ISSN 2616-5775 Vol. 5, Issue 13: 55-61, DOI: 10.25236/AJCIS.2022.051309

Credit Card Fraud Detection Based on Random Forest Model

Peilin Li

College of Art and Science, The Ohio State University, Columbus, Ohio, 43210, United State

Abstract: *This paper uses a classifier named random forest to detect credit card fraud. Credit card fraud is one of the main issues in the economic industry. To construct a credit card fraud detection system, a certain amount of samples is required. In this paper, a dataset containing 284,807 credit card transactions is used. This dataset has gone through the PCA transformation and includes 492 frauds out of 284,807 transactions. Based on the huge amount of data and imbalanced samples, this paper compresses the dataset and uses the synthetic minority over-sampling technique (SMOTE) to address the problem of imbalanced samples. Also, in this paper, we use random forest as a classification model while constructing the fraud detection system/method.*

Keywords: *Credit Card Fraud, Random Forest, SMOTE, Prediction Model, Data Science*

1. Introduction

Credit card fraud has been a common problem for years. The popularity of online payment and other card-not-present payment methods provides people with much more convenience. However, it also leads to various kinds of credit card fraud [1][4]. Therefore, the number of credit card fraud has grown rapidly with the rise of online payment over the past few years. An industry research paper estimates that merchants will lose around $130 billion to fraud between 2018 and 2023 [5]. Other than that, the fraudulent activity will also affect the cardholders and the acquirers/issuers (banks). For cardholders, their cards are stolen, which means, at the same time, their information is also leaked. And information leakage may cause more potential cybercrimes such as identity theft and hacking [7]. However, compared to cardholders, banks seem to be the ones who bear the cost of the fraud. As long as the cardholder reports a suspicious charge and the fraud has been confirmed, the bank is responsible to issue a chargeback to the cardholder. The cost of the issued chargeback and the cost of preventing credit card fraud become the main cost that the bank has to incur [6]. In this case, it is necessary to eliminate credit card fraud. One of the most effective ways to prevent credit card fraud would be credit card fraud detection. As long as the fraud is detectable, we are able to stop it from the very beginning.

2. Sample Description

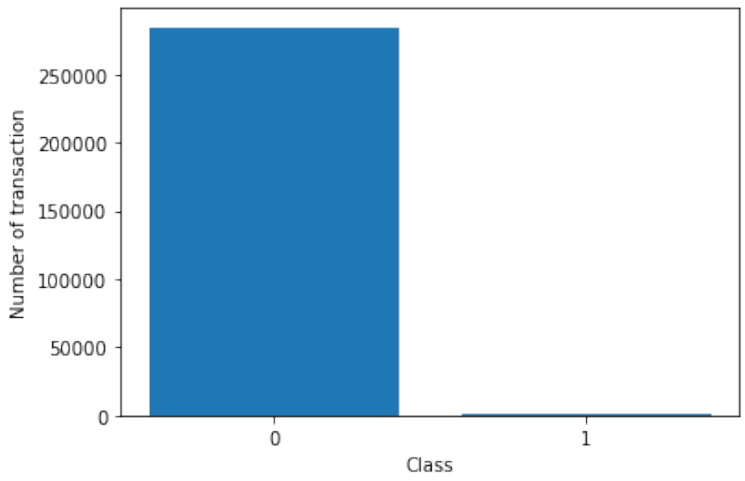

Figure 1: Number of transactions for class 0 and class 1

ISSN 2616-5775 Vol. 5, Issue 13: 55-61, DOI: 10.25236/AJCIS.2022.051309

To construct a fraud detection method, a large amount of data will be needed. In this paper, we would adopt a set of data that contains credit card transactions made by European cardholders as an example. This dataset includes 284,807 credit card transactions and 492 transactions were considered credit card fraud. There is a huge difference between the number of normal transactions and fraudulent transactions, which means the dataset is highly imbalanced (shown in Fig. 1). Each transaction contains 30 numerical features and 1 binary feature. And each feature has gone through the PCA transformation except for the feature "Time", which refers to the second elapsed between each transaction, and the feature "Amount", which refers to the transaction amount. To differentiate the normal transaction and fraudulent transaction, we would take value 1 in case of fraud and 0 otherwise [2].

To check the independence of each data, we arbitrarily choose two feature names V2 and V3 from the database and plot the distribution for each feature (shown as Fig. 2, Fig. 3.).

Both feature's distributions are nearly normal distributions. That also indicates that the PCA transformation has reduced the dimensions of data and made variables independent.

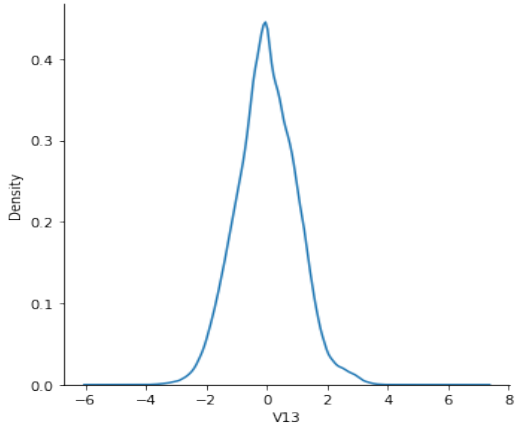

Figure 2: The distribution plot for vector 13

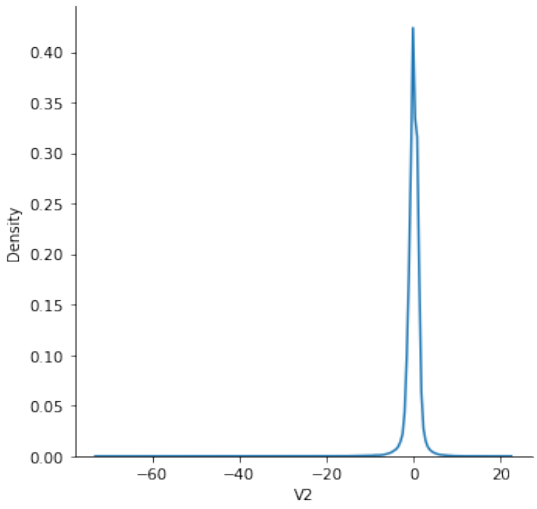

Figure 3: The distribution plot for vector 2

Before we train the model, we first need to split the data into two segments: the training set and the testing set. In this paper, we would use the train_test_split() function in the sklearn.model_selection library in Python [10]. We set the test size as 0.3, which means we randomly select 70% of the dataset as a training set and the rest of the dataset (30% of the dataset) as a testing set. Therefore, we have a training set containing 199346 samples and a testing set containing 85443 samples. In the training set, there are 199019 negative samples and 345 positive samples, which means the training set is highly imbalanced.

ISSN 2616-5775 Vol. 5, Issue 13: 55-61, DOI: 10.25236/AJCIS.2022.051309

2.1 SMOTE

It is not appropriate to directly use a highly imbalanced dataset as the input of our model, because an imbalanced dataset will lead to many unexpected behaviors. For example, it is highly possible to misclassify positive samples (minority class) as negative samples (majority class) due to the huge difference between the number of these two classes [8]. To deal with the problem of imbalanced data, in this paper, we applied the synthetic minority over-sampling technique (SMOTE) to the dataset.

There are two approaches to deal with the problems of class imbalance. One is to assign different costs to the training samples, the other one is to resample the whole dataset: oversample the minority class and/or sample the majority class. SMOTE is an approach of oversampling the minority class by generating synthetic samples [8]. To generate synthetic samples, first, neighbors from the k nearest neighbors would be randomly chosen based on the needed number of synthetic samples and one sample would be generated in the direction of each of the chosen neighbors. Then calculate the difference between the feature vector(sample) under consideration and its neighbor. Multiply the difference by a random number between 0 and 1 and add it to the feature vector under consideration. In this case, SMOTE can improve the accuracy of the classifier [8]. Other than this, it has also been proved that the combination of SMOTE and under-sampling performs better than plain under-sampling [8].

In this paper, we also apply SMOTE to address the problem of the imbalanced dataset. As the original dataset has been divided into a training set and a validation set, we would apply SMOTE to the training set before the training process starts. As shown in Fig. 4, there are 199019 normal samples and 345 abnormal samples in the training set. Then we invoke the SMOTE() method from the imbalanced-learn library, which is an implementation of SMOTE in Python[9] As synthetic samples have been created, now we have the 199019 abnormal data as well(as shown in Fig. 5). And this training set is ready to go through the training process at this point.

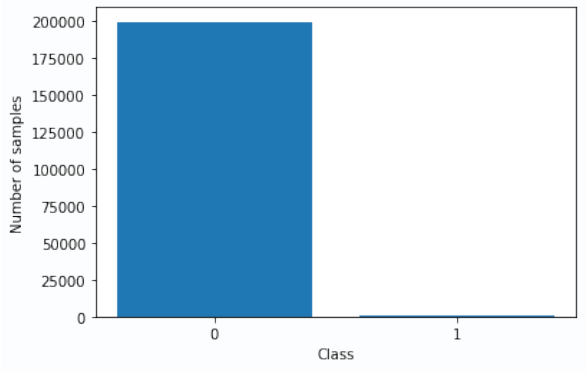

Figure 4: Number of transactions for class 0 and class 1 before SMOTE

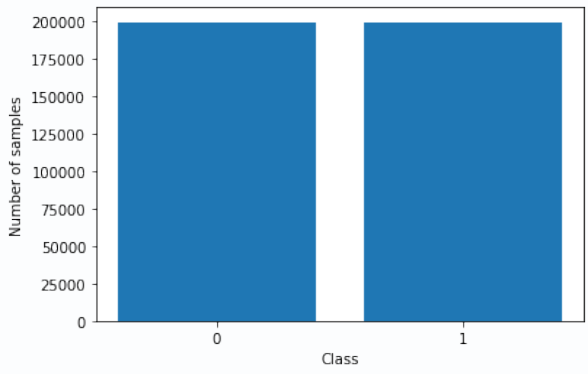

Figure 5: Number of transactions for class 0 and class 1 after SMOTE

3. Model Construction

In this paper, we used a random forest classifier for the model training process.

ISSN 2616-5775 Vol. 5, Issue 13: 55-61, DOI: 10.25236/AJCIS.2022.051309

3.1 Decision Tree & Random Forest

The main concept of the random forest is built based on the concept of the decision tree classifier. Decision tree classifier (DTC) is an approach to multi-stage decision-making [11]. Basically, the main idea of DTC is to partition a set of entities into smaller classes based on the selected partition rule [14].

A decision tree consists of a set of nodes and a set of edges. Nodes are subdivided into internal nodes, leaf, and root: a node that doesn't have a proper descendant is called a leaf (or a terminal), which contains a class label, a node that has no edge enter is called the root of the tree, and all the other nodes are internal nodes. Each internal node represents a "supporting attribute" that would be used for node partition. And each node represents the value of the supporting attribute used as criteria to classify objects. To calculate an appropriate value for each supporting attribute during classification, we would also need to choose criteria such as Gini impurity or entropy which could be used to measure the quality of a split. In this paper, we would use the entropy introduced by Shannon [12] as the criterion. The formulas of entropy and information gain are shown as:

$$E = -\sum_{i=1}^{n} p_i \log_2 p_i \quad (1)$$

$$I = E_{\text{parent}} - \sum_{i=1}^{c} \frac{|Vi|}{|data|} E_i \quad (2)$$

where E is the entropy, I is the information gain, pi is the fraction of class i objects at the given node, Vi is the possible number of class i objects, and data is the total number of samples. Entropy is a measure of the uncertainty of the classification result, and the value of the information gain represent the difference of the entropy between the lower level and the upper level [13][14]. For each level/attribute, we would choose the value with the lowest entropy, i.e., the largest information gain, as the criteria to split the dataset into two subsets. The process of drawing the decision tree is iterative. It would first choose the supporting attribute which has the largest information gain as the root node and then split the dataset into two subsets based on the value of the attribute. Then repeat the previous steps until there is no other available supporting attribute or the entropy is null [15].

Decision tree has proven its value in classification. However, there are still some possible drawbacks to the decision tree. For example, we can't simultaneously optimize both accuracy and efficiency [11]. Random Forest was invented aiming at improving classification accuracy. Random forest is an ensemble classifier that produces multiple decision trees and concludes the classification result based on the results of the decision trees [16]. The procedure of random forest is: first, randomly select samples from the training set; then use the selected samples to generate a decision tree as mentioned above; repeat the first two steps N times so that multiple decision trees can be created, where N is the expected number of the decision trees. As the random forest classifier has been built, the most popular result/class among the decision trees would be considered the final result of the classifier [16]. In this paper, we would use the RandomForestClassifier() method from the scikit learn to generate a classifier name "classifier" [17]. Then we would set the number of trees in the forest as 1000, and the criterion as entropy. Then we would use the fit() method to build a forest of trees using the specific training set generated before [17]. At this point, we now have a random first containing 1000 independent decision trees using entropy as the criterion for node partition.

4. Model Evaluation

4.1 ROC

As the model has been constructed, we would test the model and measure the classifier performance. The receiver operator characteristics (ROC) graph is one of the most intuitive metrics for evaluating and comparing the algorithm and it has been used in machine learning since around 1989 [18]. ROC graphs are two-dimension graphs which plot the true positive (TP) rate on the y-axis and the false positive (FP) rate on the x-axis [18]. In this case, the results of the classifier can be divided into four classes: "if the instance is positive and it is classified as positive, it is counted as a true positive (TP); if it is classified as negative, it is counted as a false negative (FN). If the instance is negative and it is classified as negative, it is counted as a true negative (TN); if it is classified as positive, it is counted as a false positive (FP)." [19][21]. Therefore, while evaluating the performance of the classifier, we usually consider the true positives as benefits and the false negatives as costs. And the formula for TP rate and FP rate can be denoted as:[18]

ISSN 2616-5775 Vol. 5, Issue 13: 55-61, DOI: 10.25236/AJCIS.2022.051309

$$\text{TP rate} \approx \frac{\text{Positives correctly classified}}{\text{Total positives}} \quad (3)$$

$$\text{FP rate} \approx \frac{\text{Negatives incorrectly classified}}{\text{Total negatives}} \quad (4)$$

In general, a ROC graph depicts the relative tradeoff between the TP rate and the FP rate [18]. A result of a discrete classifier such as the decision tree classifier is in turn corresponding to one ROC point. As more results are generated and we connect all the ROC points, the ROC "curve" will be generated. Each ROC curve generated from a finite dataset is a step function, but it would approach a true curve as the number of samples increases. Therefore, based on the composition of the ROC graph, a "more northwest" ROC curve usually corresponds to a better classifier since it has a lower expected cost [18].

4.2 AUC

ROC graphs can be used for depicting the performance of a classifier, but it is still hard to tell the difference between the classifiers by comparing the ROC curve. In this case, a scalar metric would be a better choice. Therefore, instead of comparing the ROC curve, we would compare the area under the ROC curve (AUC) [20]. AUC is the portion of the area of the ROC space. Therefore, the value of AUC is always between 0 and 1 [18]. The value of AUC also represents "the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance." [18]. In general, a realistic classifier should have an AUC greater than 0.5. While evaluating the performance of a classifier or comparing two classifiers, a classifier with greater AUC would be considered as the classifier that has better performance.

In this paper, we would use ROC and AUC to test the performance of the model. We would first use the plot_roc_curve() method from scikit learn [19] to plot the ROC curve and Fig.6 shows the ROC curve for the dataset.

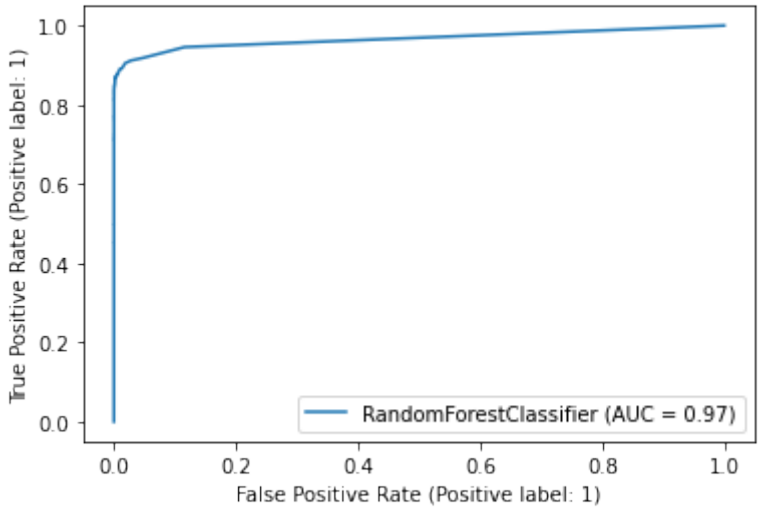

Figure 6: ROC curve of the model

As shown in Fig. 6, the ROC curve is close to the top-left corner and AUC is approximately 0.97, which is higher than 0.5 and close to 1. Therefore, based on the result of the ROC graph and the value of AUC. we can conclude that the classifier has a relatively good performance.

4.3 Precision, Recall, F-1 Score

Except for the ROC graph and AUC, we would also use other classifier performance metrics, including precision, recall, and F-1 score, to evaluate the classifier.

Precision, which is also known as the positive predictive value (PPV), and recall are common metrics for classifier evaluation [21]. The definitions of precision and recall are shown below:

$$\text{Precision} = \frac{\text{tp}}{\text{tp+fp}} \quad (5)$$

$$\text{Recall} = \frac{\text{tp}}{\text{tp+fn}} \quad (6)$$

ISSN 2616-5775 Vol. 5, Issue 13: 55-61, DOI: 10.25236/AJCIS.2022.051309

According to the definitions of precision and recall, we can see that there is a slight difference between precision and recall. Precision measures the proportion of the samples classified as positive that are truly positive but recall measures the proportion of the positive samples that are correctly labeled [21]. The greater the values of these two metrics, the better the classifier. In this paper, we would invoke the precision_score() method [22] and recall_score() method [23] from scikit learn to calculate the precision and recall of the classifier. And the precision score is approximately 0.95 and the recall score is approximately 0.76.

F-1 score(F-measure) is another classifier performance metric based on precision and recall, as shown below [21]:

$$F-measure = 2 \times \frac{Recall \times Precision}{Recall + Precision} \quad (7)$$

Based on the definition of the F-1 score, when the values of both recall and precision are 1, i.e., the classifier performs perfectly, the F-1 score will be 1 which is also the maximum value for the F-1 score. Therefore, the range of the F-1 score is between 0 and 1. The greater the F-1 score is, the better the classifier. In this paper, we would invoke the f1_score() method from scikit learn [24] and the returned f-1 score of the classifier is approximately 0.85.

5. Conclusions

Concerning the increasing cost of credit card fraud, it is our belief that a credit card fraud detection system can provide an effective way to prevent credit card fraud. Based on the result of the model evaluation, most of the classification evaluation metrics are relatively high. And that shows this model can effectively detect credit card fraud. Therefore, banks, as the one that bears most of the cost of credit card fraud, and merchants would benefit from the model since it could avoid credit card fraud by detecting the credit card fraud from the very beginning and warning the cardholders, and therefore, decrease the number of credit card fraud. But we still have a further way to go. This model still has some limitations. For example, since the dataset used for model training is limited, it is set to detect the credit card fraud that has been observed or defined. However, nowadays, with the rapid development of information technology, more and more new credit card frauds occur. Due to the limited training samples, the model may not be able to detect other new kinds of fraud as time pass. Other than this, this model could only detect credit card fraud when it already happened because the dataset only shows the features of transactions. But if we include more features such as the consumption habits of the customers, the model can detect what kinds of customers would be more likely to encounter credit card fraud. And therefore, we can even prevent credit card fraud before it happens. Data collection is a worldwide problem for prediction model constructions. Therefore, we would collect more data and more features about credit card fraud to improve the performance of this credit card fraud detection model if possible.

References

[1] Delamaire, Linda, Hussein Abdou, and John Pointon. "Credit card fraud and detection techniques: a review." Banks and Bank systems 4.2 (2009): 57-68.

[2] "Credit Card Fraud Detection." Google Search, Google, https://www.google. com.hk/search?q =citation%2Bgenerator%2BMLA&newwindow=1&ei=w1dGY6CZGrfN2roP9eyh0AY&ved=0ahUKE wjghqfFgdr6AhW3plYBHXV2CGoQ4dUDCA4&uact=5&oq=citation%2Bgenerator%2BMLA&gs_lp= Egdnd3Mtd2l6uAED-AEBMgUQABiABDIFEAAYgAQyBBAAGB4yBBAAGB4yBBAAGB4yBBAAGB4y BBAAGB4yBBAAGB4yBBAAGB7CAgoQABhHGNYEGLADwgIEEAAYQ5AGCkjfDVCDB FiPC3ABeAHIAQCQAQCYAYQEoAH5CaoBBzMtMS4xLjHiAwQgTRgB4gMEIEEYAOIDBCBGGACI BgE&sclient=gws-wiz.

[3] Browne, Michael W. "Cross-validation methods." Journal of mathematical psychology 44.1 (2000): 108-132.

[4] Barker, Katherine J., et al. "Credit Card Fraud: Awareness and Prevention." Journal of Financial Crime, vol. 15, no. 4, 2008, pp. 398-410. HeinOnline, https://heinonline-org.proxy.lib.ohio-state. edu/ HOL/P?h=hein.journals/jfc15&i=398.

[5] S. T. King, N. Scaife, P. Traynor, Z. Abi Din, C. Peeters and H. Venugopala, "Credit Card Fraud Is a Computer Security Problem," in IEEE Security & Privacy, vol. 19, no. 2, pp. 65-69, March-April 2021, doi: 10.1109/MSEC.2021.3050247.

[6] Bhatla, Tej Paul, Vikram Prabhu, and Amit Dua. "Understanding credit card frauds." Cards business review 1.6 (2003): 1-15.

ISSN 2616-5775 Vol. 5, Issue 13: 55-61, DOI: 10.25236/AJCIS.2022.051309

[7] Chevers, Delroy A. "The impact of cybercrime on e-banking: A proposed model." CONF-IRM. 2019.
[8] Chawla, Nitesh V., et al. "SMOTE: synthetic minority over-sampling technique." Journal of artificial intelligence research 16 (2002): 321-357.
[9] "Smote#." SMOTE - Version 0.9.1,https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html.
[10] "Sklearn.model_selection.train_test_split." Scikit, https://scikit-learn.org/stable/modules/generated/sklearn.model selection.train test split.html.
[11] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, no. 3, pp. 660-674, May-June 1991, doi: 10.1109/21.97458.
[12] Shannon, Claude Elwood. "A mathematical theory of communication." The Bell system technical journal 27.3 (1948): 379-423.
[13] Rényi, Alfréd. "On measures of entropy and information." Proceedings of the fourth Berkeley symposium on mathematical statistics and probability. Vol. 1. No. 547-561. 1961.
[14] Breiman, L. (1984). Classification And Regression Trees (1st ed.). Routledge. https://doi. org/ 10.1201/9781315139470
[15] Li, Xiang, and Christophe Claramunt. "A spatial entropy-based decision tree for classification of geographical information." Transactions in GIS 10.3 (2006): 451-467.
[16] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org /10.1023/ A: 1010933404324
[17] "Sklearn.ensemble.randomforestclassifier." Scikit, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.
[18] Fawcett, Tom. "An introduction to ROC analysis." Pattern recognition letters 27.8 (2006): 861-874.
[19] "Sklearn.metrics.plot_roc_curve." Scikit, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.plot_roc_curve.html.
[20] N. Seliya, T. M. Khoshgoftaar and J. Van Hulse, "A Study on the Relationships of Classifier Performance Metrics," 2009 21st IEEE International Conference on Tools with Artificial Intelligence, 2009, pp. 59-66, doi: 10.1109/ICTAI. 2009. 25.
[21] Davis, Jesse, and Mark Goadrich. "The relationship between Precision-Recall and ROC curves." Proceedings of the 23rd international conference on Machine learning. 2006.
[22] "Sklearn.metrics.precision_score." Scikit, https://scikit-learn.org/stable/modules/ generated/sklearn.metrics.precision_score.html.
[23] "Sklearn.metrics.recall_score." Scikit, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html.
[24] "Sklearn.metrics.f1_score." Scikit, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.