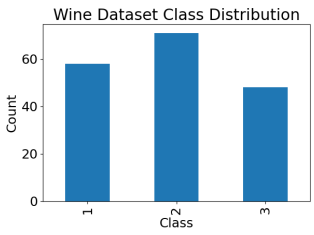
Exploring Supervised Learning Techniques on Different Datasets

Patricia Bata
bata.patricia@gmail.com

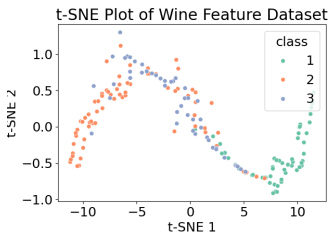
***Abstract*—This paper aims to demonstrate the conditions that affect the performance of different supervised learning algorithms, mainly Decision Trees, KNN, Boosting, Neural Networks, and SVM. Two multi-label classification datasets were used due to their varying behavior in output class and training features. An iterative process using grid search and validation curves was performed to obtain final model metrics for comparison. SVM (with feature scaling) and Boosted Trees performed the best on the Wine dataset, as it consisted of only numerical features and the output classes were distributed equally, implying that the dataset was linearly separable. On the other hand, SVM (no feature scaling) and Neural Networks performed the best on the Student Performance dataset, which had mixed features and imbalance classes, highlighting the capability of the two models to take into account multiple features and categorize them more robustly compared to other algorithms.**

I. Introduction - Dataset Explanation

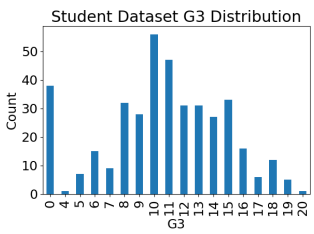
The "best model" is generally open to interpretation depending on (1) how the data behaves (i.e. its quantity, quality and type of features, the target class distribution) and (2) the main goal of the model you want to train. This paper aims to demonstrate how important understanding both your dataset and your end goal is to developing a well-performing tool for analysis.



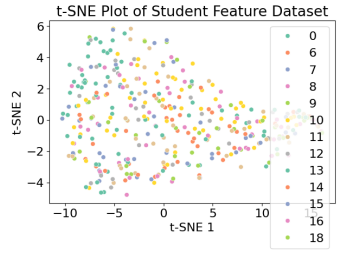
(a) Wine Category Distribution



(b) t-SNE of Wine Features



(c) Student Score Distribution



(d) t-SNE of Student Features (without Dropping)

Fig. 1: Dataset Feature and Target Distribution

The two datasets used in this analysis are both multi-label classification problems with variations in both the distribution of target class and in the features available to split the dataset into their respective categories.

The Wine dataset [1] is an analysis of wines grown within the same region in Italy from different cultivars. These cultivars make up the target variable (referred to as "class") with 13 purely numerical features describing each wine by its physical (color intensity, hue) and chemical (alcohol, alkalinity) attributes.

This dataset is considered to be well-behaved due to its balanced class distribution (Fig. 1a). Since all of features are numerical, feeding them into the algorithms caused no problems and no feature engineering was necessary (such as one-hot encoding). Resulting in a relatively small number of features fed into the algorithms to prevent possible confounding variables or overfitting.

With this, no data cleaning was done before feeding the training set into the algorithms. All features were included in the analysis and the standard, unstratified variation of the `KFold`[9] was used to split the dataset into 5 k-folds for validation, all while ensuring that each class was well represented in each fold. Weighted F1 score was used as the performance metric to compare models since the dataset is balanced and with less categories, our ideal model can focus true positives without sacrificing true negatives [8].

On the other hand, the Student Performance dataset [2] is not as well-behaved. The data documents students' grades from two Portuguese secondary education schools, with the students' demographics, family attributes, and school related features. There are two versions of the dataset, but this paper uses the math version. The target class is labeled as "G3", which denotes a student's final grade. Compared to the Wine dataset, Fig. 1c shows that there are much more classes to categorize and its distribution across isn't as well-distributed.

The dataset also includes a mix of categorical (school, family status) and numerical features (age, failures). Among the numerical features, students' intermediate grades from 1st and 2nd period are included, labeled as G1 and G2 respectively. These two features are strongly correlated to G3 which heavily influences models' performance when these are considered during training.

Contrasting these attributes to the Wine dataset, more data cleaning was done before training. Firstly, students having a unique G3 score (i.e. only 1 data point in

the class) were excluded from the dataset. In order to address the imbalance, this analysis experimented with using SMOTE was done on the remaining 16 classes (to increase variance per class being modeled) and compared it to model performances when dropping rare classes that comprised less than 2% of the dataset. After comparing the results, SMOTE showed either marginal improvement in test set performance or much worse compared to those trained on the dataset with rare classes dropped. This can be attributed to the minority classes being too rare and misrepresenting the class thus leading inaccurate and unreliable models [11].

To make sure each class was well-represented during cross validation, `StratifiedKFold` [9]. One-hot encoding was necessary in order to include categorical features in our analysis since all 5 algorithms required integers, floats or Boolean values as inputs. This resulted in a total of 41 training features. However during preliminary training, model performance suffered from using all features due to potential confounding (from one-hot encoding) and potential overfitting. Therefore, a mix of 20 categorical and numerical features was found to work best and used for each algorithms' model training. Both G1 and G2 were removed from the training features to avoid multicollinearity, which may lead to unstable and inaccurate coefficient and weight estimates [7]. Lastly, balanced accuracy was used as the model performance metric as comparison due to the imbalanced nature of the dataset, and the ideal model can correctly identify those in the minority classes as well as those in the majority.

Besides the feature and target distributions, the datasets vary in their class separability. To help visualize this, a scatter plot of two t-SNE components for the wine (Fig. 1b) and student (Fig. 1d) datasets are above.

In higher dimensions, it's easier to separate the classes in the Wine dataset compared to Student Performance. It's important to note that the Wine dataset cannot be separated by a single linear separator, even though there are visible differentiable regions for each class. On the other hand, the classes in the Student Performance dataset are mixed, with multiple overlaps between classes and no class-specific regions. Thus, this paper hypothesizes that the Wine dataset will perform best with the boosted trees and SVM algorithms, while the Student dataset with neural networks and SVM algorithms. This is because boosted trees and SVMs can find splits in higher dimension despite its non-linearity, while neural networks and SVM algorithms are robust enough to identify patterns inherent in the dataset even without clear differentiating features.

For each dataset and each algorithm, an initial grid search was done over multiple hyperparameters in order to determine the top 5 parameter combinations. Then the top 2 parameters with the most variations in the top models were explored further by both expanding the range and by observing the validation curves to ensure that the optimum points do not overfit or underfit given the training data. This was done using `validation_curves` function [9]. The intermediate chosen parameters were then run through the `learning_curve` [9] to observe the model's performance with different training proportion sizes. This was done iteratively until a balance was found between train and validation performance. The final model is then trained using `cross_validation` [9] to get the average model performance – quantified by the chosen score metric and confusion matrices – and run time. Each final model per algorithm was compared to each other to determine the best model for the dataset. All of the functions used for modeling were taken from the `scikit-learn` module [9] on Python.

II. Decision Trees

Decision Trees work best on datasets that can be definitively split by numerical thresholds and categories into classes. It works even better when a singular feature exists that can split categories into large chunks and this feature acts as the root node [4]. This paper used the `DecisionTreeClassifier` function [9].

For this analysis, the main parameter used for pruning the trees during training was the maximum tree depth. Not pruning or limiting this can quickly lead to overfitting. The two non-pruning parameter experimented with were minimum samples per leaf and minimum samples per split. For both the Wine and Student Performance dataset, the performance of these two parameters were comparable, so this paper focuses on minimum samples per leaf.

Comparing the learning curves of both datasets in Fig. 3a and 5a, Decision Trees does a great job with not overfitting when using all the training data, which is evidenced with the validation curves following the trend of training curves. In other words, the validation curves don't dip or decrease as training score increases.

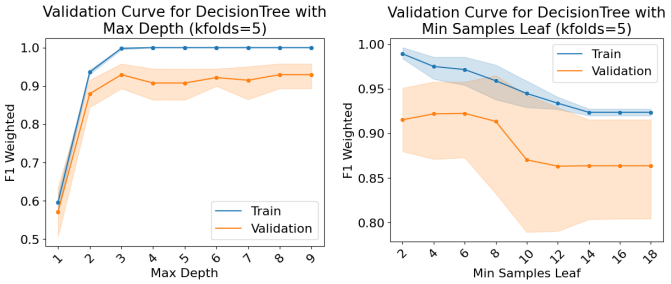
A. Wine

Fig. 2 shows the validation curves of the two most influential parameters while tuning. From these two plots, a max tree depth of 3 and a minimum sample per leaf of 8 maximize performance without overfitting. These parameters were used in the final model, whose results can be see in Fig. 16.

In the learning curves (Fig. 3a), even though both train and validation scores increase as training proportion increases, there's no drastic change in the weighted F1 score past the 0.2 training size proportion. This implies that its sufficient to train on 20% of the dataset without sacrificing information. Fig. 3b shows that Decision Trees does well with splitting the data into their categories. This can be attributed to the the wines having distinguishing separable chemical and physical features, translating to quantifiable splits in the decision tree.

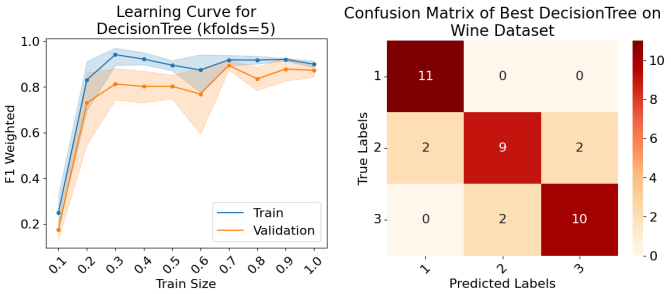
B. Student Performance

Similar to Wine, Student Performance's most helpful parameters during gridsearch was max tree depth



(a) Max Tree Depth (b) Min Sample Leaf

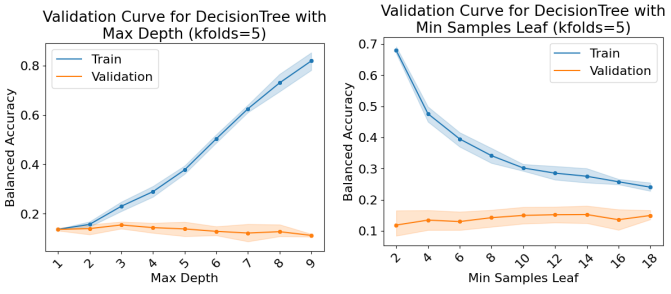
Fig. 2: Validation Curves on Decision Tree Parameters (Wine)



(a) Learning Curves (b) Confusion Matrix

Fig. 3: Best Decision Tree Performance (Wine)

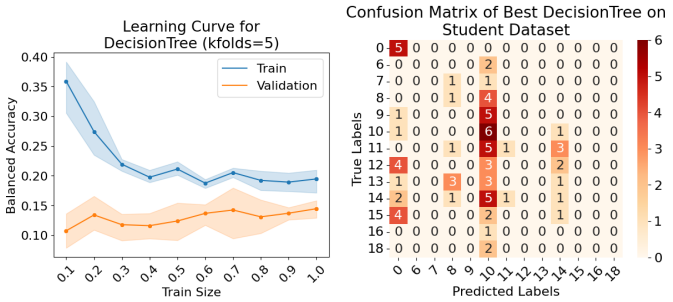
and minimum sample per leaf. However, these two parameters tended to overfit and underfit the model, respectively. This aligns with our hypothesis that using purely the training features to split the dataset is nearly impossible (Fig. 1d) – deepening the tree causes the model to be too confident while maximizing samples per leaf generalizes splits based on features too much, making the model under aggressive and decisive. Hence for the final model, a max tree depth of 3 and minimum sample per leaf of 2 was used.



(a) Max Tree Depth (b) Min Sample Leaf

Fig. 4: Validation Curves on Decision Tree Parameters (Student Performance)

In the learning curves (Fig. 5a) of Student Performance, initially the model overfits but gradually gets better, but it never plateaus or converges. This signifies that increasing the training data adds complexity to the dataset, probably due to the increasing number of output classes. In addition, it also implies that for decision trees to perform better, it's recommended to get more data to find more definitive splits in classification. This is supported by the confusion matrix on the test set (Fig. 5b) where the model classified almost all datapoints on the mode class.



(a) Learning Curves (b) Confusion Matrix

Fig. 5: Best Decision Tree Performance (Student)

III. KNN

The k-Nearest Neighbors (KNN) algorithm works best with clearly separate class clusters based on features since it relies on classifying new data points based on it's n-neighbors [6]. This paper used `KNeighborsClassifier` function.

The number of neighbors heavily influences how well the model predicts since this its basis, hence both dataset models were tuned on this parameter. The second hyperparameter to tune was more challenging to determine. Tuning on leaf size (when using Tree based algorithms), and algorithm types proved to have no effect on both train and validation performance. The best secondary parameter that affected performance was the the power metric, which specified between the Manhattan (p=1) or Euclidean (p=2) distance to use during computing the nearest neighbors.

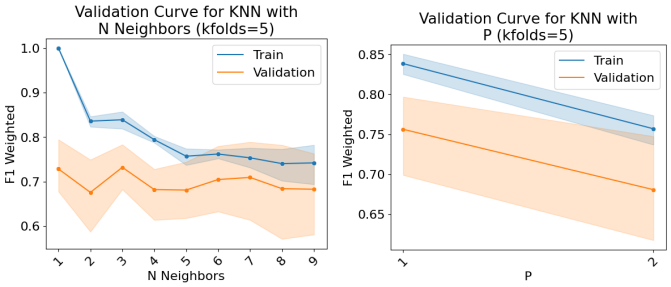
The learning curves for both datasets (Fig. 7a and 9a) follow the same pattern since during training, KNN does not technically compute anything for learning, which is why it's called a "lazy learner" [6]. It merely takes a look at all datapoints and determines the class via look-up table. Thus with more data, the validation set does better since there are more neighbors that increase feature variance to make more accurate predictions.

A. Wine

Fig. 6 shows the validation curves of n-neighbors and power metric while tuning. The most optimal number of neighbors that optimizes the validation set score without overfitting on the train set is 5. The power parameter that worked best for the model is 1 (Manhattan distance). Although it's a categorical parameter, it made the biggest impact in model performance compared to algorithm type and leaf size.

These results align with how the dataset is structured as seen in Fig. 1b. Since the classes clump together

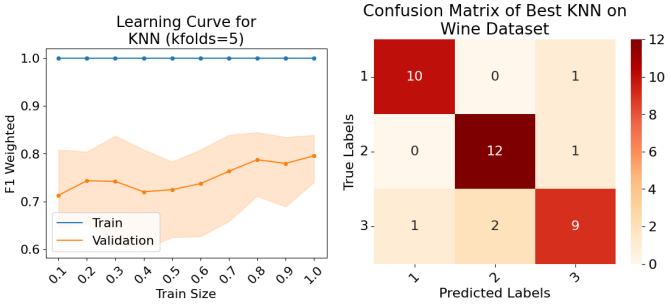
with some overlap, using too little doesn't maximize the clustered classes, but using too many neighbors adds noise due to the overlap. In addition, the Manhattan is preferred over Euclidian since the class clusters aren't restricted to a specific section.



(a) n-Neighbors (b) Power parameter

Fig. 6: Validation Curves on KNN Parameters (Wine)

The wine dataset had separable class divisions in its t-SNE projection (Fig. 1b), however there are overlap areas where neighbors don't belong to one class. This makes KNN not the most optimal algorithm for this dataset (Fig. 7b).



(a) Learning Curves (b) Confusion Matrix

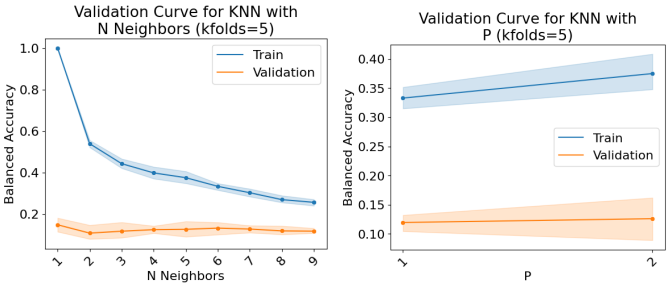
Fig. 7: Best KNN Performance (Wine)

B. Student Performance

Fig. 8 shows similar patterns with the same hyper-parametrized metrics compared to the Wine dataset. With the number of neighbors, as the number increases the less accurate the algorithm is at predicting since adding more neighbors in this case adds noise given the large overlap between data points (as seen in Fig. 1d). Here however, Euclidian distance performs marginally better since there's the potential addition of noise of neighbors obtained by getting the nearest neighbors using absolute distances given the class overlap on certain feature areas. Since the data point neighbors don't give much information, given that the classes are scattered around with no clear division, KNN does the worst at classifying the student dataset (Table I).

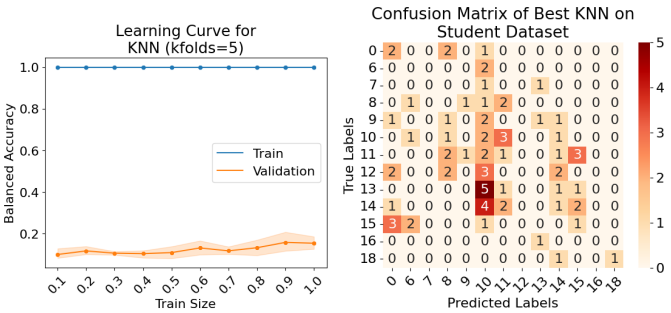
IV. Boosting

Boosting relies on ensemble learning, where multiple weak learners are trained sequentially to reduce



(a) n-Neighbors (b) Power parameter

Fig. 8: Validation Curves on KNN Parameters (Student Performance)



(a) Learning Curves (b) Confusion Matrix

Fig. 9: Best KNN Performance (Student)

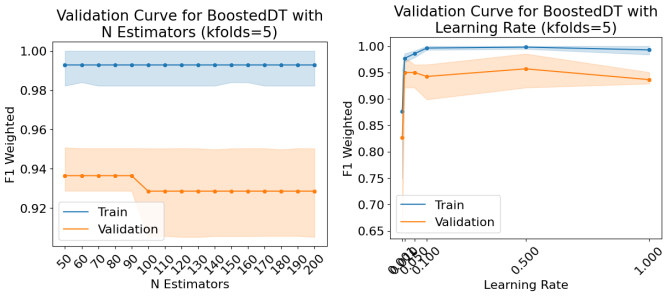
errors and avoid overfitting [10]. It builds upon the Decision Tree concept by assigning higher weights to data points that are wrongly classified. This paper used the `AdaBoostClassifier` function [9], with the weak learners limited to `DecisionTreeClassifier` and `RandomForestClassifier` while limiting the tree depth to 1 to ensure weak learners are trained.

The number of estimators plays a big role in model performance because with each estimator, the weak learners glean more information, make different mistakes, and vary the weight placed on decisions made. Hence, it's the best parameter to tune for optimal performance, and to verify if the model under or overfits. Learning rate comes hand-in-hand with number of estimators since higher learning rates increase the contribution of each classifier to prediction so it's important to balance both of these factors and find the optimal combination without under or overfitting.

A. Wine

Fig. 10 shows the validation curves of the number of estimators and learning rate while tuning. The number of estimators very gradually increases up to 90, then dips and starts overfitting. This may be because there are not many features present in the dataset and further splitting the dataset leads to more aggressive and less ideal splits. On the other hand, the learning rate stays at a slight plateau after 0.05. There is no drastic dip in

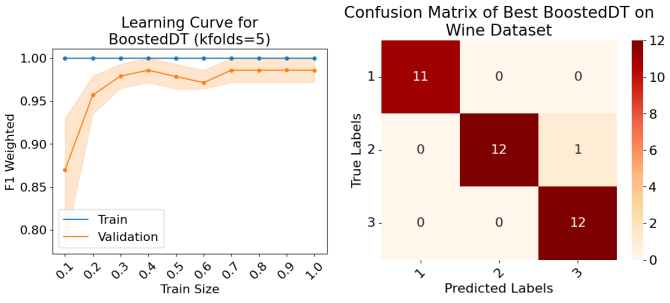
model performance, thus even at high learning rates, the model doesn't overfit. This is further supported by the high performance metric and excellent splits in the confusion matrix (Fig. 11b).



(a) n-Estimators (b) Learning Rate

Fig. 10: Validation Curves on AdaBoost Parameters (Wine)

The learning curve in Fig. 11a shows that the model converges when using the entire dataset in training, signifying that there's no under or overfit. The training metric stays constantly high across all training size proportions indicates that Boosted Trees are able to split the classes efficiently and as the proportion increases, the algorithm identifies more features and thresholds to split better on.



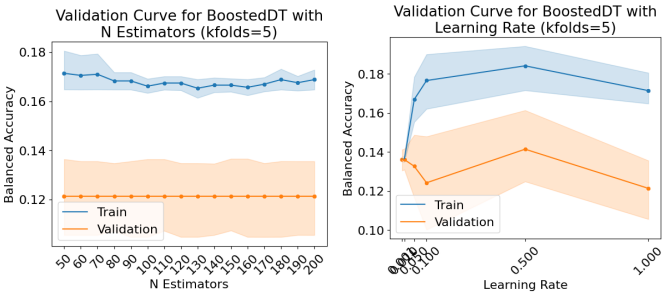
(a) Learning Curves (b) Confusion Matrix

Fig. 11: Best AdaBoost Performance (Wine)

B. Student Performance

On the other hand, the Student dataset's validation curves demonstrate that trained models give us little to no information. Despite the increase in estimators, the algorithm does not show any improvement in splitting the dataset efficiently. This implies that the training features don't have any distinguishable divide between the students' score category. Thus the best choice is to assign the minimum number of estimators to prevent overfitting. In addition, the optimal learning rate is 0.5, leaning on the aggressive weight for the models to make up for less estimators.

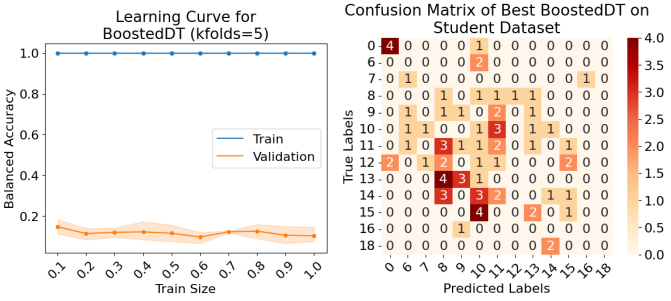
The learning curve in Fig. 13a supplements our observation that Boosted Trees does poorly classifying the dataset. Even with an increase in training set proportion,



(a) n-Estimators (b) Learning Rate

Fig. 12: Validation Curves on Boosted Tree Parameters (Student Performance)

the validation set performance does not improve even with near perfect performance on the train set. This implies that getting more data will not improve the performance of the model and this algorithm is not suited for this dataset. The features on its own cannot separate the data points into their classes. The aligns with Fig. 13b, showing that the model's predictions are sporadic and unreliable, not getting predictions correct on a single class.



(a) Learning Curves (b) Confusion Matrix

Fig. 13: Best Boosted Trees Performance (Student Performance)

V. Neural Networks

Neural Networks are more versatile in handling splitting datasets with no clear distinguishing features. Under the hood of the black box model, it can identify hidden interactions and splits better than most algorithms. However they lack explainability and take the longest to train and tune [12]. To limit the depth of the network, this analysis used the `MLPClassifier` function [9].

Because neural networks are flexible in training and tuning, there are multiple hyperparameters to work with when optimizing the model. This paper conducted a grid search over the: learning rate, alpha, activation function, depth of hidden layers, and size of hidden layers. The number of epochs were set constant throughout all models to standardize tuning.

To evaluate how well neural networks perform on each dataset, the loss curves (loss per epoch) for each

dataset is shown below (Fig. 14). Both curves indicate that the neural networks were able to learn from the dataset and minimize the loss, with both datasets plateauing at the 150th epoch. This shows that increasing number of epochs past 200 will not significantly improve model performance and an early stop at 150 is sufficient to learn enough to minimize the loss to increase modeling speed.

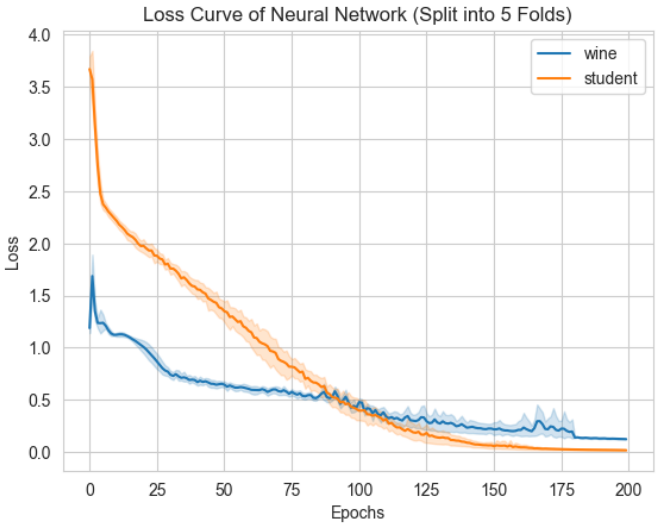


Fig. 14: Loss Curves from Best Neural Network for Wine and Student Performance Dataset

The two most influential parameters were hidden layer size and learning rate, which aligns mathematically aligns with how neural nets learn, since these two are the driving force for what and how fast the model studies the dataset features to split them into their respective classes.

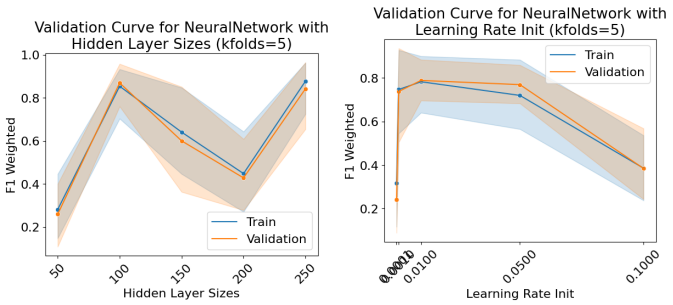
A. Wine

Fig. 15 shows the validation curves of hidden layer size and learning rate while tuning. The behavior of hidden layer sizes implies that the validation set performs just as well as the train set, especially with the overlap in confidence intervals. A similar trend can be seen with the learning rate. This can be an indication that neural networks can identify the patterns in the Wine dataset well, with a slight risk of overfitting.

The learning curve in Fig. 16a follows a similar pattern to the validation curves in Fig. 15, supporting the claim that the algorithm was able to see the dataset patterns enough to have similar model performance to the trained dataset. The confusion matrix in Fig. 16b shows that even though the trends are similar to the train set, the model does not overfit, however the misclassificatied data points may be indicators that outliers cannot be handled by the model.

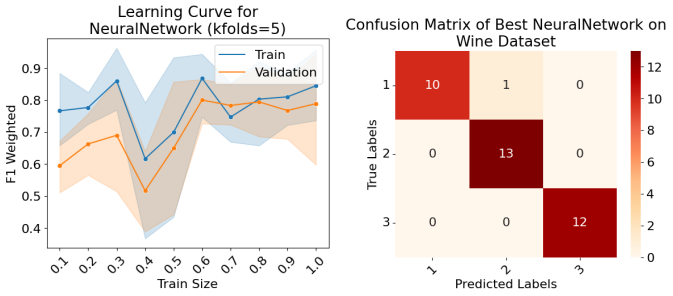
B. Student Performance

Contrasting the Wine dataset, the validation curves (Fig. 15) of the same hyperparameters in the Student



(a) Hidden Layer Sizes (b) Learning Rate

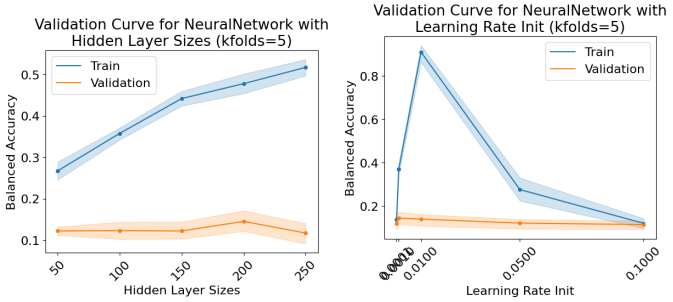
Fig. 15: Validation Curves on Neural Network Parameters (Wine)



(a) Learning Curves (b) Confusion Matrix

Fig. 16: Best Neaural Network Performance (Wine)

Performance dataset don't converge or overlap. In the hidden layer sizes, the model starts to overfit the data then reaches a balance at 200, then dips again, indicating that a more complex network would have the tendency to overfit the model. A similar trend can be seen in the learning rate. As the network tries to learn quicker and more aggressively, its performance on the validation set suffers. This may be because a deeper network analyzes the patterns in the training but these patterns do not translate to the validation or test set. In addition, aggressively learning these patterns with a higher learning rate can lead to overfitting.



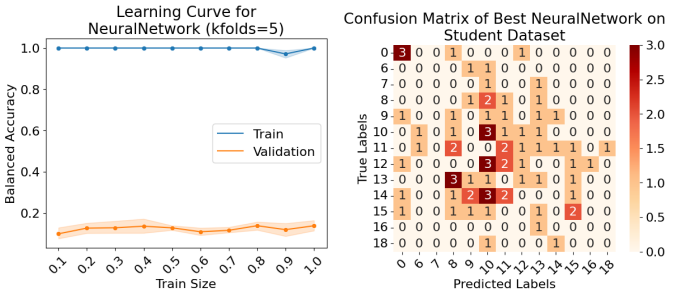
(a) Hidden Layer Sizes (b) Learning Rate

Fig. 17: Validation Curves on Boosted Tree Parameters (Student Performance)

The learning curve in Fig. 18a supports are findings from the validation curves as well. Despite great model

performance on the train set, the validation set does not improve with more data. This contrasts to the Wine dataset the similar line trends shhow that the train set in Wine represents the validation and test data well.

However, despite the poor performance across training size proportions, the neural network was still able to pick up patterns and intricacies other algorithms could not, thus doing a better job than most, evidenced by good classifications in 3 of 16 classes in Fig. 18b and the 2nd best performing model in Table I.



(a) Learning Curves (b) Confusion Matrix

Fig. 18: Best Boosted Trees Performance (Student)

VI. SVM

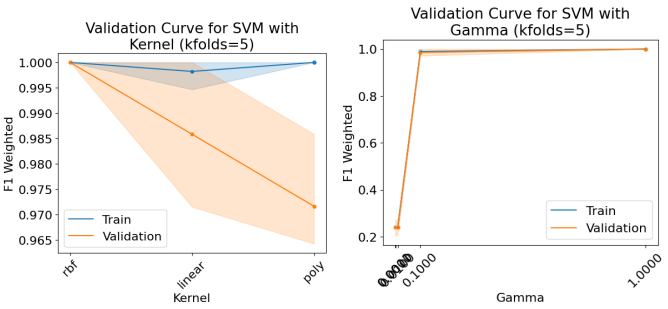
Support Vector Machines (SVMs) classify datasets by finding hyperplanes that split them up into their target classes. Feature scaling and normalization are preprocessing methods come highly recommended to improve model performance and convergence speed. This helps the hyperplane get a more restricted state to split the output classes much easier [3]. This analysis used the `SVC` function [9] for modeling.

Kernel functions are used to map datasets to higher dimensions to make class separation much easier – thus making SVMs one of the, if not the most robust algorithm out of the all ones analyzed [5]. The different types of kernel functions were investigated using validation curves, as well as the kernel coefficient (gamma). If the best kernel function isn't either RBF, polynomial, or sigmoid, this parameter is scrapped from the model fit.

A. Wine

Fig. 19 shows the validation curves of kernel types and coefficient (gamma). For the Wine dataset, the radial basis function (RBF) yielded the best results. This aligns with non-linearity of the features, represented by Fig. 1b. The kernel coefficient on the other hand, did not show any variation in the validation set from the test set, so the highest value (1) was used in the best model to force the algorithm to focus within closer neighbors and avoid noise, imposing influence on far away datapoints in high dimension.

The learning curve in Fig. 20 shows that the SVM model generally performs better as more training data is used and it doesn't reach an under or overfitting point.



(a) Kernel (b) Gamma

Fig. 19: Validation Curves on SVM Parameters (Wine)

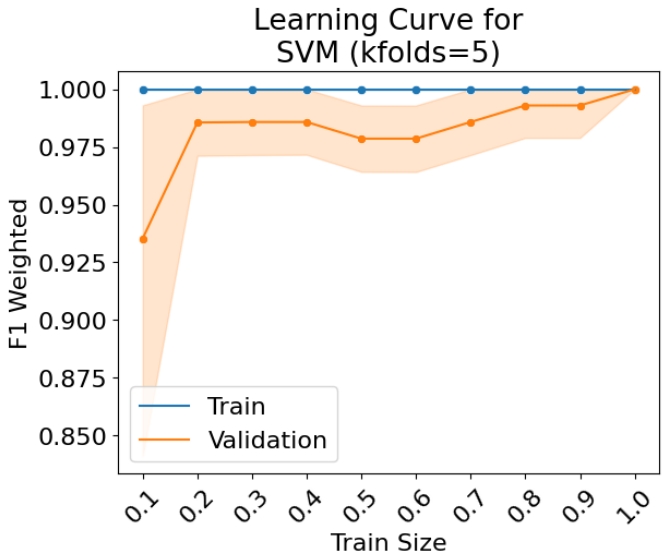


Fig. 20: Best SVM Model Learning Curve - MinMaxScaler (Wine)

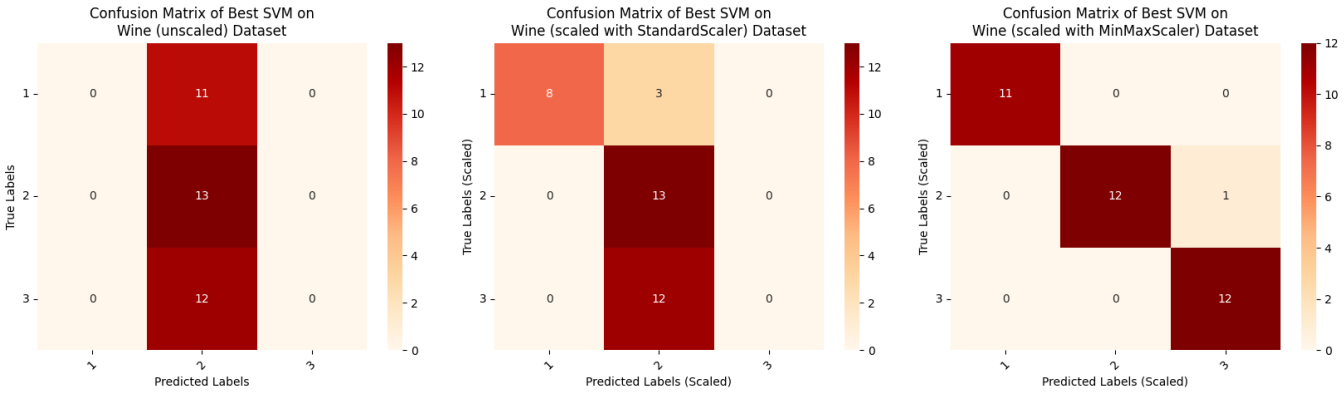
The algorithm was able to divide the dataset into its classes almost perfectly.

This is behavior is further supported by the confusion matrix of the best SVM (and best overall) model in Fig. 21c. Almost all data points in the test set were classified correctly, with one data point being the exception. Fig. 21 demonstrates the importance and effect of scaling on this dataset. Given that the dataset is separable in higher dimensions, scaling the features shrinks the space taken up by each class, making it easier to separate. Particularly the MinMaxScaler forces features to fall on a scale of 0 to 1, and quantifying physical and chemical compositions that way allows smaller class clusters to form.

B. Student Performance

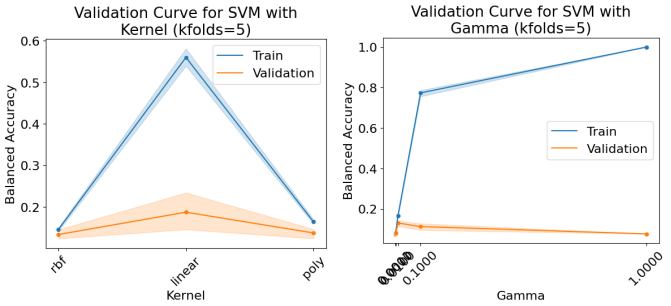
Fig. 22 shows the validation curves. Unlike those from the Wine dataset, the best kernel to use is linear, thus rendering the kernel coefficient unnecessary in tuning.

The learning curve in Fig. 23a shows that as more training samples are added, the model gets less accurate



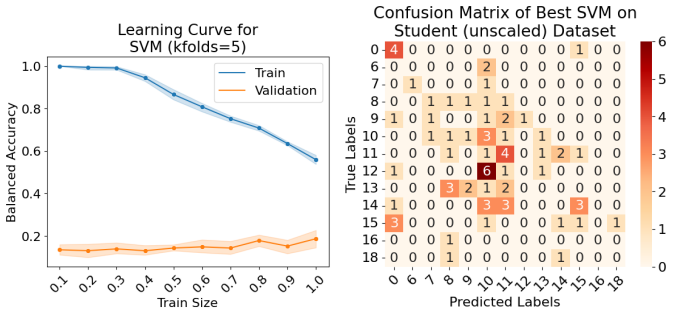
(a) No Scaling (b) StandardScaler (c) MinMaxScaler

Fig. 21: Confusion Matrix of SVM Models on Wine Database with Different Scaler Functions



(a) Kernel (b) Gamma

Fig. 22: Validation Curves on Boosted Tree Parameters (Student Performance)



(a) Learning Curves (b) Confusion Matrix (no scaler)

Fig. 23: Best SVM Performance (Student Performance)

due to the indeterminate standout features and thresholds of data points in the same class. However, as the model trains and tries to separate them, the validation set slowly improves. It's possible that SVM has room for improvement with more datapoints introduced in order to determine the more optimal split. Although not obvious, the SVM confusion matrix in Fig. 23b has the most correct true positives classified, however the Student Performance dataset did not benefit from scaling. This may be because the volume of inter-class overlap is not addressed by scaling the values.

VII. Dataset Comparison

Table I summarizes the best models across each dataset and supervised learning algorithms. It includes the performance metric – weighted F1 for Wine and balanced accuracy for Student Performance – and wall clock times for training (in seconds).

Generally, the well-behaved Wine dataset consistently has better model results across all learning algorithms, with the lowest performance metric being 0.7261 (KNN) and highest 0.9722 (SVM). The Student Performance dataset on the other hand performed poorly. The biggest culprit is the indistiguishable class clusters using the features given. Because of the volume of mixed data and under-represented classes, the lowest performance metric was obtained via the 0.1068 (KNN) algorithm and the highest 0.1424 (SVM).

TABLE I: Summary of Algorithm Performance (Avg. Score and Run Time)

Algorithm	**Wine (F1 Score)**	**Student (Bal. Acc.)**
Decision Trees	0.8059 (0.0012 sec)	0.1181 (0.0022 sec)
KNN	0.7261 (0.0007 sec)	0.1068 (0.0011 sec)
Boosting	0.9497 (0.0518 sec)	0.1075 (0.0037 sec)
Neural Networks	0.8340 (0.3485 sec)	0.1239 (0.2466 sec)
SVM	0.9722 (0.0009 sec)	0.1424 (0.0200 sec)

The fastest model to "train" is the KNN algorithm, but this is because it doesn't train in the tradition sense – it merely looks at a datapoint's nearest neighbors whose category you have, look up, and assigns it to

the new data point. The slowest is consistently the Neural Network algorithm, which makes sense due to the backpropagation, and loss computation.

VIII. Conclusion

Different datasets require different approaches and algorithms to achieve the best model – it depends on the dataset attributes and the overall goal. Given a well-behaved dataset with non-linearly separable features, SVM and Boosting works best, as shown with the Wine dataset. For an imbalanced dataset with less standout features and thresholds, a Neural Network or SVM works best. Trade-offs with learning algorithms play a part in choosing, ranging from explainability, complexity, parameters to tune, etc. It's important to experiments and create plot to ensure that the parameters chosen to model do not under or overfit on the training data.

For improved model performance, other machine learning methods can be used in conjunction with hyperparameter tuning with different algorithms. More advanced feature engineering techniques (i.e. PCA) and selection may help improve the Student Performance dataset.

References

[1] Stefan Aeberhard and M. Forina. Wine. UCI Machine Learning Repository, 1991. DOI: https://doi.org/10.24432/C5PC7J.

[2] Paulo Cortez. Student Performance. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5TG7T.

[3] Mario Filho. Does svm need feature scaling or normalization? https://forecastegy.com/posts/does-svm-need-feature-scaling-or-normalization/, 2023.

[4] Prashant Gupta. Decision trees in machine learning. https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052, 2017.

[5] Viljay Kanade. What is a support vector machine? working, types, and examples. https://towardsdatascience.com/when-not-to-use-neural-networks-a97608dbd3f6, 2022.

[6] Aditya Kumar. Knn algorithm: When? why? how? https://towardsdatascience.com/knn-algorithm-what-when-why-how-41405c16c36f, 2020.

[7] Sujatha Mudadla. Why we have to remove highly correlated features in machine learning?, 2023.

[8] Motunrayo Olugbenga. Balanced accuracy: When should you use it? https://neptune.ai/blog/balanced-accuracy, 2023.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] Anshul Saini. Adaboost algorithm: Understand, implement and master adaboost. https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/, 2024.

[11] Swastic Satpathy. Smote for imbalanced classification with python. https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/, 2023.

[12] Ygor Serpa. When not to use neural networks. https://towardsdatascience.com/when-not-to-use-neural-networks-a97608dbd3f6, 2022.