Quick test and classification of Li battery function

Anye Shi[1,2,3]

1Electrical and Computer Engineering Department, University of California, San Diego

2Department of Chemistry and Biochemistry, University of California, San Diego

3Nanoengineering Department, University of California, San Diego

anshi@ucsd.edu

Abstract*: *Understanding the "property-function" relationship is really crucial to rationally design and optimize fabrication procedure to obtain high-performance Li-battery devices. However, traditional cyclic voltammetry curve characterization (CVC) takes long time with really strict experimental requirement. Resistance is a promising substitution to evaluate the charge transportation performance, reflecting materials fabrication uniformity and charge separation ability for as-obtained devices. But no one proves the relationship between CVC and resistance, directly. Our idea is to use machine learning method to predict if there is certain relationship between CVC and resistance. And then we use resistance value to quick test and classify the function of our Li batteries.

INTRODUCTION

Due to ultrafast charge transfer rates, high electronic capacity and extra-long carrier diffusion length, Li batteries have been widely used in mobile electronic devices. Despite great progress in electronic performance in the last few years, there are still some problems that need to be settled in manufacturing process. One of the most important questions is to fabricate high performance battery with superb stability and durability. Thus, it is crucial to understand the "structural-functional" relationship to optimize experimental parameters for long-lasting electronic devices.

One of the most important properties for battery is cyclic voltammetry curve (CVC), which describes the efficiency for charge transfering. To evaluate CVC rates accurately, most straightforward way is to perform CVC scanning for several times. However, this characterization method is extremely complicated and time consuming, limiting further application. Thus, our idea is to find feasible and simple characterization methods to replace CVC method for devices functional testing. [1,2]

Resistance is a promising property to evaluate the battery performance, reflecting materials fabrication uniformity and charge separation ability for as-obtained devices.[3,4] But there is no relative research work to confirm that we could use "resistance" value to test if one battery works well or not. Since it is "indirect" characterization, we can't state or assert devices with smaller resistance would have improved CVC efficiency with 100% certainty. By analyzing 20 publications for resistance characterization, almost all of these article use words,[5-25] like "suggest" or "indicate" to describe the relationship between resistance and CVC. Machine learning method was considered to be a powerful technique to classify samples into several categories. Our idea is to use machine learning method to predict if we could use resistance value to test CVC properties of devices, which then could be used to evaluate if our battery works well or not.

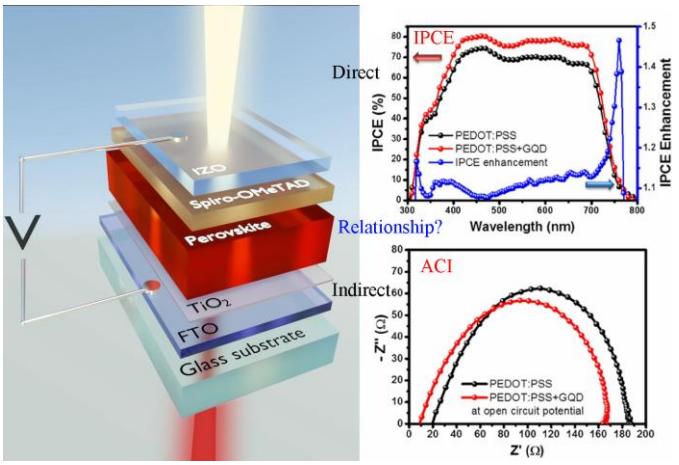


Fig. 1.1 Schematic picture of typical Li cell and example of CVC and resistance characterization.

Data and Features

Machine learning is a powerful weapon to predict structure-activity relationship in material science fields based

on tons of available and supportive experimental results. However, the most difficult part is to obtain enough data with sufficient features. Here, we characterize the CVC and resistance for 643 as-prepared devices with different Unit label in our lab (Nanoengineering Department in UCSD). To simplify our model, we denote "Work" when CVC values are beyond 10% and "Not Work" when CVC values are lower than 10%, shown in Figure 1.2. (CVC value is also called IPCE value.) To get reliable resistance value, two types of characterizations (three-electrode and four-electrode characterizations) were employed. 1000 times scanning tests were performed for each device using three-electrode method. We set the dataset name as "B" using 1 V applied voltage and "B2" using 1.5 V applied voltage. In addition, we performed four-electrode system to measure resistance with 120 repeating trials (dataset name "s").For each unit, there are 1000 observations for test B, 1000 observations for test B2, and 120 observations for test S. So we have 2120 features for each sample. Then, we perform binary classifier model with K-Nearest Neighbors method, Logistic Regression method, Bayes method, Random Forest method and other statistical learning method to study whether resistance could be potential to evaluate CVC performance for Li-battery devices.[26-28]

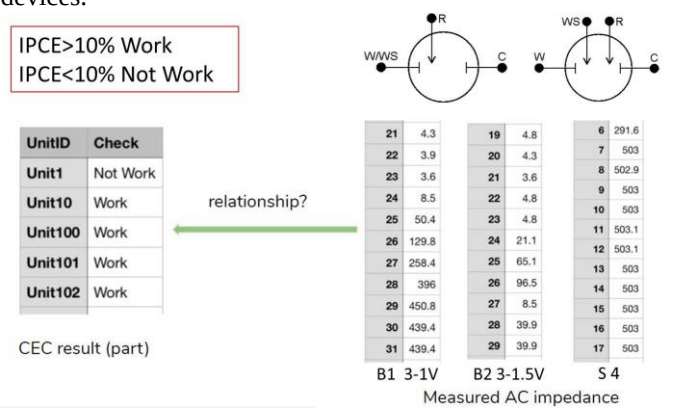


Fig. 1.2 CVC and resistance characterization results with data sample. Scheme of three electrode and four electrode measurement also shows on the top of right panel.

Methods

1. 1 Data cleaning

There are two categories of data samples. The first part is the data from CVC. We put all these data in "Results.csv". While the second part is the test data from resistance test, where we save all the data in file folder called "resistance". For the first part of data in file "Results.csv", there are 656 rows of data with certain unit ID and results. For clarity, we sort the data according to the unit ID. Typically, there are two kinds of outcomes: work or not work. To simplify the calculation process, we denote "Work" as "1" and "Not Work" as "0", and save the data as a list called "label" in Python. For the second part, we built a dictionary to save resistance data in Python. There are 3 types of resistance data, so we categorize these data into 3 different groups, which can be extracted by data ['B'], data ['B2'] and data['s']. For each unit, there should be 3 files for each unit, however, few of samples have more or less than three files. We checked all the data files and delete all the abnormal dataset. Then we only need to consider 643 units. And then we check if there is any data missing for both CVC data and resistance data: We denote the cleaned inspection data as list 'label_clean'. The number of "Work" is 441, which counts for about 68.58%. The number of "Not Work" is 202, which counts for about 31.42%. So we will balance the "class weight" for each method. We separate these data into training data set and test data set randomly with the ratio of 7:3.

1.2 Model building and evaluation

. There are two kinds of inspection results: "Work" or "Not Work". So that is a kind of binary classification problem. There are various kinds of method to address these problems. In order to find a suitable and accurate model to predict whether function F for certain product would work or not, eight models could be employed:

• K-Nearest Neighbors method: The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point.

• Logistic Regression method: Logistic regression is named for the function used at the core of the method, the logistic function. The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

• Bayes method

• Random Forest method: Random Forest has nearly the same hyperparameters as a decision tree or a bagging classifier. The difference is that Random Forest adds additional randomness to the model while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features.

• Decision Tree method

• Linear Discriminant Analysismethod

• Quadratic Discriminant Analysis method

• Support VectorMachine method

Results and Discussion

2.1 PRELIMINARY MODEL SELECTION

The chosen training data set was used to train the eight models we mentioned above. We first run all the models without tuning the parameters. Here, for each method, we use the cross validation method in the training procedure, and set cv=5. The results for Test B, Test B2 and Test S are shown in Fig. 2.1-2.3, respectively.

Figure 2.1: Preliminary Models for Test B

	Model	Fitting time	Scoring time	Accuracy	Precision	Recall	F1_score	AUC_ROC
5	Random Forest	0.087360	0.046800	0.964368	0.961007	0.957270	0.964323	0.973446
1	Decision Tree	0.383761	0.040560	0.939972	0.932493	0.930149	0.939887	0.930149
6	K-Nearest Neighbors	0.066082	0.622697	0.939948	0.950978	0.911831	0.938363	0.922368
0	Logistic Regression	1.011484	0.067762	0.911106	0.907890	0.887007	0.910066	0.887570
4	Quadratic Discriminant Analysis	0.172000	0.116520	0.764583	0.749416	0.680319	0.744758	0.738291
3	Linear Discriminant Analysis	0.256240	0.040560	0.760259	0.731678	0.746889	0.763978	0.789992
7	Bayes	0.074880	0.099840	0.728827	0.732425	0.763858	0.736951	0.766691
2	Support Vector Machine	0.699084	0.787686	0.708875	0.850136	0.545074	0.612721	0.666020

Figure 2.2: Preliminary Models for Test B2

	Model	Fitting time	Scoring time	Accuracy	Precision	Recall	F1_score	AUC_ROC
7	Bayes	0.070803	0.107005	0.979975	0.972543	0.983437	0.980165	0.983380
5	Random Forest	0.124007	0.071804	0.979950	0.973632	0.981505	0.980071	0.989457
6	K-Nearest Neighbors	0.073004	0.632436	0.977728	0.973914	0.976125	0.977792	0.990616
0	Logistic Regression	0.391973	0.038641	0.964467	0.961175	0.959053	0.964399	0.991359
1	Decision Tree	0.340376	0.064203	0.964443	0.961358	0.957367	0.964417	0.957367
3	Linear Discriminant Analysis	0.243607	0.046602	0.895477	0.897158	0.858926	0.892600	0.881445
4	Quadratic Discriminant Analysis	0.150683	0.120522	0.824387	0.869447	0.734784	0.805312	0.744837
2	Support Vector Machine	0.569628	0.802436	0.722233	0.855004	0.566010	0.639202	0.829376

Figure 2.3: Preliminary Models for Test S

	Model	Fitting time	Scoring time	Accuracy	Precision	Recall	F1_score	AUC_ROC
5	Random Forest	0.030280	0.01268	0.982222	0.978308	0.981221	0.982280	0.992166
6	K-Nearest Neighbors	0.009760	0.06700	0.980000	0.975495	0.979608	0.980079	0.991014
1	Decision Tree	0.003120	0.01248	0.975556	0.973742	0.970507	0.975479	0.970507
7	Bayes	0.003120	0.01248	0.973333	0.974320	0.964977	0.973189	0.968548
3	Linear Discriminant Analysis	0.029960	0.00996	0.871111	0.885550	0.810484	0.863952	0.730991
0	Logistic Regression	0.096841	0.00824	0.844444	0.863131	0.771544	0.833172	0.752650
2	Support Vector Machine	0.034520	0.05112	0.728889	0.815778	0.617166	0.669917	0.973906
4	Quadratic Discriminant Analysis	0.020200	0.01456	0.688889	0.344444	0.500000	0.561988	0.270737

We evaluate the results from two aspects: model efficiency and model accuracy. Seven factors (fitting time, scoring time, accuracy, precision, recall, F1_score and AUC_ROC) were chosen to evaluate model accuracy. We calculate both fitting time and scoring time to ensure this model could be efficient enough to widely applied in database with tons of data set. More importantly, we pay much attention to simulation accuracy for each model to confirm that we could obtain convinced predicting results. Thus, we rank the model according to the accuracy. Also, we found that all the accuracy related factors (precision, recall, F1_score and AUC_ROC) would be high for certain model whose accuracy is satisfying. Since Test B and Test B2 have the same dimensions of variable, we first compared these two dataset in Fig. 2.1 and Fig. 2.2. We could observe that Test B2 has better accuracy compared with Test B, indicating that we could obtain better predicting results if we use Test B2 rather than Test B. Later, we compare the modeling results for Test B2 and Test S roughly. For Test B2 the following three methods have good and similar accuracy:

• K-Nearest Neighbors method
• Random Forest method
• Decision Tree method

And for Test S, the following four methods have good and similar accuracy:

• K-Nearest Neighbors method
• Random Forest method
• Bayes method
• Decision Tree method

To better demonstrate the optimal resistance data set and modeling method, we run each for the above methods for Test B2 and S for 100 times, and calculate themean value of each result. Here, we use the cross validation method in the training procedure, and set cv=5. The results for Test B2 and Test S are shown in Fig. 2.4-2.5, respectively.

Figure 2.4: Selected Preliminary Models for Test B2

	Model	Fitting time	Scoring time	Accuracy	Precision	Recall	F1_score	AUC_ROC
2	K-Nearest Neighbors	0.028661	0.404744	0.982730	0.979462	0.981009	0.982745	0.989612
1	Random Forest	0.033248	0.021518	0.981149	0.976345	0.980848	0.981217	0.988793
0	Decision Tree	0.090886	0.018364	0.963821	0.961609	0.954986	0.963625	0.954986

Figure 2.5: Selected Preliminary Models for Test S

	Model	Fitting time	Scoring time	Accuracy	Precision	Recall	F1_score	AUC_ROC
2	K-Nearest Neighbors	0.002156	0.023859	0.983044	0.978154	0.983512	0.983112	0.990802
1	Random Forest	0.011949	0.005844	0.982247	0.978328	0.981283	0.982288	0.991155
0	Decision Tree	0.005613	0.002843	0.979288	0.975639	0.977207	0.979319	0.977207
3	Bayes	0.001848	0.003545	0.975170	0.975809	0.967125	0.974988	0.968468

Figure 2.6: K-Nearest Neighbor and Random Forest Method for Test B2

	Model	Fitting time	Scoring time	Accuracy	Precision	Recall	F1_score	AUC_ROC
0	Random Forest	0.045155	0.026283	0.982310	0.976174	0.984005	0.982423	0.990548
1	K-Nearest Neighbors	0.029714	0.412600	0.982126	0.979073	0.980102	0.982135	0.989110

Figure 2.7: K-Nearest Neighbor and Random Forest Method for Test S

	Model	Fitting time	Scoring time	Accuracy	Precision	Recall	F1_score	AUC_ROC
0	Random Forest	0.069561	0.019783	0.984981	0.979845	0.986326	0.985062	0.993241
1	K-Nearest Neighbors	0.002439	0.027443	0.982556	0.979013	0.981246	0.982583	0.991551

According to as-obtained results, we could observe that Test B2 and Test S have satisfying accuracy('>0.98') for K-Nearest Neighbor and Random Forest method. So next, we focus on improving K-Nearest Neighbor and Random Forest method based on Test B2 and Test S by tuning the parameters.

2.2 PARAMETERS TUNING

For tuning the parameters, I mainly use *GridSearchCV* in python. For Random Forest method, we pay attention to three kinds of parameters:

• max_features: Increasing max_features usually improves the performance of model. However, it is not necessarily since it will reduce the diversity of individual trees in a random forest. Also increasing max_features will slow down the algorithm. Therefore, we need to achieve the balance and choose the best max_features. Here I choose 11 for Test B2 and 9 for test S.

• n_estimators: This is the number of trees to build before making average prediction. The greater the number of trees, the better the performance, but the speed of running the code will be slower. Therefore, we should choose a value that is comparable to the processor's capabilities, which will predict more robust and stable. Here I choose 30 for Test B2 and 60 for test S.

• min_sample_leaf : A leaf is the end node of a decision tree. If the leaves are small, the model captures the noise in the training data and produces an overfitting. In general, we should choose a number larger than a certain value. Here I choose min_sample_leaf to be 40 for both Test B2 and test S.

For KNN Method, I focus on the following four parameters and also use *GridSearchCV* to tune the model:

• algorithm: I choose 'auto' for both Test B2 and test S.
• leaf_size: I choose 30 for both Test B2 and test S.
• n_neighbors: I choose 5 for both Test B2 and test S.

• weights: I use 'uniform' for Test B2 and 'distance' for test S.

2.3 VOTING CLASSIFIER

We use K-Nearest Neighbor and Random Forest method at the same time by Voting Classifier in Python. Since there are only two kinds of models, we consider the Soft Voting Classifier. We run the models for 100 times, and calculate the mean value of each result. We use the training data to train the Voting Classifier, and use the test data to evaluate the model. The results for Test B2 and test S are shown in Fig. 2.8-2.9. Also, the ROC-Curve of Voting Classifier for Test B2 and test S are shown in Fig. 2.10-2.11.

Figure 2.8: Voting Classifier for Test B2

	Model	Accuracy	Precision	Recall	F1_score	AUC_ROC
0	Ensembling_soft	0.985907	0.994076	0.985114	0.989548	0.993317

Figure 2.9: Voting Classifier for Test S

	Model	Accuracy	Precision	Recall	F1_score	AUC_ROC
0	Ensembling_soft	0.985596	0.995567	0.983297	0.989358	0.9939

Figure 2.10: ROC Curve of Voting Classifier for Test B2

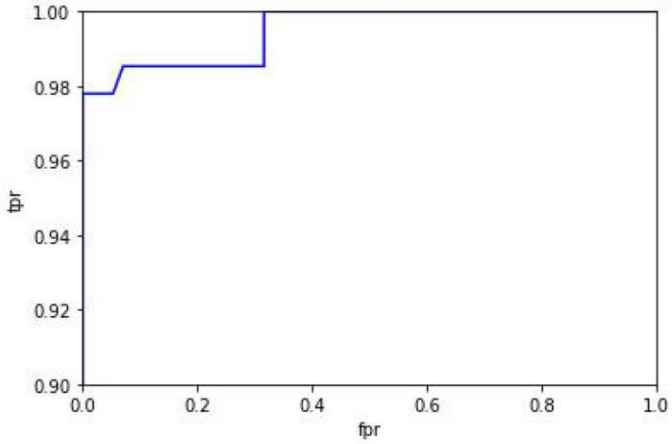
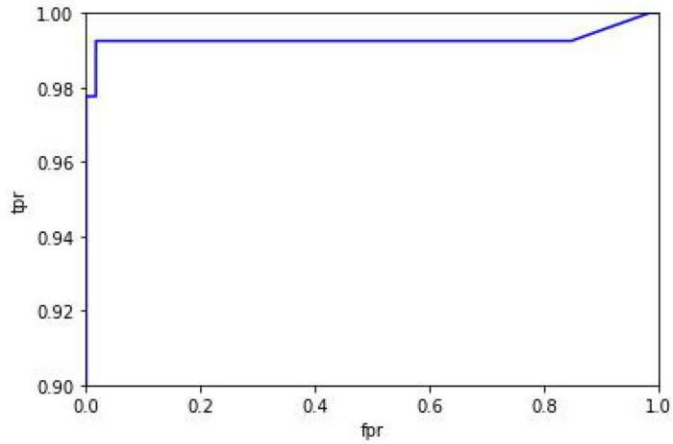


Figure 2.11: ROC Curve of Voting Classifier for Test S



Here we can see that predicting by voting classifier based on Test S data or Test B2 data is the optimal classification method:

For accuracy, it is about 98.6%, which is the highest among all the methods. It means almost all the units could get the correct results predicted by the method.

For Precision, it is about 99.5%, which is the highest among all the methods. It shows that almost all the units that are successfully predicted to be "work" for devices with lower *resistance* value..

For AUC-ROC, it is almost 99.4%, which is the area below the ROC curve. It means the voting classifier works very well overall.

Conclusions

We are pretty sure to conclude that there is certain relationship between CVC and resistance due to satisfied modeling accuracy from dataset "B", "B2" and "S". More specifically, the lower resistance value is, the higher CVC is. This result meets our guessing and expectation. Sample "s" exhibits the highest accuracy using KNN method, indicating that four-electrode system will provide us more reliable resistance value. It is the first time to direct prove the relationship between CVC and resistance value, which will definitely be helpful to simplify the characterization procedure for peer-research group. And it provides direct evidence that we could evaluate batteries function by testing its resistance value.

To polish and improve our work, we still need to take time to tune the parameter for specific model. Also, we could extract some other typical electrochemical properties (open-current density…) to studyif they have certain relationship with CVC value.

References

[1] Stranks, S. D. et al. Electron-hole diffusion lengths exceeding micrometer in an organometal trihalide perovskite absorber. Science, 342, 341–344 (2013).
[2] Nie, W. et al. High-efficiency solution-processed perovskite solar cells with millimeter-scale grains. Science 347, 522–525 (2015).
[3] Lee, M. M. et al. Efficient hybrid solar cells based o meso-superstructured organometal halide perovskites. Science 338, 643–647 (2012).
[4] Jeon, N. J. et al. Compositional engineering of perovskite materialsfor highperformance solar cells. Nature 517, 476–480 (2015).
[5] Kulbak, M. et al. Cesium enhances long-term stability of lead bromide perovskite-based solar cells. J. Phys. Chem. Lett. 7, 167–172 (2016).
[6] Eperon, G. E. et al. Inorganic caesium lead iodide perovskite solar cells. J. Mater. Chem. A 3, 19688–19695 (2015).
[7] Zhang, T. et al. Bication lead iodide 2D perovskite component to stabilize inorganic α-CsPbI3 perovskite phase for high-efficiency solar cells. Sci. Adv. 3, 1700841 (2017).
[8] Hu, Y. et al. Bismuth incorporation stabilized α-CsPbI3 for fully inorganic perovskite solar cells. ACS Energy Lett. 2, 2219–2227 (2017).
[9] Choi, H. et al. Cesium-doped methylammonium lead iodide perovskite light absorber for hybrid solar cells. Nano Energy 7, 80–85 (2014).
[10] Sutton, R. J. et al. Bandgap-tunable cesium lead halide perovskites with high thermal stability for efficient solar cells. Adv. Energy Mater. 6, 1502458 (2016).
[11] Ma, Q. et al. Hole transport layer free inorganic CsPbIBr2 perovskite solar cell by dual source thermal evaporation. Adv. Energy Mater. 6, 1502202 (2016).

[12] Chen, C.-Y. et al. All-vacuum-deposited stoichiometrically balanced inorganic cesium lead halide perovskite solar cells with stabilized efficiency exceeding 11%. Adv. Mater. 29, 1605290 (2017).
[13] Beal, R. E. et al. Cesium lead halide perovskites with improved stability for tandem solar cells. J. Phys. Chem. Lett. 7, 746–751 (2016).
[14] Li, X. et al. CsPbX3 quantum dots for lighting and displays: room-temperature synthesis, photoluminescence superiorities, underlying origins and white light-emitting diodes. Adv. Funct. Mater. 26, 2435–2445 (2016).
[15] Protesescu, L. et al. Nanocrystals of cesium lead halide perovskites (CsPbX3, X=Cl, Br, and I): novel optoelectronic materials showing bright emission with wide color gamut. Nano Lett. 15, 3692–3696 (2015).
[16] Li, B. et al. Graded heterojunction engineering for hole-conductor-free perovskite solar cells with high hole extraction efficiency and conductivity. Adv. Mater. 29, 1701221 (2017).
[17] Swarnkar, A. et al. Quantum dot–induced phase stabilization of α-CsPbI3 perovskite for high-efficiency photovoltaics. Science 354, 92–95 (2016).
[18] Li, X. et al. Improved performance and stability of perovskite solar cells by crystal crosslinking with alkyphosphonic resistanced ω–ammonium chlorides. Nat. Chem. 7, 703–711 (2015).
[19] Sesta, B. et al. 1H NMR, surface tension, viscosity, and volume evidence of micelle-polymer hydrophobic interactions: LiPFN-PVP system. J. Phys. Chem. B 101, 198–204 (1997).
[20] Zhao, Y. et al. A polymer scaffold for self-healing perovskite solar cells. Nat. Commun. 7, 10028 (2016).
[21] Yin, Q. et al. Micellization and aggregation properties of sodium perfluoropolyether carboxylate in aqueous solution. J. Ind. Eng. Chem. 42, 63–68 (2016).
[22] Ummadisingu, A. et al. The effect of illumination on the formation of metal halide perovskite films. Nature 545, 208–212 (2017).
[23] Kramer, D. Dependence of surface stress, surface energy and surface tension on potential and charge. Phys. Chem. Chem. Phys. 10, 168–177 (2008).
[24] Nam, J. K. et al. Potassium incorporation for enhanced performance and stability of fully inorganic cesium lead halide perovskite solar cells. Nano Lett. 17, 2028–2033 (2017).
[25] Lorena, A. C. et al. A review on the combination of binary classifiers in multiclass problems. Artif. Intell. Rev. 30, 19-37 (2008).
[26] Shiraishi, Y. et al, Statistical approaches to combining binary classifiers for multi-class classification. Neurocomputing 74, 680-688 (2011).
[27] Parker, C. et al, On measuring the performance of binary classifiers. Knowl. Inf. Syst. 35, 131-152 (2013).