Available online: www.ijsrmes.com

International Journal for Scientific Research in Modern Engineering and Science

International Journal for Scientific Research in Modern Engineering and Science, 3(4):67- 77 (2022)

MACHINE LEARNING BASED DEVICE TYPE CLASSIFICATION FOR IOT DEVICE

Divya Shukla [1]*, Mohan Rao Mamdikar [2]

[12]PDepartment of Department of Computer Science & Engineering, Vishawavidyalaya Engineering College, Ambikapur

Abstract

The Internet of Things (IOT) refers to a network that uses the internet to connect various devices. IoT helps to transmit data between devices, track and monitor devices, and other things. IoT makes objects "smart" by allowing them to transmit data and automate tasks without physical interference. In this paper we've used and proposed the Machine Learning Classifier including Gaussian Naïve Bayes Classifier, Random Forest Classifier, K-Nearest Neighbours Classifier, Support Vector Machine Classifier, Gradient Boosting Classifier. In the first step, the data of 28 IoT devices were taken for model training and testing process by using several data pre-processors including data cleaning and splitting, standardising features, Numerical imputation, etc. In the concluding step, the accuracy score of the Random Forest Classifier was found to make the most accurate prediction with an accuracy score of 95.2% among the five classifiers.

Keywords: *Internet of Things, Machine Learning, Machine Classifier, RFC, KNN, SVM, GBC, GNB, Standardizing Features*

* *Corresponding author*

1. INTRODUCTION

The Internet of Things (IoT) stems from the idea of interconnecting most contemporary devices in the network. Many of these devices are wirelessly connected to facilitate the deployment process. Recently, there has been an explosion of embedding wireless capabilities in several devices, which is expected to reach 42 billion devices worldwide by 2025 [1]. Network connected devices include devices such as internet-enabled appliances, medical devices, smart locks, wearable's, home monitoring sensors, cameras, industrial sensors and actuators, and many more [2, 3, 4]. These devices collect a large amount of sensitive information about

the user's whereabouts, health, behavior, and environment [4]. They are also responsible to perform tasks that are safety-critical, such as safe flying of UAVs, automatically regulating

one's heart rate and delivering drugs, controlling entry to one's residence, controlling gas and electric appliances, etc. [4]. For example, a smart garage door provides access to the house premises, a remotely programmed pacemaker controls the electrical pulses applied to the heart [5, 4], and a smart insulin pumps continuously monitor and adjust insulin delivered to diabetic patients [6, 4] The security of these devices is prone to two significant vulnerabilities; first, the insecurities and vulnerabilities in the firmware and second, the secrets utilized to bootstrap security are prone to compromise. The Common Vulnerabilities and Exposures (CVE) database of vulnerabilities alone consists of over 900 records related to the keyword "IoT" depicting the vulnerabilities of firmware [7]. Furthermore, the compromised secrets can be utilized by an unauthorized party to inject/modify sensitive data [11, 12, 13, 10, 8]. Therefore, both vulnerabilities compromise the security of the whole network. One of the available ways to address this issue is to use a policy-based access control to prevent insecure devices from taking control of the home network [14]. However, the state-of-the-art requires manual configuration of the policies and is not capable of automatically distinguishing device capabilities to enforce the policy.

2. MACHINE LEARNING CLASSIFIER

2.1 Random Forest Classifier

The Random Forest Classifier is based on a decision tree like structure at its core, and it can be categorized as an ensemble-based learning method used for making classifications. The algorithm is called ensemble-based because it makes a prediction using an ensemble of large amounts of different and completely uncorrelated decision trees. The final result is based on the predictions made by each individual decision.

2.2 K-Nearest Neighbors Classifier

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

2.3 Gradient Boosting Classifier

The Gradient Boosting Classifier is a supervised machine learning algorithm based on the ensemble technique, which means that it utilizes the predictions made by several different weak decision trees to give a strong final prediction. The gradient boosting algorithm uses an additive approach to build the model by typically adding several decision trees sequentially, where in each iteration, the successor tree utilized the results generated by its predecessor tree to reduce error. The processes, also shown in Fig. 2.1, effectively reduces the error over several iterations, which helps the algorithm to provide its final predictions.

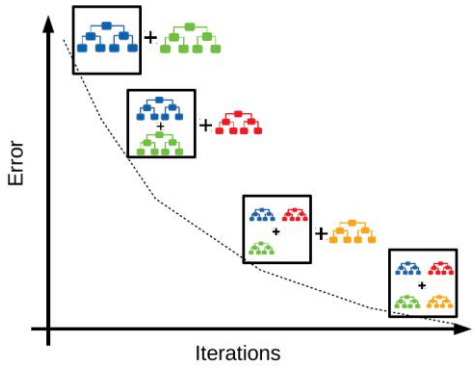


Fig 2.1 Gradient Boosting Classifier

2.4 Support Vector Machine Classifier

The Support Vector Machine Classifier is a supervised machine learning algorithm, mostly used for solving classification problems. The algorithm plots all data points with an ‘n’ number of features in an n-dimensional space, and the coordinate value of each data point is the value of the feature. Finally, classification is performed by finding hyper planes that differentiates the multiple classes, and if a test data point can be placed within a certain hyperplane, it will share the same class with the data points in its neighborhood. Fig. 2.2.

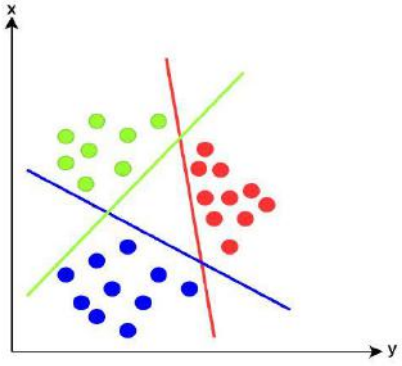


Fig. 2.2 Support Vector Machine Classifier

2.5 Gaussian Naive Bayes Classifier

The Gaussian Naive Bayes Classifier is often used for classification jobs where the values of all features are continuous and distributed in a Gaussian distribution. The algorithm is called naive because it implies that the presence of any feature is completely independent of the existence of any other feature. It is based on the Bayes

theorem (Eq .1) which helps define the probability of the occurrence of hypothesis A after the data B, is already given.

$$P(A|B) = P(B|A)P(A)/P(B) \quad \text{(Eq .1)}$$

3. METHODOLOGY

3.1 Data Set

The data collected include more than 28 IoT devices such as cameras, motion sensors, health monitors, appliances, etc. A subset of data collected over the period of six months has been made available as open-source. We chose 15 devices as shown in Table 3.1 and obtained relevant information from packet capture files by extracting important features using tshark into comma separated value files (.csv). After capturing the "n" packets from the pcap file and the "f" features using tshark, we built the fingerprint matrix (n×f). We further classified the different devices according to the type of device they are and assigned them a class for the model training and testing process. Traffic between all devices listed in the table and the access point was acquired from data collected at the University of New South Wales [15].

Table 3.1: List of IoT devices and their classes

Device	**Name Category**	**Class**
Amazon	Echo Smart Home Assistant	1
Netatmo Welcome	Smart Camera	2
Samsung SmartCam	Smart Camera	2
Dropcam	Smart Camera	2
Insteon Camera	Smart Camera	2
Belkin Wemo switch	Smart Electrical 3 and Lighting	3
Light Bulbs LiFX Smart Bulb	Smart Electrical 3 and Lighting	3
Belkin wemo motion sensor	Smart Sensor	4
Netatmo weather station	Smart Sensor	4
Withings Smart scale	Smart Sensor	4
Withings Aura smart sleep sensor	Smart Sensor	4
Triby Speaker	Smart Sensor	4
Samsung Galaxy Tab	Non-Iot	0
Laptop	Non-Iot	0
iPhone	Non-Iot	0

3.2 Data Pre-Processing

For better classification results the data was preprocessed by using several techniques. Following are the techniques used in this paper.

3.3 Data Cleaning and Splitting

The pcap files had several data points and characteristics that were not relevant. Therefore, once the pcap files were converted to csv files, we removed packets with source ethernet addresses that were not required in our model. This included all outgoing traffic from the access point, since it does not require classification and would only bias the predictions made by the classifier due to the large amounts of such packets present in the data. The empty columns representing empty values for features were also removed since they do not provide any contribution to the classification process. Finally, the entire dataset was labeled in different classes for training and testing. The data in the csv was loaded into a data frame and the labels were separated out of the data frame splitting the data into X (features) and Y (labels). Then both X and Y were randomly split into train and test data sets in a ratio of 80% and 20% respectively.

3.4 Standardizing Features

Standardizing features is a requirement for many classifiers to achieve high accuracy. We used the standard sklearn scale method to standardize the features in our dataset, which subtracts the mean from the values and then scales it to unit variance,

$z = v - m/s$ (Eq .2)

where, v = values of the sample dataset, m = mean of training samples and s = standard deviation of the training samples

3.5 Numerical Imputation

We use numerical imputation to assign a value to missing values in any feature. It is better than removing the entire packet since that would affect the amount of data needed to make accurate classifications. Therefore, missing values are imputed to the median values of each individual feature. This process is done by utilizing the fill na method provided by the pandas data analysis tool. First, we use a random forest search classifier to extract feature importance scores from the 19 features shown in Fig.3.1 After calculating the importance scores for the features, we set a threshold of 0.05 for the importance score of the features and eliminated all features with importance scores below the threshold.

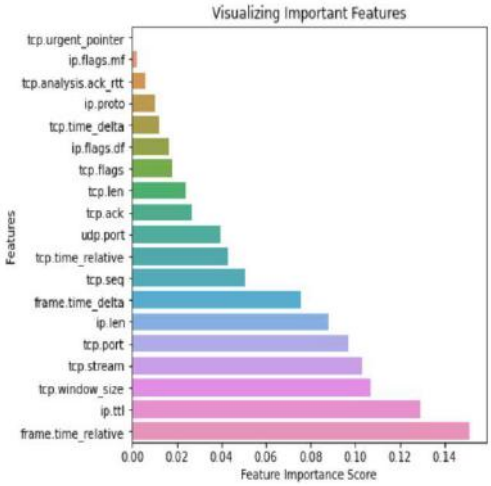


Fig. 3.1 Feature importance score

The data that we used to classify the type of IoT devices had several features that were not useful for making predictions. Such features included tcp.urgent pointer, ip.flags.mf, tcp.analysis.ack rtt, ip.proto, tcp.time delta, ip.flags.df, tcp.flags, tcp.len, tcp.ack, udp.port, tcp.time relative, tcp.seq. These features were removed from the dataset because they did not provide any valuable contribution, negatively affected the required runtime and memory usage, and reduced the accuracy of the classifiers used.

3.6 Training and Testing

After collecting and pre-processing the dataset, 80% of it was used to train each of these five classifiers individually: Random Forest Classifier (RFC)[23], K-Nearest Neighbors Classifier (KNN)[17], Support Vector Machine Classifier (SVM)[18], Gradient Boosting Classifier (GBC)[19], and Gaussian Naive Bayes Classifier (GNB)[20]. The training process was performed using the fit method provided by sklearn, which fits the model onto the data to later provide predictions [21]. During the training process, the time taken to fully train each model was recorded and is shown in Table 3.2. During testing, the labels for the test data were predicted by the previously trained models, and the predicted labels were compared to the actual labels to calculate the accuracy of the classifications provided by each model. This was done by the use of accuracy score function provided by the sklearn metrics library [22]. Finally, the time required to train each model was calculated and the average time required to classify a single data point is shown in Table 3.2 Model.

Table 3.2: Time required for training each model and the average time required to classify one device.

Model	**Train Time**	**Test Time**
Gaussian Naive Bayes Classifier	50 ms	0.23ms
Random Forest Classifier	18.17 sec	0.63 ms
K-Nearest Neighbors Classifier	0.34 sec	0.92 ms
Support Vector Machine Classifier	110 min	3.67 ms
Gradient Boosting Classifier	2.9 min	6.89 ms

4. RESULTS ANALYSIS

We present the results of the classification algorithms employed in this thesis to identify the device type-level classification. To achieve the most accurate classification of all the IoT devices, we trained five different models including Random Forest Classifier (RFC), K-Nearest Neighbors (KNN), Support Vector Machine Classifier (SVM), Gradient Boost Classifier (GBC) and, Gaussian Naive Bayes (GNB) classifier and then evaluated their performance. The metrics used to analyze each model performance were the F1 score and the accuracy score.

F1 Score

The F1 score is an evaluation metric used to determine the performance of a machine learning classifier and is defined as the harmonic mean of recall and precision. It gives a better insight about the classification made by each device type classifier as it not only calculates the number of misclassifications made by the different models but helps identify the types of mis classifications made.

$$Precision = \frac{Number\ of\ True\ Positives}{\text{Number of True Positives + Number of False Positives}}$$

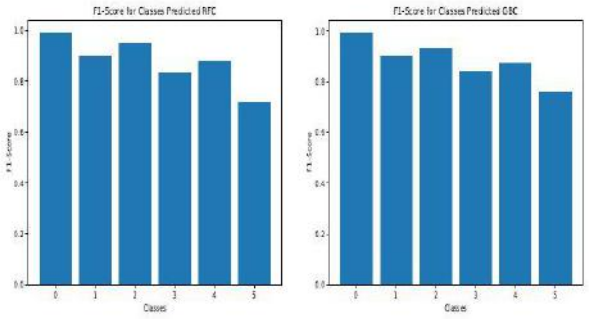
Recall, also known as sensitivity, is the ratio between the correct true positives and the total sum of the number of false negatives and true positives and is given by

$$Recall = \frac{Number\ of\ True\ Positives}{\text{Number of True Positives + Number of False Negatives}}$$

The value of the F1 score can range between 0 and 1 where 1 is the highest score a model can achieve and the values of both precision and recall are the highest. The formula for the F1 score is given by

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

We calculate the F1 score using the evaluation metric library provided by sklearn metrics The F1 score for each class within each model is plotted in the Fig. 4.1 where class 0 represent non-IoT devices, class 1 represents Home Assistants, class 2 represents Smart Camera, class 3 represents Smart Bulb & Smart Electrical, class 4 represents Smart Sensors, and class 5 represents Smart Speaker. The highest average F1 score for all classes was provided by the Random Forest Classifier (RFC) where it can further seen that the model is near perfect for differentiating between an IoT and a non-IoT device.



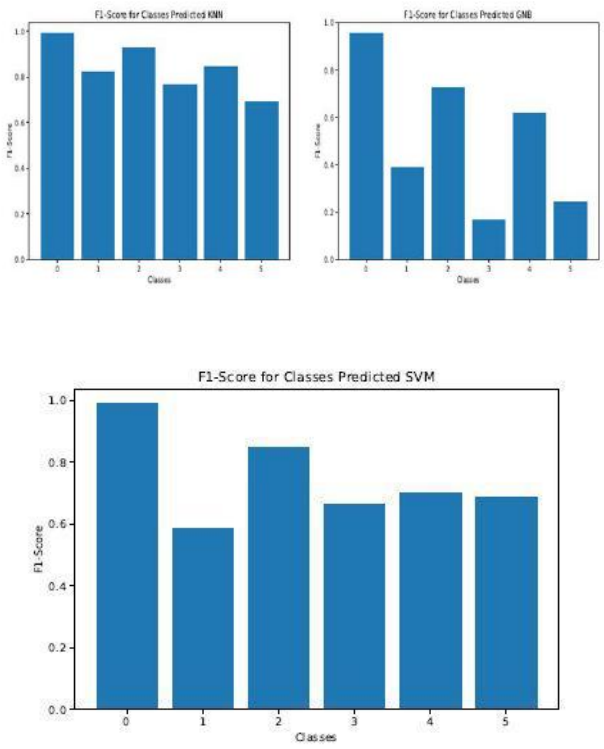


Fig 4.1: F1 scores for Random Forest Classifier (RFC), Gradient Boost Classifier (GBC), K-Nearest Neighbors Classifier (KNN), Support Vector Machine Classifier (SVM), and Gaussian Naive Bayes Classifier (GNB).

Accuracy Score

The accuracy score is an evaluation metric used for machine learning models to measure their performance by determining the ratio between the number of correct predict tions made by the classifier and the total number of predictions to be made Additionally, the percentage of this score can be calculated to obtain the accuracy of a classifier in terms of a percentage.

Accuracy Score = Number of correct predictions/Number of total predictions

The function to calculate the accuracy of our machine learning models used to perform multi-class classification was provided by the sklearn. metrics library [24]. After making the predictions, we established that the Random Forest Classifier (RFC) was the most accurate model for making predictions with an accuracy of 95.2%. The second most accurate classifier was the Gradient Boost Classifier (GBC) with an accuracy of 94.8%, then the K-Nearest Neighbors Classifier (KNN) with accuracy 93.3%, Support Vector Machine Classifier (SVM) with accuracy of 88.3%, and finally the Gaussian Naive Bayes Classifier (GNB) with accuracy of 76.8%.

Furthermore, using the accuracy score, we also determine the 95% confidence interval for each classifier. The method essentially provides an upper bound and alower bound for accuracy, which represents all the possible values accuracy can have. Therefore, when any device needs classification, there will be a 95% likelihood for it to be classified will lie between that range and is given by the following equation

$$95\%\ Confidence\ Interval = 1.96 \times \sqrt{\frac{(accuracy \times (1 - accuracy))}{n}}$$

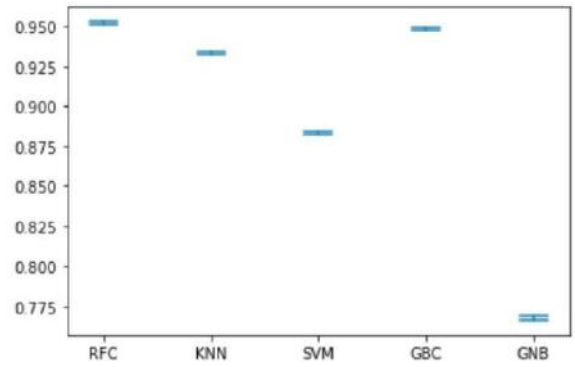


Fig 4.2 Accuracy vs Model plot for Confidence Interval

The upper and lower bounds of the 95% confidence interval for each classifier are shown in the Table4.1 and its plot is shown in Fig.4.2.

Table 4.1: Upper bound and lower bound of 95% Confidence Interval (CI)

Model	CI Upper Bound	CI Lower Bound
Random Forest Classifier	0.9527	0.9513
K-Nearest Neighbors Classifier	0.9338	0.9322
Support Vector Machine Classifier	0.8840	0.8820
Gradient Boosting Classifier	0.9487	0.9473
Gaussian Naive Bayes Classifier	0.7694	0.7666

5. CONCLUSION

We used the Random Forest Classifier, Gradient Boost Classifier, K-Nearest- Neighbors Classifier, Support Vector Machine Classifier, and the Naive Bayes Classifier to identify whether a device in the network is an IoT device or not, and then further classified the type of IoT device. Of all models, the Random Forest Classifier was able to make the most accurate prediction with an accuracy score of 95.2% and the highest average F1 score. Gaussian Naive Bayes Classifier was found to be the fastest classifier for training and testing; however, we prefer to use the Random Forest Classifier, as it still required less time than most other models for training and testing and provided the most accurate
results.

References

[1] PIDC. IoT growth demands rethink of long-term storage strategies, says idc, Jul 2020.

[2] T. Ahmad and D. Zhang. Using the internet of things in smart energy systemsand networks. Sustainable Cities and Society, page 102783, 2021.

[3] R. S. Lee. Smart city. In Artificial Intelligence in Daily Life, pages 321–345. Springer, 2020.

[4] N. Ghose, L. Lazos, and M. Li. In-band secret-free pairing for cots wirelessdevices. IEEE Transactions on Mobile Computing, 2020

[5] Z. Yi, F. Xie, Y. Tian, N. Li, X. Dong, Y. Ma, Y. Huang, Y. Hu, X. Xu, D. Qu, et al. A battery-and leadless heart-worn pacemaker strategy. Advanced Functional Materials, 30(25):2000477, 2020

[6] J. Kesavadev, B. Saboo, M. B. Krishna, and G. Krishnan. Evolution of insulin delivery devices: from syringes, pens, and pumps to DIY artificial pancreas. Diabetes Therapy, 11:1251–1269, 2020.

[7] CVE. IoT search results. https://cve.mitre.org/cgi-bin/cvekey.cgi? keyword={IoT}. [Online; accessed 20-Mar-2022].

[8] D. Freed, J. Palmer, D. Minchala, K. Levy, T. Ristenpart, and N. Dell. "A Stalker's Paradise": How Intimate Partner Abusers Exploit Technology. In Proc. of CHI Conference on Human Factors in Computing Systems, page 667. ACM, 2018.

[9] C. Meraki. Applying policies by device type. https://documentation.meraki.com/MR/Group_Policies_and_Block_Lists/Applying_Policies_by_Device_ Type, 2021. [Online; accessed 14-Feb-2021].

[10] S. Havron, D. Freed, R. Chatterjee, D. McCoy, N. Dell, and T. Ristenpart. Clinical computer security for victims of intimate partner violence. In Proc. Of USENIX Security Symposium, pages 105–122, 2019.

[11] T. Armerding. The 18 biggest data breaches of the 21stcentury.https://www.csoonline.com/article/2130877/ the-biggest-data-breaches-of-the-21st century.html, 2018. [Online; accessed 10-Feb-2020].

[12] M. Vanhoef and F. Piessens. Key reinstallation attacks: Forcing nonce reuse inWPA2. In Proc. of ACM Conference on Computer and Communications Security (CCS). ACM, 2017.

[13] R. Chatterjee, P. Doerfler, H. Orgad, S. Havron, J. Palmer, D. Freed, K. Levy, N. Dell, D. McCoy, and T. Ristenpart. The spyware used in intimate partner violence. In Proc. of IEEE Symposium on Security and Privacy (SP), pages 441–458, 2018.

[14] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. T. Polk, et al. Internet x. 509 public key infrastructure certificate and certificate revocation list (CRL) profile. RFC, 5280:1–151, 2008.

[15] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath,and V. Sivaraman. Classifying IoT devices in smart environments us ing network traffic characteristics. IEEE Transactions on Mobile Computing, 18(8):1745–1759, 2018

[16] Scikit-LeSklearn.ensemble.randomforestclassifier. https://scikitlearn.org/stable/modules/generated/sklearn.ensemble. RandomForestClassifier.html.[Online; accessed 22-Mar-2022].

[17] Scikit-Learn.Sklearn.ensemble.kneighborsclassifier. https://Scikitlearn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.[Online; accessed 22-Mar-2022].

[18] Scikit-Learn.Sklearn.ensemble.svm.svc. https://scikit-learn.org/stable/ modules/generated/sklearn.svm.SVC.html.[Online; accessed 22-Mar-2022].

[19] Sklearn.ensemble.gradientboostingclassifier.https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html. [Online; accessed 22-Mar-2022].

[20] Sklearn.ensemble.naivebayes.gaussiannb. https://scikitlearn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html. [Online; accessed 22-Mar-2022].

[21] Scikit-Learn. Developing scikit-learn estimators. https://scikit-learn.org/ stable/developers/develop.html. [Online; accessed 24-Mar-2022].

[22] Scikit-Learn. sklearn.metrics.accuracy score. https://scikit-learn.org/ stable/modules/generated/sklearn.metrics.accuracy_score.html. [Online; accessed 24-Mar-2022].

[23] S.-Y. Hwang and J.-N. Kim. A malware distribution simulator for the verification of network threat prevention tools. Sensors, 21(21):6983, 2021

[24] Scikit-Learn. sklearn.metrics.accuracy score. https://scikitstable/modules/generated/sklearn.metrics.accuracy_score.html. [Online;accessed 24-Mar-2022].