Article

Improve Ubiquitination-site Prediction Using Physicochemical Properties and Machine Learning with Grid Search

Xiaoye Mo, Samuel Cho and Xia Jiang

Department of Biomedical Informatics, University of Pittsburgh, Pittsburgh, PA,USA
* Correspondence: xij6@pitt.edu

ABSTRACT

Background
Ubiquitination plays an important role in protein post-translational processes and has been found to be involved in a number of regulatory functions including proteasome degradation, DNA repair, transcription, signal transduction, endocytosis, and sorting. As the identification of ubiquitination site is critical to furthering our understanding of the mechanism of ubiquitination, various experimental and machine learning methods have been used to conduct this task. It has been an important but challenging task to improve the accuracy of ubiquitination site prediction. In this research, we explore the possibility of improving the prediction performance of machine learning by incorporating grid search in the training process.

Method
We developed grid search procedures for each of six widely used machine learning methods including NB, LR, DT, SVM, LASSO, and KNN, and applied them to ubiquitination site prediction using the six PCP datasets that were previously developed. For each of the ML methods, we developed a set of values for each of the tunable hyperparameters available to the method. These sets of values then can be combined to form a large grid of hyperparameter settings, and each of these settings is used in the grid search. We integrated 5-fold cross-validation in grid search to train and test ML models and applied an additional independent validation procedure by conducting a pre-training 80-20 sample split. We evaluated the performance of the six methods by comparing them side by side for each of the six datasets. We also compared the grid search results with the results that were previously published without doing grid search. To optimize the prediction performance, we trained 1.1 million ML models in total through grid search.

Results
We compared the overall prediction performance of these six methods, as well as their prediction performance when working with balanced vs. imbalanced data, and large-scale vs. small-scale data. From the perspective of dataset, we find that the overall performance of every PCP dataset has been significantly improved in this study compared to the previous study, with the percentage increase of the average AUC of all six datasets ranging from 7.9% (PCP-4) up to 17.0% (PCP-1). From the perspective of method, we find that three out of four methods significantly benefit from grid search comparing to their previously published non-grid search results, with the maximum AUC improvement as high as 47% (LASSO on PCP-5), 43.3% (NB on PCP-1), and 33.7% (SVM on PCP-6). SVM overall ranks number one, followed by KNN as the number two performer based on their average AUCs on all datasets. But these two also ranked the top two (KNN 76 days and SVM 15 days) in terms of the total running time that they need to do grid search. We also find that SVM, KNN, and DT tend to handle small-scale and imbalanced datasets better, while LR, and LASSO are doing well with large-scale and balanced datasets. NB is more sensitive to data imbalance while less sensitive to the size of a dataset.

Conclusions
Our results show that using grid search has improved the performance of ubiquitination prediction significantly. We find that the performance of a method is closely related to its hyperparameter setting and the type of data it handles. Even though SVM is on average an outperformer, none of the methods can provide the best performance for all datasets. When sufficient computing resources are well accessible, grid search is an effective way to identify both a top performing model for a machine learning method and a suitable machine learning method for a particular dataset.

Keywords: ubiquitination; ubiquitination-site; PCP; protein physicochemical properties; prediction; machine learning; grid search

BACKGROUND

Ubiquitin is a small regulatory protein that is found in nearly all eukaryotic cells, and is involved in a number of regulatory functions. Ubiquitin interacts with other proteins through an enzymatic, post-translational modification process known as ubiquitination (also referred to as ubiquitylation) [1,2]. During ubiquitination, ubiquitin binds to specific points on substrate proteins, known as ubiquitination sites. The binding consists of three enzymatic steps—activation, conjugation, and ligation—and the binding can take place with a single ubiquitin protein or multiple ubiquitin proteins linked into chains [1–3]. Ubiquitination has been found to be involved in regulatory functions such as proteasome degradation, DNA repair, transcription, signal transduction, endocytosis, and sorting [1–4]. Due to its role in many important regulatory processes, research into the molecular mechanisms of ubiquitination has been widely conducted [5]. One obstacle in researching ubiquitination is the difficulty in identifying the ubiquitination sites [4]. A number of experimental methods have been implemented to isolate ubiquitinated proteins in order to identify the ubiquitination sites, including the use of high-throughput mass spectroscopy, antibodies, binding proteins, and combined liquid-chromatography/mass-spectroscopy techniques [6–11]. However, due to the dynamic, transient, and reversible nature of the ubiquitination process, these experimental methods are very time- and labor-intensive and costly [3,12,13]. To improve efficacy and efficiency, as well as reduce cost, machine learning methods have been used to predict ubiquitination sites by learning from existing knowledge of protein sequences[3–5,12,13].

Machine learning is a powerful tool for classification. The Naïve Bayes (NB) classifier is a widely used machine learning method based on probability computations and aims to calculate conditional dependency with prior probability [14]. Rish conducted an empirical study of the Naïve Bayes (NB) classifier, finding that even if the classification decision probability estimates are inaccurate, the method still performs well in practice [15]. NB has been used widely in text and document classification tasks, where it has been found to give good results [16–18]. Logistic Regression (LR) has also found uses in a number of applications, such as forecasting potential for financial distress in companies, classification of Click or tap here to enter text.EEG readings, and the preservation of digital privacy [19–21]. Decision Trees (DT), another useful machine learning method, has great potential for land cover mapping problems [22]. Also, DT can be used to help in medical diagnosis and breast cancer survivability predictions [23,24]. Compared to other methods, Support Vector Machines (SVM) is better known and often utilized in tasks ranging from spam mail categorization, landslide susceptibility assessments, breast cancer prediction, and coronavirus disease detection, often with promising results [25–28]. Least Absolute Shrinkage and Selection Operator (LASSO) is a linear regression-based classification method that can be used in the prediction of malignancy [29]. The K Nearest Neighbors (KNN) algorithm has found potential for use in multi-labelling problems, such as facial expression recognition and road surface classification problems [30–32]. With the development of machine learning methods, more and more applications of machine learning will come out, which will help people to address some of the difficult problems such as ubiquitination-site prediction.

The performance of a machine learning model depends not only on the dataset, but also on the hyperparameters' settings. Grid search is a hyperparameter tuning method which generates new hyperparameters' settings through combinations of the available sets of hyperparameter values [33]. Grid search can be used with various machine learning methods, and in general the final performance after tuning is better [34]. Grid search has been applied in several research fields, for example, Seyedzadeh et al. used grid search to find a better hyperparameter setting for SVM to do building energy loads prediction [35]. In HIV/AIDS research, Belete and Huchaiah used grid search in predicting test results, with promising findings [36]. In the education field, Sadiq and Ahmed applied grid search to Decision Tree and SVM models to better classify students' performance in class and predict their final exam results [37]. For text classification problems, Ghawi and Pfeffer also developed a better grid search with KNN, which could have faster speed of tuning [38]. Moreover, when facing food storage issues, Liu et al. used the Decision Tree method to analyze and predict problems relating to grain storage through several factors; it was then used in a grid search to improve the final performance [39].

Machine learning methods have been applied to the ubiquitination site prediction problem. Researchers have tried different methods to get different features from protein sequences [40, 41,42]. Chen et al. developed CKSAAP_UbSite, which is a SVM-based method of ubiquitination site prediction that takes in k-spaced AA pairs as features [12]. HCKSAAP_UbSite, also developed by Chen et al., improves on previous work through combining SVM and LR [42]. Moreover, Chen et al. developed UbiProber, which is a tool for large-scale ubiquitination site prediction, based on SVM and KNN algorithms [41]. Cai et al. conducted ubiquitination site prediction using five different machine learning methods with the averaged PCP values of a protein segment [5]. Among different kinds of features, PCP is particularly valuable because it can define protein structures and join biochemical properties together in a usable manner [5,43,44]. But none of these previous studies has used grid search to optimize the prediction performance.

In this research, we hypothesized that optimizing machine learning hyperparameter setting with grid search can significantly improve ubiquitination site prediction using protein physicochemical properties. To test this hypothesis, we used grid search on six ML methods—NB, LR, DT, SVM, LASSO, and KNN—and applied them to ubiquitination site prediction on six PCP datasets [5]. Also, with 5-fold cross-validation method, the results would be much more confidential. With the help of grid search, better performance ubiquitination site predication application may come out.

METHODS

Tuning is the process of identifying the set of parameter values that are expected to produce the best prediction model from all sets of parameter values examined. Grid search is designed for conducting parameter tuning in a systematic way by going through all possible sets of hyperparameter values within a given range. In this study, we optimized the prediction performance of six classic machine learning methods by conducting hyperparameter tuning via grid search. These methods include Naïve Bayes (NB), Logistic Regression (LR), Decision Tree (DT), Support Vector Machine (SVM), Least Absolute Shrinkage and Selection Operator (LASSO), and k-Nearest Neighbors (KNN). We used the scikit-learn package [45,46] in Python to implement these machine learning classifiers. Like most machine learning methods, these methods have hyperparameters that we can tune to achieve better prediction performance. We conducted grid search to tune the hyperparameter values for each of the six ML methods. We next briefly describe these ML methods and each of their hyperparameters. The detailed hyperparameters and their values that we tested for each method are shown in Table 1.

Table 1. Machine Learning Hyperparameters and Values

Method	Hyperparameter Name	Values
GaussianNB	priors	none, [0.1,0.9], [0.2, 0.8], [0.3, 0.7], [0.4,0.6], [0.5,0.5], [0.6,0.4], [0.7,0.3], [0.8, 0.2], [0.9,0.1]
	var_smoothing	10^{-9}, 10^{-8}, 10^{-7}, …, 10^{-1}
LR	penalty	L2, none
	solver	newton-cg, lbfgs, liblinear, sag, saga
	class_weight	balanced, none
	C	10^{-4}~ 10^{4} (300 numbers distributed evenly on log scale)
DT	criterion	gini, entropy
	splitter	best, random
	max_depth	2, 4, 6, …, 32
	max_features	auto, sqrt, log2
	min_samples_split	0.1, 0.2, 0.3, …, 1.0
	min_samples_leaf	1
	max_leaf_nodes	7, 10, 15, None

	class_weight	None, balanced
SVM	C	2^{-5}, 2^{-3}, 2^{-1}, …, 2^{15}
	gamma	2^{-15}, 2^{-11}, 2^{-7}, …, 2^{3}
	kernel	rbf, linear, sigmoid, poly
LASSO	C	10^{-5}~ 10^{5} (400 numbers distributed evenly on log scale)
	solver	liblinear, saga
KNN	k_neighbors	1, 2, 3, …, 300
	weights	uniform, distance
	algorithm	auto, ball_tree, kd_tree, brute
	leaf_size	10, 14, 18, …, 38
	metric	euclidean, manhattan, chebyshev

Naive Bayes (NB) [47–53] represents a special type of Bayesian network (BN) model. A BN consists of a directed acyclic graph (DAG) $G = (V, E)$, whose nodeset V contains random variables and whose edges E represent relationships among the random variables. NB is a simplified BN which normally only contains one parent node and a set of children nodes. When a NB model is used to conduct classification, it is called a NB classifier. We used the GaussianNB classifier in this study, because of the binary nature. "var_smoothing" is a parameter that adds a user-defined value to the distribution's variance, in order to stabilize calculation and smooth the curve. We tested 11 "prior" combinations and 9 "var_smoothing" values from 1e-9 to 1e-1.

Logistic Regression (LR) [53–56] is a supervised learning classification method well suited for binary classification problems. It is named after the logistic function: a core function of LR for nonlinear transformation of the output value. Regularization can be used to train models that tend to have better generalization on unseen data by preventing the algorithm from overfitting the training dataset. C is the inverse of the regularization strength ($C=1/\lambda$). Smaller values of C result in stronger regularization. We tested 300 evenly spaced values of C on a logarithmic scale between 10-4 and 104. We also used L2 method to regularize the LR model.

Decision Tree (DT) [57–60] is a machine learning method, with which data are processed as a tree-like structure. Each internal node represents a test on a feature and each leaf node represents a class value. The parameter "max_depth" indicates how deep the tree can be; the deeper the tree, the more splits it will have, thus capturing more information about the data. We fit a decision tree with depths ranging from 2 to 32. The parameter "min_samples_split" governs the minimum number of samples required to split an internal node. The values we tested in our grid search are 0.1, 0.2, 0.3, 0.4, 0.6, 0.8, 0.9 and 1. "max_features" indicated the maximum number of features allowed when building a decision tree; we tested all values under 'none', 'log2' and 'sqrt'. The parameter "max_leaf_nodes" controls the maximum number of leaf nodes of each decision tree, and we tested values 7, 10, 15, and none. The parameters "max_depth" and "max_leaf_nodes" are important in controlling overfitting. The function "criterion" was used to measure the quality of a split; we tested with values 'gini' and 'entropy'.

Support Vector Machine (SVM) [61–67] is a machine learning method that utilizes support vectors to identify hyperplanes within a maximum margin. It is used for both regression and classification tasks. Support vectors are sets of data points contained within a margin to the hyperplane, and can influence the position and orientation of the hyperplane, which in turn can be used to classify or separate additional data points. C is an important hyperparameter that trades misclassification of training examples for simplicity of the decision surface. Smaller values of C result in a smoother decision surface, while larger values give the model greater freedom in selecting samples as support vectors. We tested values of C in the range 2-5, 2-3… 215. The parameter γ dictates how far the influence of a single training example can reach, which is defined as the inverse of the radius of influence of samples selected by the model as support vectors. Low values of γ indicate a "far" range of influence, while high values indicate a "close" range of influence. We tested values of γ in the range 2-15, 2-13… 23.

Least Absolute Shrinkage and Selection Operator (LASSO) [68–71] is a regression-based classifier method capable of conducting variable selection and regularization in order to enhance prediction performance and control overfitting. Alpha (α) is the sum of absolute values of coefficients, defines the trade-off between balancing residual sums of squares and magnitudes of coefficients. α can take various values that are greater than 0. We tested 400 evenly spaced α values on a logarithmic scale between 10-5 and 105

K-Nearest Neighbor (KNN) [72–74] is a supervised machine learning method used for both classification and regression tasks. KNN predicts the class value of an incoming sample according to its k-nearest neighboring data points. It assumes that cases with similar covariate values are near to each other. The parameter "k_neighbors" defines the number of selected training samples close in distance to a query point used to predict the label of the query. We tested 300 values in the range from 1 to 300. The parameter "weights" defines the weighting criteria used to assign a value to a query point. We tested both the two available values for weights, 'uniform' and 'distance'. 'Uniform' assigns uniform weights to each neighbor. 'Distance' assigns weights to neighbors proportional to the inverse of the distance from the query point, with closer neighbors weighing more. The parameter "metric" is used to choose the method for calculating distance. We tested all available values: 'euclidean', 'manhattan', and 'chebyshev'.

Datasets

The six PCP (PhysicoChemical Property) datasets were made available via a previous study [40]. As shown in Table 2, dataset 1 through 3 are balanced, because each of them contains the same numbers of positive and negative cases, while dataset 4 through 6 are imbalanced, because each of them contains different numbers of positive and negative cases. Moreover, datasets 1 and 6 are small-scale datasets, with fewer than a thousand cases each, and datasets 2 through 5 are large-scale datasets with thousands of cases each. Each of the six datasets contains 531 features.

Table 2. Case and Feature Counts of the Six Datasets

	Total # of cases	# Positive cases	# Negative cases	# Columns
PCP-1	300	150	150	531
PCP-2	6838	3419	3419	531
PCP-3	12236	6118	6118	531
PCP-4	4608	363	4345	531
PCP-5	3651	131	3520	531
PCP-6	676	37	639	531

Performance metrics and 5-fold cross validation

We performed grid search and recorded 64 different output values for each of the models trained. Contained within the output data is information about the computer system used, computation time, and measures for model performance. For a given binary diagnostic test, a *receiver operator characteristic* (ROC) curve plots the true positive rate against the false positive rate for all possible cutoff values. The *area under curve* (AUC) of an ROC measures the discrimination performance of a model [75]. We conducted 5-fold cross-validation to train and evaluate each model in a grid search. The entire dataset was split into a train-test set containing 80 percent of the cases and an independent validation set containing the remaining 20 percent. We then performed a 5-fold cross validation by evenly dividing the train-test set into 5 portions. The division was done mostly through random selection, making sure to keep approximately 20% of the positive cases and 20% of the negative cases within each portion to ensure a representative fraction of the dataset. Training and testing were repeated 5 times. Each time, a unique portion was used as the testing set to test the model learned from the training set, which combined the remaining four portions. The best performing model would be selected and reported by grid search based on the mean test AUCs resulting from 5-fold cross-validation.

RESULTS

Table 3. The Validation AUCs of the Best-Performing Models Selected by Grid Search

	P 1	P 2	P 3	P 4	P 5	P 6	erage
B	758	619	659	653	686	649	654
R	748	635	665	702	743	484	663
T	684	623	651	655	644	637	649
VM	726	674	713	725	658	770	711
SSO	751	636	665	697	735	488	662
NN	738	647	678	624	697	744	671
erage	734	639	672	676	677	612	668
aximum	758 (NB)	674 (SVM)	713 (SVM)	725 (SVM)	743 R)	7 (SVM)	711 (SVM)

As described previously, we conduct grid search with 5-fold cross-validation to identify the best-performing set of hyperparameter values based on the mean test AUCs. The best model would then be refitted from the entire train-test set using the best-performing set of hyperparameters values. The best model was validated and the validation AUCs were obtained using the remaining 20% of data, which were not used in the model training process. Table 3 shows the validation AUCs of the best performance models in ubiquitination sites prediction for each method for the six PCP datasets. Table 3 also contains the average and maximum AUCs of all methods for each of the six PCP datasets. The hyperparameter values of the best-performing models learned from six datasets are reported in Table 4.

We compared our results obtained using grid search with some previously published results obtained without using grid search [40]. Specifically, the previous search explored the feasibility of using machine learning methods to conduct ubiquitination sites prediction with the PCP datasets. During that research, four classic machine learning methods including NB, SVM, LR, and LASSO were tested with the six PCP datasets using default hyperparameter values. Table 5 shows the side-by-side comparison of our grid search result with the previously published non-grid search result that was obtained using the same method and same dataset. The percentage improvement of a grid search result upon the matching non-grid search result was also included in Table 5. We also included in Table 5 the average and maximum percentage AUC improvement of all datasets using grid search vs not using grid search for each of these four methods.

To further compare the grid search and non-grid search results, we conducted a Wilcoxon rank sum test using the wilcox.test() function included in the R package. We conjectured that grid search can overall significantly improve the prediction performance of machine learning methods. For each machine learning method, we paired the grid search result with the matching non-grid result that was obtained using the same PCP dataset, and then conducted a pairwise right-tailed (greater) Wilcoxon test for the method. The statistical testing results are included in Table 6.

Figure 1 contains a panel of six figures, one for each of the six PCP datasets. Each of these figures consists of the ROC curves of the best performance models, one for each of the six machine learning methods, and demonstrates a side-by-side performance comparison of the six methods. Figure 2 contains a panel of six boxplots, one for each of the six PCP datasets. We compared the mean test AUC values of all six methods side by side in these boxplots.

Table 4. The Hyperparameter Values of the Best-Performing Models

	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6
NB	{'priors': [0.6, 0.4], 'var_smoothing' : 0.001}	{'priors': [0.1, 0.9], 'var_smoothin g': 0.01}	{'priors': [0.9, 0.1], 'var_smoothing': 0.0001}	{'priors': [0.9, 0.1], 'var_smoothing': 0.0001}	{'priors': [0.1, 0.9], 'var_smoothin g': 0.01}	{'priors': [0.9, 0.1], 'var_smoothing': 1e-09}

Table 5. Comparison of Grid Search Results to the Previously Published Non-Grid Search Results

LR	{'C': 0.001250230145991456, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'sag'}	{'C': 0.3848436109487429, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'lbfgs'}	{'C': 0.0024620924014946257, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'newton-cg'}	{'C': 0.0029619103297725533, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'sag'}	{'C': 111.38398264826183, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'sag'}	{'C': 0.00015391855229902055, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'sag'}
DT	{'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 4, 'max_features': 'log2', 'max_leaf_nodes': 10, 'min_samples_split': 0.2, 'splitter': 'random'}	{'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 10, 'max_features': 'auto', 'max_leaf_nodes': None, 'min_samples_split': 0.3, 'splitter': 'best'}	{'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 12, 'max_features': 'auto', 'max_leaf_nodes': None, 'min_samples_split': 0.1, 'splitter': 'best'}	{'class_weight': None, 'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'max_leaf_nodes': 7, 'min_samples_split': 0.1, 'splitter': 'random'}	{'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 2, 'max_features': 'auto', 'max_leaf_nodes': 7, 'min_samples_split': 0.3, 'splitter': 'best'}	{'class_weight': None, 'criterion': 'gini', 'max_depth': 2, 'max_features': 'log2', 'max_leaf_nodes': 7, 'min_samples_split': 1.0, 'splitter': 'best'}
SVM	{'C': 0.125, 'gamma': 3.0517578125e-05, 'kernel': 'rbf'}	{'C': 8, 'gamma': 3.0517578125e-05, 'kernel': 'rbf'}	{'C': 32, 'gamma': 3.0517578125e-05, 'kernel': 'rbf'}	{'C': 512, 'gamma': 3.0517578125e-05, 'kernel': 'linear'}	{'C': 0.5, 'gamma': 3.0517578125e-05, 'kernel': 'linear'}	{'C': 0.5, 'gamma': 0.5, 'kernel': 'sigmoid'}
LASSO	{'C': 0.015239032437388288, 'solver': 'liblinear'}	{'C': 0.3063483452696084, 'solver': 'liblinear'}	{'C': 0.1720239724793599, 'solver': 'liblinear'}	{'C': 21.920584507173235, 'solver': 'saga'}	{'C': 0.09659672580093832, 'solver': 'liblinear'}	{'C': 0.003814640664431178, 'solver': 'saga'}
KNN	{'algorithm': 'auto', 'leaf_size': 10, 'metric': 'manhattan', 'n_neighbors': 69, 'weights': 'uniform'}	{'algorithm': 'ball_tree', 'leaf_size': 10, 'metric': 'manhattan', 'n_neighbors': 91, 'weights': 'uniform'}	{'algorithm': 'auto', 'leaf_size': 10, 'metric': 'manhattan', 'n_neighbors': 67, 'weights': 'uniform'}	{'algorithm': 'kd_tree', 'leaf_size': 26, 'metric': 'chebyshev', 'n_neighbors': 99, 'weights': 'uniform'}	{'algorithm': 'kd_tree', 'leaf_size': 22, 'metric': 'euclidean', 'n_neighbors': 98, 'weights': 'uniform'}	{'algorithm': 'auto', 'leaf_size': 10, 'metric': 'chebyshev', 'n_neighbors': 61, 'weights': 'uniform'}

AUCs		Naïve bayes	SVM	LR	LASSO
Dataset 1	Non-GS	0.529	0.66	0.724	0.693
	GS	0.758	0.726	0.748	0.751
	Percentage improvement	43.3%	10.0%	3.3%	8.4%
Dataset 2	Non-GS	0.533	0.604	0.641	0.604
	GS	0.619	0.674	0.635	0.636
	Percentage improvement	8.6%	11.6%	-0.9%	5.3%
Da	Non-GS	0.514	0.61	0.648	0.613

	GS	0.659	0.713	0.665	0.665
	Percentage improvement	28.2%	16.9%	2.6%	8.5%
Dataset 4	Non-GS	0.604	0.667	0.72	0.5
	GS	0.653	0.725	0.702	0.697
	Percentage improvement	8.1%	8.5%	-2.5%	39.4%
Dataset 5	Non-GS	0.551	0.676	0.724	0.5
	GS	0.686	0.658	0.743	0.735
	Percentage improvement	24.5%	-2.7%	2.6%	47.0%
Dataset 6	Non-GS	0.513	0.576	0.555	0.5
	GS	0.549	0.770	0.484	0.488
	Percentage improvement	7.0%	33.7%	-12.8%	-2.4%
Average percentage improvement over all datasets		19.95%	13.00%	-1.28%	17.70%
Max percentage improvement over all datasets		43.30% (Dataset 1)	33.70% (Dataset 6)	3.30% (Dataset 1)	47.00% (Dataset 5)

Table 6. Comparison of Grid Search and Non-Grid Search Results

X, Y (X greater than Y)	W	p-value	95 % CI
GS, Non-GS NB	35	0.0022	[0.051, Inf]
GS, Non-GS SVM	30	0.0325	[0.014, Inf]
GS, Non-GS LR	22.5	0.2602	[-0.08196, Inf]
GS, Non-GS LASSO	31	0.0219	[0.03299, Inf]

Figure 1. ROC Curves of Different Best Models Selected by Grid Search

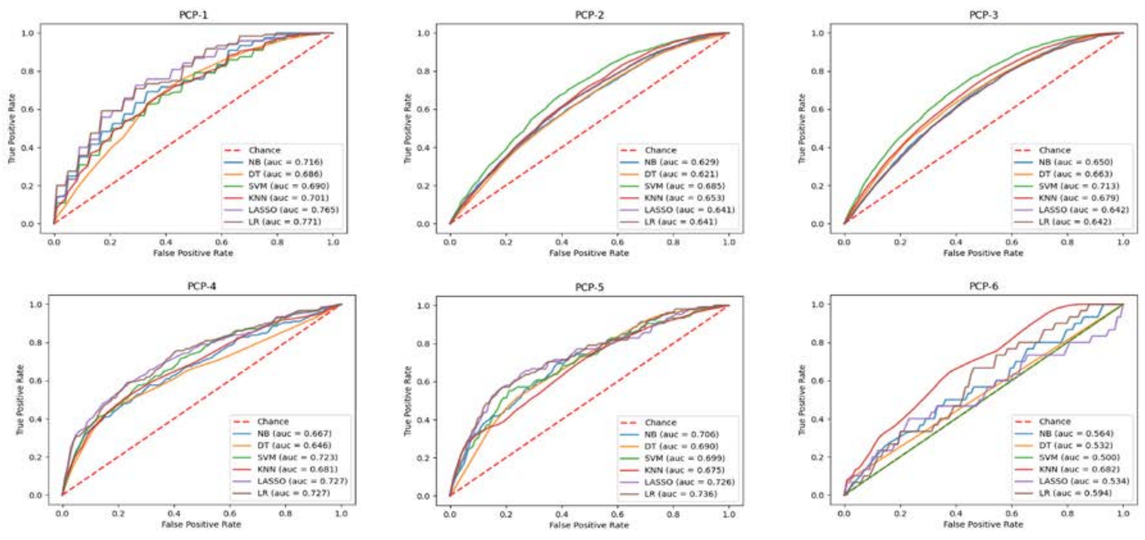
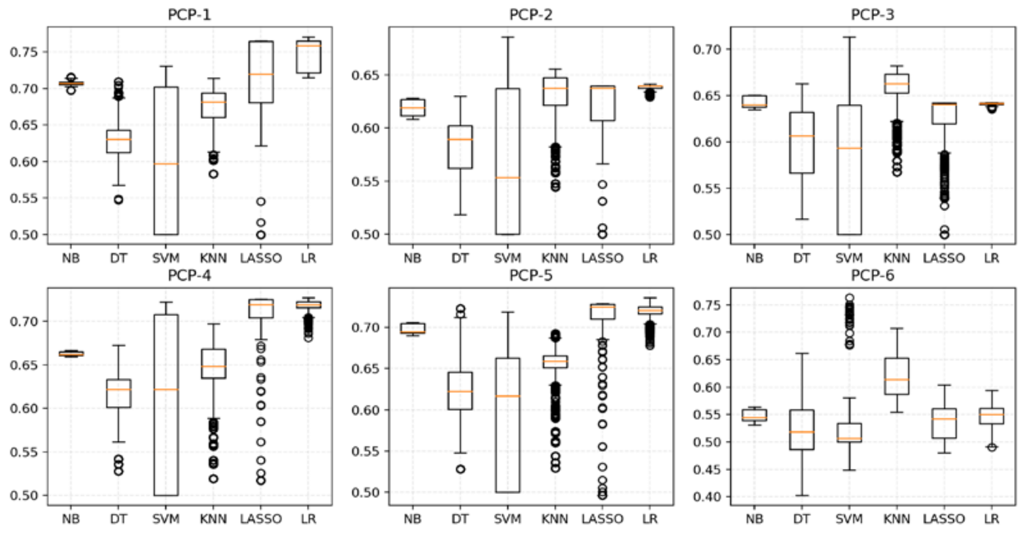


Figure 2. Mean Test AUCs of All Methods for Each of the Datasets



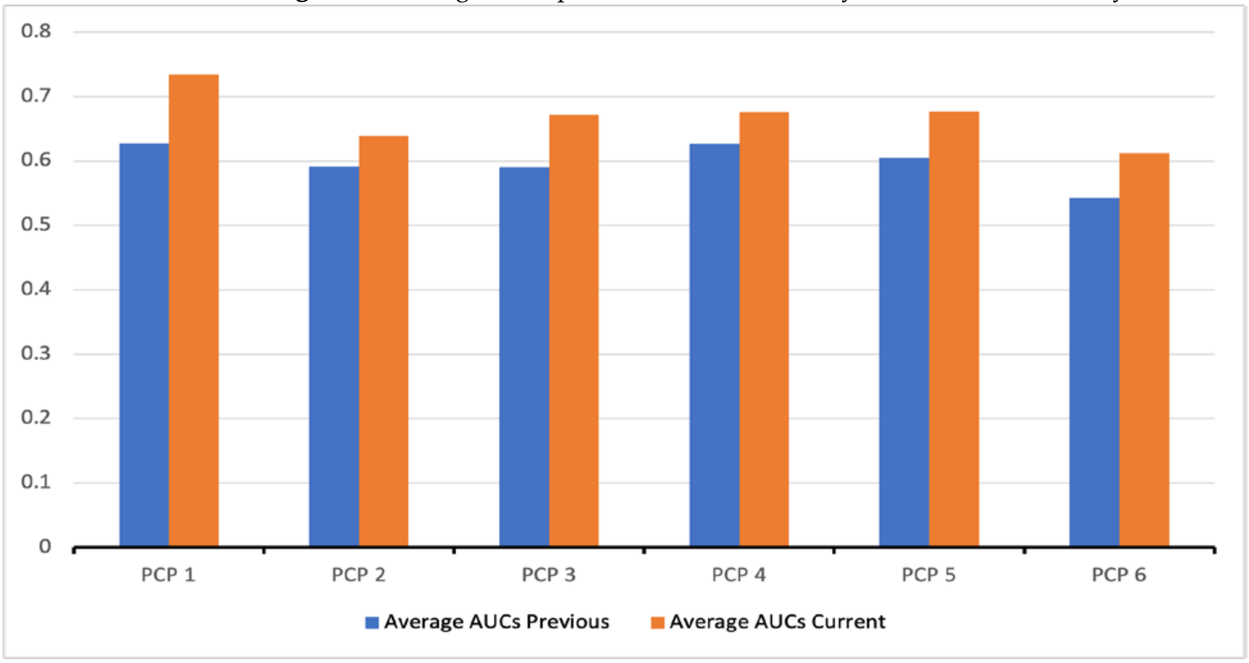
We compared per each dataset the average AUCs of all methods obtained in this research and the average AUCs of all methods previously published to get an idea of the overall prediction performance improvement from the perspective of datasets. A detailed comparison is shown in Table 7 and by the bar chart in Figure 3. In Table 7, we included both the average AUCs values, and the percentage AUC increase of the current study from the previously published research. In Figure 3, the orange bars represent the average AUCs of all methods for a dataset obtained in this research by doing grid search, and the blue bars represents the average AUCs of all methods for the same dataset, which were previously published.

Table 7. Average AUCs of This Study vs. Previously Published of Each Dataset

	PCP 1	PCP 2	PCP 3	PCP 4	PCP 5	PCP 6
Average Previous	0.628	0.591	0.591	0.626	0.604	0.542

Average Current	0.734	0.639	0.672	0.676	0.677	0.612
Percentage Increase	17.0%	8.1%	13.8%	7.9%	12.0%	12.8

Figure 3. Average AUC per Dataset in This Study vs. the Previous Study

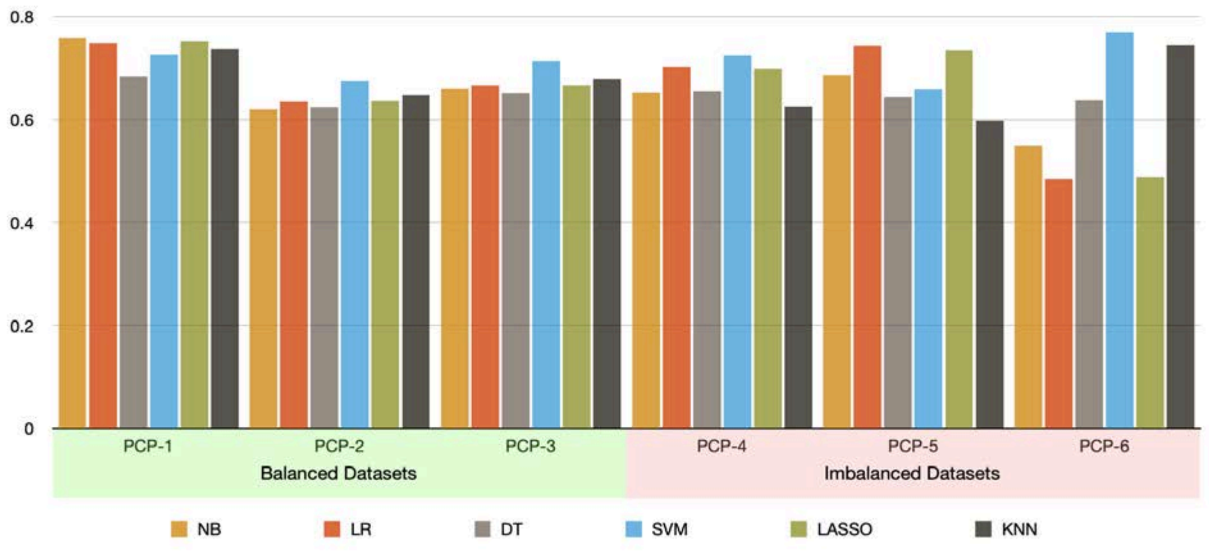


In order to understand the potential effects of data imbalance on prediction performance, we separated the six PCP datasets into two groups, the balanced group and imbalanced group. As shown in Table 2, PCP-1, PCP-2, and PCP-3 are in the balanced group, and PCP-4, PCP-5, and PCP-6 are among the imbalanced group. Using Table 8 together with a bar graph as shown in Figure 4, we compare the performance of the two groups side by side in terms of the validation AUC of the best model of each of the six machine learning methods. We also included the percentage AUC improvement of the balanced group relative to the imbalanced group in Table 8.

Table 8. Average AUCs of Balanced vs. Imbalanced Datasets

	NB	LR	DT	SVM	LASSO	KNN	Average
Balanced	0.679	0.683	0.653	0.704	0.684	0.688	0.682
Imbalanced	0.629	0.643	0.645	0.718	0.64	0.655	0.655
Percentage Increase	-7.3%	-5.8%	-1.1%	1.9%	-6.4%	-4.8%	-3.9%

Figure 4. Performance Comparison of Balanced vs. Imbalanced Datasets



We also separated the six PCP datasets into a large-scale group and a small-scale group based on the number of cases contained in a data set. Based on Table 2, PCP-2, PCP-3, PCP-4, and PCP-5 fall into the large-scale group, and each of them contains more than 3000 cases. PCP-1 and PCP-6 are in the small-scale group, and they each contains only a few hundreds of cases. We compare the two groups side by side, in terms of the validation AUC of the best model of each of the six machine learning methods, using a bar graph in Figure 5 and Table 9. We included the percentage AUC improvement of the large-scale group over the small-scale group in Table 8. We analyzed experiment time per method per dataset, number of models trained, and total experiment time and the summary data are shown in Table 10.

Table 9. Average AUCs of Large-scale vs. Small-scale Datasets

	NB	LR	DT	SVM	LASSO	KNN	Average
Small-scale	0.654	0.616	0.6605	0.748	0.6195	0.741	0.67308333
Large-scale	0.654	0.68625	0.64325	0.6925	0.68325	0.637	0.666
Percentage Increase	0.1%	11.4%	-2.6%	-7.4%	10.3%	-14.1%	-1.1%

Figure 5. Performance Comparison of Large-Scale and Small-Scale Datasets.

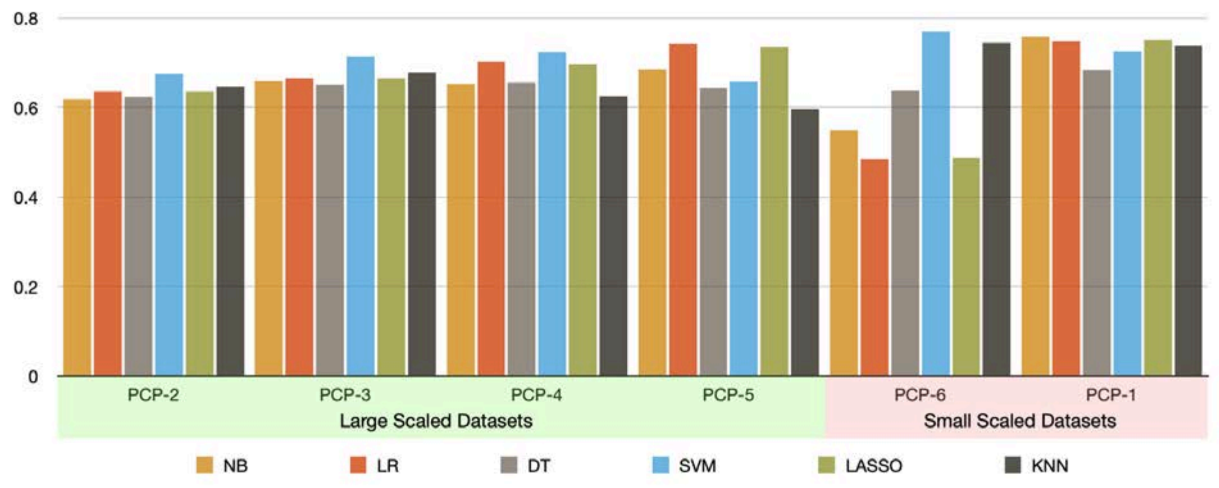


Table 10. Experiment Time Per Method Per Dataset, Number of Models Trained, and Total Experiment Time (seconds)

	NB	LR	DT	SVM	LASSO	KNN
Dataset 1	0.003	0.119	0.004	14.543	0.392	0.049
Dataset 2	0.103	3.092	0.057	177.097	22.607	13.330
Dataset 3	0.204	5.582	0.118	494.286	31.800	40.034
Dataset 4	0.075	1.943	0.052	39.451	1.976	9.214
Dataset 5	0.058	1.627	0.038	30.822	1.518	5.737
Dataset 6	0.008	0.241	0.010	10.620	0.309	0.041
# of Models Trained	2430	153000	345600	9900	24000	576000
Total Time (days)	0.002	3.720	0.185	14.644	2.713	76.000

DISCUSSION

As Shown in Table 3 and Figure 1, the highest validation AUC we obtained using grid search in this study was 0.77, and this was achieved by SVM when learning from dataset PCP-6. This is impressive because PCP-6 was the performance "bottleneck," and ranked the lowest at an averaged AUC of 0.54 among all six datasets in our previous study that did not conduct grid search [40]. SVM with grid search also brings the best performing models out of all six methods for datasets PCP-2, PCP-3, and PCP-4. Furthermore, SVM ranks number 1 among all methods in terms of the average AUC of all datasets as shown in Table 3 Column 8. Table 5 shows that SVM is indeed one of the methods that benefits greatly from grid search, with 13% averaged performance improvement and 33% maximum performance improvement among all datasets, when compared to its non-grid search results that were previously published. Note that all four best-performing SVM models have overall different hyperparameter settings based on Table 4. For example, the best-performing model for PCP-6 is a SVM model that uses the sigmoid kernel, the best-performing model for PCP-4 is a SVM model that uses the linear kernel, while the best-performing models for PCP-2 and PCP-3 are two SVM models that uses the rbf kernel but different 'C' values. This indicates that there is not a unique optimal hyperparameter setting that fits all datasets, which reflects further the importance of hyperparameter tunning with grid search to identify the best hyperparameter setting for each dataset.

Table 3 also shows that the second highest validation AUC (0.758) was achieved by a NB model resulting from grid search, trained from dataset PCP-1. Table 5 shows that NB on average benefits the most from grid search among all machine learning methods, with 19% average performance improvement of all datasets and 43.3% maximum performance improvement on dataset PCP-1, when compared to its non-grid search results that were previously published.

As shown in Table 5, LASSO also benefits greatly from grid search, ranking number one in terms of maximum percentage improvement (47% on dataset 5) and number two in terms of average percentage improvement (17.7% of all datasets). Note that in the previous study [40] LASSO performed very poorly on the three imbalanced datasets, which are PCP-4, PCP-5, and PCP-6, with an averaged AUC of 0.5 over these three datasets. By incorporating grid search in this study, the average AUC of LASSO over the three imbalanced datasets is increased 28% to arrive at 0.64.

According to Table 5, LR is the method that benefits from grid search the least, with its 3% maximum AUC improvement on PCP-1 and 1.28% decreased average AUC of all datasets. The possible reasons for this are 1) the default hyperparameters values for LR used in the previous study happened to be the best or near the best values; and 2) the hyperparameters of LR are not as sensitive to tuning as the ones of some other methods such as NB and LASSO.

Our statistical testing results as shown in Table 6 are consistent with the percentage improvement results shown in Table 5. They further demonstrate that grid search has significantly (at alpha=0.05) improved the prediction performance for three out of the four methods, namely NB, LASSO, and SVM, with p-values of 0.0022, 0.0219, and 0.0325 respectively. In this study, we also included DT and KNN, which are two widely used prediction methods. Since these two methods were not included in the previous study, they are not included in Table 5 and 6 which compare the current study and the previous study. However, as shown in Table 3, the average grid search results of these two methods are in line with the other four methods. Out of all six methods, KNN ranks number two in terms of the average AUC of all datasets. Although DT ranks at the bottom of the list, its average AUC (0.649) of all datasets is not too far off the overall average AUC (0.668) of all datasets and all methods, which can be obtained based on Table 3.

As Figure 2 shows, some methods can provide stable results regardless of what hyperparameters are used. For example, NB's interquartile ranges of 6 datasets are all very small, which means all results remain in a small range. However, SVM's interquartile ranges are the largest, from Figure 2, it ranges from 0.5 to almost the top. One interesting thing is that, although SVM is not stable for different parameters tested during training, it can provide good results on each dataset, as the maximum of the boxplot can almost reach the top.

From the perspective of the datasets, the best performer for PCP-1 is NB; the best performer for PCP-2, PCP-3, PCP-4, and PCP-6 is SVM; the best performer for PCP-5 is LR. This indicates that there is not a single method that performs the best for all datasets. Also, from the perspective of the datasets, we compare the average AUC of all methods for each dataset obtained in this study with the matching result of the previous study via Table 7 and Figure 3. We find that the overall performance of every PCP dataset has been significantly improved in this study, with the AUC increase ranging from 7.9% (PCP-4) to 17.0% (PCP-1).

It has been reported that data imbalance may have negative effect on prediction performance [76]. Due to this reason, we compare the AUCs of the three balanced datasets with the AUCs of the three imbalanced datasets side by side using a bar chart in figure 4. We also compare the average AUCs of all balanced vs all imbalanced data among all methods in Table 8. Our results overall substantiated the previous report but with some exceptions. We see from Table 8 that the data imbalance affects the prediction performance of each method somewhat differently. It negatively affected the performance of NB the most since the average AUC of all imbalanced data for NB is 7.2% lower than that of all balanced datasets. LASSO, LR and KNN are the next three methods following NB that are negatively affected by data imbalance, with an average AUC decrease of 6.9%, 6.2%, and 4.8% each respectively all imbalanced vs all balanced. DT is also affected negatively but only slightly. Interestingly, SVM seems to be affected positively by data imbalance, because its average AUC of all imbalanced datasets ends up 1.9% higher than that of all balanced datasets. Figure 4 shows a "dent" by PCP-6 on the imbalanced side, indicating that some of lowest AUCs occurred here. Although PCP-4, PCP-5, and PCP-6 are all imbalanced datasets, PCP-6 contains the least number of positive cases among all three. PCP-4 and PCP-5 each has more than 100 positive cases, but PCP-6 only has 37. This may have contributed to the performance drop of some of the methods including NB, LR, and LASSO when dealing with PCP-6. In the meantime, we notice from Figure 4 that SVM, KNN, and DT all performed very well with PCP-6, and this may indicate that these three methods are good in handling an imbalanced dataset that has a very low number of minor cases.

With Figure 5 and Table 9, we compare the average AUC of all large-scale datasets with that of all small-scale datasets for each method. We find that LR (11.4%) and LASSO (10.3%) benefit greatly from large-scale dataset, while KNN (-14.1%), SVM (-7.4%), and DT (-2.6%) are negatively affected by large-scale datasets, indicating that they can do better with small-scale datasets. NB (0.1%) is most insensitive to the size of a dataset. Figure 5 shows the curve on the large-scale side is overall smoother than that on the small-scale side, which may indicate that most of the methods handle the large-scale datasets more consistently than they do with the small-scale datasets.

We recorded the running time that it takes to train and fit a model when doing 5-fold cross-validation in grid search and computed the overall running time. As shown in Table 10, NB and DT are the two fastest methods among the six methods. In terms of the total running time, SVM and KNN are the slowest methods conducting grid search. They took weeks to finish all grid searches, which are time-consuming tasks. Grid search can help identify an outperforming model, but it can be costly in terms of computation time for some methods.

CONCLUSIONS

From the perspective of dataset, we find that the overall performance of every PCP dataset has been significantly improved in this study comparing to the previous study, with the increase of the average AUC of all six datasets ranging from 7.9% (PCP-4) up to 17.0% (PCP-1). From the perspective of method, we find that three out of four methods significantly benefit from grid search comparing to their previously published non-grid search results, with the maximum AUC improvement as high as 47% (LASSO on Dataset PCP-5), 43.3% (NB on PCP-1), and 33.7% (SVM on PCP-6). SVM overall ranks number one and followed by KNN as the number two performer based on their average AUCs of all datasets. But these two also ranked the top two (KNN 76 days and SVM 15 days) in terms of the total running time that they need to do grid search. We also find that SVM, KNN, and DT tend to handle small-scale and imbalanced datasets better, while LR, and LASSO are doing well with large-scale and balanced dataset. NB is more sensitive to data imbalance while less sensitive to the size of a dataset.

Finally, the performance of a method is closely related to its hyperparameter setting and the type of data it handles. Even though SVM is on average an outperformer, none of the methods can provide the best performance for all datasets. When computing resource is well accessible, grid search is an effective way to identify both a top performing model for a machine learning method and a suitable machine learning method for a particular dataset.

DECLARATIONS

Funding
Research reported in this paper was supported by the U.S. Department of Defense through the Breast Cancer Research Program under Award No. W81XWH1910495 (to XJ). Other than supplying funds, the funding agencies played no role in the research.

Authors' Contribution
XJ originated the study. XM, SC, and XJ wrote the first draft of the manuscript. XM and SC implemented the ML methods, and conducted the experiments. XM and XJ analyzed the results and performed the statistical analyses. All authors contributed to the preparation and revision of the manuscript.

Acknowledgement

Thank UPMC Hillman Cancer Center Academy for partially support this work.

REFERENCES

1. Welchman, R.L.; Gordon, C.; Mayer, R.J. Ubiquitin and Ubiquitin-like Proteins as Multifunctional Signals. *Nature reviews Molecular cell biology* **2005**, *6*, 599.

2. Herrmann, J.; Lerman, L.O.; Lerman, A. Ubiquitin and Ubiquitin-like Proteins in Protein Regulation. *Circulation research* **2007**, *100*, 1276–1291.
3. Tung, C.-W.; Ho, S.-Y. Computational Identification of Ubiquitylation Sites from Protein Sequences. *BMC bioinformatics* **2008**, *9*, 310.
4. Walsh, I.; di Domenico, T.; Tosatto, S.C.E. RUBI: Rapid Proteomic-Scale Prediction of Lysine Ubiquitination and Factors Influencing Predictor Performance. *Amino Acids* **2014**, *46*, 853–862.
5. Cai, B.; Jiang, X. Computational Methods for Ubiquitination Site Prediction Using Physicochemical Properties of Protein Sequences. *BMC Bioinformatics* **2016**, *17*, doi:10.1186/s12859-016-0959-z.
6. Kirkpatrick, D.S.; Denison, C.; Gygi, S.P. Weighing in on Ubiquitin: The Expanding Role of Mass-Spectrometry-Based Proteomics. *Nature cell biology* **2005**, *7*, 750.
7. Peng, J.; Schwartz, D.; Elias, J.E.; Thoreen, C.C.; Cheng, D.; Marsischky, G.; Roelofs, J.; Finley, D.; Gygi, S.P. A Proteomics Approach to Understanding Protein Ubiquitination. *Nature biotechnology* **2003**, *21*, 921.
8. Wagner, S.A.; Beli, P.; Weinert, B.T.; Nielsen, M.L.; Cox, J.; Mann, M.; Choudhary, C. A Proteome-Wide, Quantitative Survey of in Vivo Ubiquitylation Sites Reveals Widespread Regulatory Roles. *Molecular & Cellular Proteomics* **2011**, *10*, M111. 013284.
9. Xu, G.; Paige, J.S.; Jaffrey, S.R. Global Analysis of Lysine Ubiquitination by Ubiquitin Remnant Immunoaffinity Profiling. *Nature biotechnology* **2010**, *28*, 868.
10. Kim, W.; Bennett, E.J.; Huttlin, E.L.; Guo, A.; Li, J.; Possemato, A.; Sowa, M.E.; Rad, R.; Rush, J.; Comb, M.J. Systematic and Quantitative Assessment of the Ubiquitin-Modified Proteome. *Molecular cell* **2011**, *44*, 325–340.
11. Radivojac, P.; Vacic, V.; Haynes, C.; Cocklin, R.R.; Mohan, A.; Heyen, J.W.; Goebl, M.G.; Iakoucheva, L.M. Identification, Analysis, and Prediction of Protein Ubiquitination Sites. *Proteins: Structure, Function, and Bioinformatics* **2010**, *78*, 365–380.
12. Chen, Z.; Chen, Y.-Z.; Wang, X.-F.; Wang, C.; Yan, R.-X.; Zhang, Z. Prediction of Ubiquitination Sites by Using the Composition of K-Spaced Amino Acid Pairs. *PloS one* **2011**, *6*, e22930.
13. Cai, Y.; Huang, T.; Hu, L.; Shi, X.; Xie, L.; Li, Y. Prediction of Lysine Ubiquitination with MRMR Feature Selection and Analysis. *Amino acids* **2012**, *42*, 1387–1395.
14. Friedman, N.; Geiger, D.; Goldszmidt, M. Bayesian Network Classifiers. *Machine Learning* **1997**, *29*, doi:10.1023/a:1007465528199.
15. Rish, I. An Empirical Study of the Naive Bayes Classifier. *IJCAI 2001 workshop on empirical methods in artificial intelligence* **2001**, *22230*, doi:10.1039/b104835j.
16. Xu, S. Bayesian Naïve Bayes Classifiers to Text Classification. *Journal of Information Science* **2018**, *44*, doi:10.1177/0165551516677946.
17. Altheneyan, A.S.; Menai, M.E.B. Naïve Bayes Classifiers for Authorship Attribution of Arabic Texts. *Journal of King Saud University - Computer and Information Sciences* **2014**, *26*, doi:10.1016/j.jksuci.2014.06.006.
18. Wibawa, A.P.; Kurniawan, A.C.; Murti, D.M.P.; Adiperkasa, R.P.; Putra, S.M.; Kurniawan, S.A.; Nugraha, Y.R. Naïve Bayes Classifier for Journal Quartile Classification. *International Journal of Recent Contributions from Engineering, Science & IT (iJES)* **2019**, *7*, doi:10.3991/ijes.v7i2.10659.
19. Hua, Z.; Wang, Y.; Xu, X.; Zhang, B.; Liang, L. Predicting Corporate Financial Distress Based on Integration of Support Vector Machine and Logistic Regression. *Expert Systems with Applications* **2007**, *33*, doi:10.1016/j.eswa.2006.05.006.
20. Subasi, A.; Erçelebi, E. Classification of EEG Signals Using Neural Network and Logistic Regression. *Computer Methods and Programs in Biomedicine* **2005**, *78*, doi:10.1016/j.cmpb.2004.10.009.
21. Chaudhuri, K.; Monteleoni, C. Privacy-Preserving Logistic Regression. In Proceedings of the Advances in Neural Information Processing Systems 21 - Proceedings of the 2008 Conference; 2009.
22. Brodley, C.E.; Friedl, M.A. Decision Tree Classification of Land Cover from Remotely Sensed Data. *Remote Sensing of Environment* **1997**, *61*, doi:10.1016/S0034-4257(97)00049-7.
23. Kononenko, I. Machine Learning for Medical Diagnosis: History, State of the Art and Perspective. *Artificial Intelligence in Medicine* **2001**, *23*, doi:10.1016/S0933-3657(01)00077-X.
24. Delen, D.; Walker, G.; Kadam, A. Predicting Breast Cancer Survivability: A Comparison of Three Data Mining Methods. *Artificial Intelligence in Medicine* **2005**, *34*, doi:10.1016/j.artmed.2004.07.002.
25. Drucker, H.; Wu, D.; Vapnik, V.N. Support Vector Machines for Spam Categorization. *IEEE Transactions on Neural Networks* **1999**, *10*, doi:10.1109/72.788645.

26. Marjanović, M.; Kovačević, M.; Bajat, B.; Voženílek, V. Landslide Susceptibility Assessment Using SVM Machine Learning Algorithm. *Engineering Geology* **2011**, *123*, doi:10.1016/j.enggeo.2011.09.006.
27. Huang, M.W.; Chen, C.W.; Lin, W.C.; Ke, S.W.; Tsai, C.F. SVM and SVM Ensembles in Breast Cancer Prediction. *PLoS ONE* **2017**, *12*, doi:10.1371/journal.pone.0161501.
28. Sethy, P.K.; Behera, S.K.; Ratha, P.K.; Biswas, P. Detection of Coronavirus Disease (COVID-19) Based on Deep Features and Support Vector Machine. *International Journal of Mathematical, Engineering and Management Sciences* **2020**, *5*, doi:10.33889/IJMEMS.2020.5.4.052.
29. Bickelhaupt, S.; Paech, D.; Kickingereder, P.; Steudle, F.; Lederer, W.; Daniel, H.; Götz, M.; Gählert, N.; Tichy, D.; Wiesenfarth, M.; et al. Prediction of Malignancy by a Radiomic Signature from Contrast Agent-Free Diffusion MRI in Suspicious Breast Lesions Found on Screening Mammography. *Journal of Magnetic Resonance Imaging* **2017**, *46*, doi:10.1002/jmri.25606.
30. Zhang, M.L.; Zhou, Z.H. ML-KNN: A Lazy Learning Approach to Multi-Label Learning. *Pattern Recognition* **2007**, *40*, doi:10.1016/j.patcog.2006.12.019.
31. Thakare, P.P.; Patil, P.S. Facial Expression Recognition Algorithm Based On KNN Classifier. *IJCSN International Journal of Computer Science and Network* **2016**, *5*.
32. Fauzi, A.A.; Utaminingrum, F.; Ramdani, F. Road Surface Classification Based on LBP and GLCM Features Using KNN Classifier. *Bulletin of Electrical Engineering and Informatics* **2020**, *9*, doi:10.11591/eei.v9i4.2348.
33. Yang, L.; Shami, A. On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice. *Neurocomputing* **2020**, *415*, doi:10.1016/j.neucom.2020.07.061.
34. Isa, S.M.; Suwandi, R.; Andrean, Y.P. Optimizing the Hyperparameter of Feature Extraction and Machine Learning Classification Algorithms. *International Journal of Advanced Computer Science and Applications* **2019**, *10*, doi:10.14569/IJACSA.2019.0100309.
35. Seyedzadeh, S.; Pour Rahimian, F.; Rastogi, P.; Glesk, I. Tuning Machine Learning Models for Prediction of Building Energy Loads. *Sustainable Cities and Society* **2019**, *47*, doi:10.1016/j.scs.2019.101484.
36. Belete, D.M.; Huchaiah, M.D. Grid Search in Hyperparameter Optimization of Machine Learning Models for Prediction of HIV/AIDS Test Results. *International Journal of Computers and Applications* **2021**, doi:10.1080/1206212X.2021.1974663.
37. Sadiq, M.H.; Ahmed, N.S. Classifying and Predicting Students' Performance Using Improved Decision Tree C4.5 in Higher Education Institutes. *Lubricants* **2019**, *7*, doi:10.3844/jcssp.2019.1291.1306.
38. Ghawi, R.; Pfeffer, J. Efficient Hyperparameter Tuning with Grid Search for Text Categorization Using KNN Approach with BM25 Similarity. *Open Computer Science* **2019**, *9*, doi:10.1515/comp-2019-0011.
39. Liu, X.; Li, B.; Shen, D.; Cao, J.; Mao, B. Analysis of Grain Storage Loss Based on Decision Tree Algorithm. In Proceedings of the Procedia Computer Science; 2017; Vol. 122.
40. Cai, B.; Jiang, X. Computational Methods for Ubiquitination Site Prediction Using Physicochemical Properties of Protein Sequences. *BMC Bioinformatics* **2016**, *17*, doi:10.1186/s12859-016-0959-z.
41. Chen, X.; Qiu, J.D.; Shi, S.P.; Suo, S.B.; Huang, S.Y.; Liang, R.P. Incorporating Key Position and Amino Acid Residue Features to Identify General and Species-Specific Ubiquitin Conjugation Sites. In Proceedings of the Bioinformatics; 2013; Vol. 29.
42. Chen, Z.; Zhou, Y.; Song, J.; Zhang, Z. HCKSAAP-UbSite: Improved Prediction of Human Ubiquitination Sites by Exploiting Amino Acid Pattern and Properties. *Biochimica et Biophysica Acta - Proteins and Proteomics* **2013**, *1834*, doi:10.1016/j.bbapap.2013.04.006.
43. Kawashima, S.; Pokarowski, P.; Pokarowska, M.; Kolinski, A.; Katayama, T.; Kanehisa, M. AAindex: Amino Acid Index Database, Progress Report 2008. *Nucleic Acids Research* **2008**, *36*, doi:10.1093/nar/gkm998.
44. Tomii, K.; Kanehisa, M. Analysis of Amino Acid Indices and Mutation Matrices for Sequence Comparison and Structure Prediction of Proteins. *Protein Engineering* **1996**, *9*, doi:10.1093/protein/9.1.27.
45. Stancin, I.; Jovic, A. An Overview and Comparison of Free Python Libraries for Data Mining and Big Data Analysis. In Proceedings of the 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2019 - Proceedings; 2019.
46. Douglass, M.J.J. Book Review: Hands-on Machine Learning with Scikit-Learn, Keras, and Tensorflow, 2nd Edition by Aurélien Géron. *Physical and Engineering Sciences in Medicine* **2020**, *43*, doi:10.1007/s13246-020-00913-z.
47. Friedman, N.; Geiger, D.; Goldszmidt, M. Bayesian Network Classifiers. *Machine Learning* **1997**, *29*, doi:10.1023/a:1007465528199.

48. Chawla, N. v.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority over-Sampling Technique. *Journal of Artificial Intelligence Research* **2002**, *16*, doi:10.1613/jair.953.
49. Neapolitan, R. *Learning Bayesian Networks*; Prentice Hall: Upper Saddle River, 2004; ISBN 0130125342.
50. McCallum, A.; Nigam, K. A Comparison of Event Models for Naive Bayes Text Classification. *AAAI/ICML-98 Workshop on Learning for Text Categorization* **1998**, doi:10.1.1.46.1529.
51. Heckerman, D.; Geiger, D.; Chickering, D.M. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning* **1995**, *20*, doi:10.1023/A:1022623210503.
52. Friedman, N.; Linial, M.; Nachman, I.; Pe'er, D. Using Bayesian Networks to Analyze Expression Data. In Proceedings of the Journal of Computational Biology; 2000; Vol. 7.
53. Ng, A.Y.; Jordan, M.I. On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. In Proceedings of the Advances in Neural Information Processing Systems; 2002.
54. Friedman, J.; Hastie, T.; Tibshirani, R. Additive Logistic Regression: A Statistical View of Boosting. *Annals of Statistics* 2000, *28*.
55. Dreiseitl, S.; Ohno-Machado, L. Logistic Regression and Artificial Neural Network Classification Models: A Methodology Review. *Journal of Biomedical Informatics* **2002**, *35*, doi:10.1016/S1532-0464(03)00034-0.
56. Nusinovici, S.; Tham, Y.C.; Chak Yan, M.Y.; Wei Ting, D.S.; Li, J.; Sabanayagam, C.; Wong, T.Y.; Cheng, C.Y. Logistic Regression Was as Good as Machine Learning for Predicting Major Chronic Diseases. *Journal of Clinical Epidemiology* **2020**, *122*, doi:10.1016/j.jclinepi.2020.03.002.
57. Safavian, S.R.; Landgrebe, D. A Survey of Decision Tree Classifier Methodology. *IEEE Transactions on Systems, Man and Cybernetics* **1991**, *21*, doi:10.1109/21.97458.
58. Breiman, L. Random Forests. *Machine Learning* **2001**, *45*, doi:10.1023/A:1010933404324.
59. Ho, T.K. Random Decision Forests. In Proceedings of the Proceedings of the International Conference on Document Analysis and Recognition, ICDAR; 1995; Vol. 1.
60. Patel, B.R.; Rana, K.K. A Survey on Decision Tree Algorithm For Classification. *Ijedr* **2014**, *2*.
61. Suykens, J.A.K.; Vandewalle, J. Least Squares Support Vector Machine Classifiers. *Neural Processing Letters* **1999**, *9*, doi:10.1023/A:1018628609742.
62. Osuna, E.; Freund, R.; Girosi, F. Training Support Vector Machines: An Application to Face Detection. In Proceedings of the Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition; 1997.
63. Cortes, C.; Vapnik, V. Support-Vector Networks. *Machine Learning* **1995**, *20*, doi:10.1023/A:1022627411411.
64. Yang, Z.R. Biological Applications of Support Vector Machines. *Briefings in bioinformatics* 2004, *5*.
65. Chih-Wei Hsu, Chih-Chung Chang, and C.-J.L. A Practical Guide to Support Vector Classification. *BJU international* **2008**, *101*.
66. Guyon, I.; Weston, J.; Barnhill, S.; Vapnik, V. Gene Selection for Cancer Classification Using Support Vector Machines. *Machine Learning* **2002**, *46*, doi:10.1023/A:1012487302797.
67. Wang, H.; Zheng, B.; Yoon, S.W.; Ko, H.S. A Support Vector Machine-Based Ensemble Algorithm for Breast Cancer Diagnosis. *European Journal of Operational Research* **2018**, *267*, doi:10.1016/j.ejor.2017.12.001.
68. Tibshirani, R.; Saunders, M.; Rosset, S.; Zhu, J.; Knight, K. Sparsity and Smoothness via the Fused Lasso. *Journal of the Royal Statistical Society. Series B: Statistical Methodology* **2005**, *67*, doi:10.1111/j.1467-9868.2005.00490.x.
69. Park, T.; Casella, G. The Bayesian Lasso. *Journal of the American Statistical Association* **2008**, *103*, doi:10.1198/016214508000000337.
70. Hastie, T.; Tibshirani, R.; Wainwright, M. *Statistical Learning with Sparsity: The Lasso and Generalizations*; 2015;
71. Meier, L.; van de Geer, S.; Bühlmann, P. The Group Lasso for Logistic Regression. *Journal of the Royal Statistical Society. Series B: Statistical Methodology* **2008**, *70*, doi:10.1111/j.1467-9868.2007.00627.x.
72. Weinberger, K.Q.; Saul, L.K. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *Journal of Machine Learning Research* **2009**, *10*, doi:10.1145/1577069.1577078.
73. Keller, J.M.; Gray, M.R. A Fuzzy K-Nearest Neighbor Algorithm. *IEEE Transactions on Systems, Man and Cybernetics* **1985**, *SMC-15*, doi:10.1109/TSMC.1985.6313426.
74. Sun, S.; Huang, R. An Adaptive K-Nearest Neighbor Algorithm. In Proceedings of the Proceedings - 2010 7th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2010; 2010; Vol. 1.
75. Bradley, A.P. The Use of the Area under the ROC Curve in the Evaluation of Machine Learning Algorithms. *Pattern Recognition* **1997**, *30*, doi:10.1016/S0031-3203(96)00142-2.

76. Cohen, G.; Hilario, M.; Sax, H.; Hugonnet, S.; Geissbuhler, A. Learning from Imbalanced Data in Surveillance of Nosocomial Infection. *Artificial Intelligence in Medicine* **2006**, *37*, doi:10.1016/j.artmed.2005.03.002.