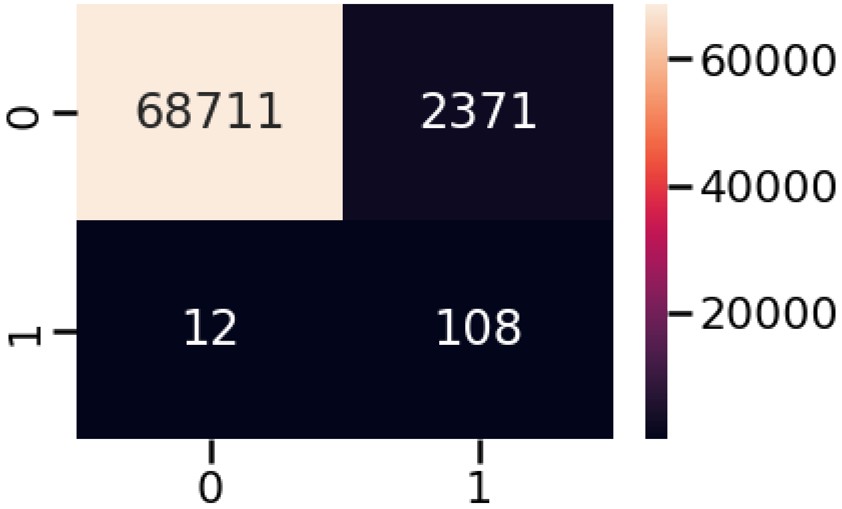
```
ann.evaluate(X_test, y_test)
```

```
    2226/2226 [==============================] - 3s 1ms/step - loss: 0.1360 - accuracy: 0.96
    [0.13596655428409576, 0.9665318131446838]
```

```
y_pred_nnUS = ann.predict(X_test)
y_pred_nnUS = (y_pred_nnUS > 0.5)
```

```
from sklearn.metrics import confusion_matrix
sns.set_context("poster")
cm = confusion_matrix(y_test, y_pred_nnUS)
sns.heatmap(cm, annot=True, fmt='g')
```

```
    <matplotlib.axes._subplots.AxesSubplot at 0x7f3614519990>
```



```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
acc = accuracy_score(y_test, y_pred_nnUS)
prec = precision_score(y_test, y_pred_nnUS)
rec = recall_score(y_test, y_pred_nnUS)
f1 = f1_score(y_test, y_pred_nnUS)
print('accuracy:', acc)
print('precision:', prec)
print('recall:', rec)
print('f1_score:', f1)
```

```
    accuracy: 0.9665318389932867
    precision: 0.04356595401371521
    recall: 0.9
    f1_score: 0.08310888803385919
```

```
model_results = pd.DataFrame([['UnderSampledNeuralNetwork', rec, prec, f1]],
               columns = ['Model', 'Recall', 'Precision', 'F1 Score'])
results = results.append(model_results, ignore_index = True)
results
```

	Model	Recall	Precision	F1 Score
0	LogisticsRegression	0.658333	0.877778	0.752381
1	OverSampledLogisticsRegression	0.908333	0.065152	0.121584
2	UnderSampledLogisticsRegression	0.925000	0.045104	0.086013
3	RandomForest	0.783333	0.940000	0.854545
4	NeuralNetwork	0.783333	0.940000	0.854545
5	OverSampledNeuralNetwork	0.858333	0.476852	0.613095
6	UnderSampledNeuralNetwork	0.900000	0.043566	0.083109

✓ 0s completed at 3:28 AM