CS 7641 Machine Learning: Supervised Learning

Jilong Cui
jcui@gatech.edu

1 OVERVIEW

In this study, the performance of five essential learning algorithms is investigated across two unique datasets.

1.1 Machine learning models

The models examined include: 1) Decision Trees, 2) Neural Networks, 3) Boosting, 4) Support Vector Machines, and 5) K-Nearest Neighbors. A thorough process was undertaken to tune these models, iterating over various parameters in an attempt to find the configuration that would optimize performance for each dataset. The outcome was a set of classification analyses that highlighted both the advantages and constraints of each algorithm when applied to different data environments.

1.2 Dataset

The datasets chosen for this investigation – UFC Fight and Wine – present distinct challenges.

The **UFC dataset** comprises over 150 features and focuses on a binary classification task: predicting whether the winner is from the red or blue corner based on these features. This dataset provides a quantified representation of metrics for fighters in both corners, combining a myriad of human elements, uncertainties, and the unpredictable nature intrinsic to competitive sports. The sheer quantity of features, coupled with the depth of human elements they capture, makes this dataset particularly demanding.

Conversely, the **Wine dataset** is characterized by a more consistent landscape, anchored by 14 specific chemical and physical attributes of wines. Unlike the human-centric nature of the UFC dataset, the Wine dataset centers on the objective task of determining a wine's class based on these attributes. With three distinct classes and a significantly smaller feature set, this dataset, despite posing a multi-class classification challenge, permits a more direct analytical path. The predictability of the chemical properties within wines makes classification more reliable and logical compared to the dynamic UFC dataset.

In essence, the juxtaposition of the UFC and Wine datasets in this study provides a holistic view of the challenges and intricacies of supervised learning. Ranging from datasets with a myriad of

features to the inherent predictability or unpredictability of data and from binary to multi-class classification, this project delineates the complexities and certitudes of algorithmic interpretation across diverse datasets.

2 METHODOLOGY

The methodology employed for this project followed a structured, data-driven approach to assess and optimize the performance of classification models:

1. **Implementation of Classification Models**: Using the **sklearn** Python library, a suite of classification models was implemented.

2. **Initial Analysis**: Initially, the default model configurations were utilized to run the analysis. This provided a baseline performance measure and understanding of each model's behavior without any specific optimization.

3. **Performance Metric - F1 Score**: The F1 score was chosen as the primary metric to judge model performance. The F1 score is a harmonic mean of precision and recall. Given the nature of both datasets, balancing false positives and false negatives was crucial. Especially in scenarios where the datasets might have imbalanced classes, an F1 score is preferable over raw accuracy as it gives a more holistic view of both false positives and false negatives.

    - **UFC Dataset**: The **f1_macro** score was selected. This averages the F1 score of each class, treating them uniformly. Given the binary nature of the UFC data, where each outcome is of equal importance, a macro-average provides an unbiased metric.

    - **Wine Dataset**: The **f1_weighted** score was employed. It calculates metrics for each label and finds their average, weighted by support (the number of true instances for each label). Given the multi-class nature of the Wine dataset, and potentially differing class distributions, a weighted average considers the significance of each class based on its prevalence in the dataset.

4. **Learning Curve Visualization**: The learning curves of the models were plotted against the number of training instances and their F1 scores. This helped in providing an initial overview of each model's performance under default settings.

5. **Parameter Tuning**: To enhance the performance of the models, a grid search was executed on a predetermined set of parameters across a range of values. This systematic exploration helped identify configurations that potentially enhanced the model's predictive capabilities.

6. **Complexity Curve**: The complexity curve (referred to as the validation curve in **sklearn**) was utilized. This curve depicts model performance against varying values of specific hyperparameters, helping in understanding the effect of individual parameters on the model's overall performance.

7. **Selection of Optimal Model**: After identifying the best parameter set, the models were redefined and their learning curves re-evaluated. This ensured that the model was calibrated optimally based on the newfound insights.

8. **Performance Evaluation**: Classification metrics and confusion matrices were derived from the final model runs. The model's efficacy was then assessed based on the observed variance and bias from the learning curves and the overall F1 scores from the predictions.

3 MODEL IMPLEMENTATION

For training and evaluation purposes, the dataset was divided in an 8:2 ratio, ensuring that the class distribution in both splits mirrored the original dataset. Additionally, validation during the model tuning process was performed using 10-fold cross validation.

3.1 Decision Tree

The **DecisionTreeClassifier** from **sklearn** was employed. The primary hyperparameters under scrutiny for tuning included:

- **Split Criterion**: Determines the function to measure the quality of a split. The two methods explored were **gini** (measuring impurity) and **entropy** (measuring information gain).

- **ccp_alpha**: A parameter that determines the complexity of the tree through cost complexity pruning. A smaller value will cause the tree to grow more, whereas a larger value will prune the tree more aggressively.

- **max_depth**: Defines the maximum depth of the tree. It can be used to control over-fitting as deeper trees capture more specifics of the training data.

Figure 1 showcases analysis curves: the initial learning curve with the default model, the complexity curve in relation to **ccp_alpha** (ranging between 0 and 0.007) and **max_depth** (ranging between 1 and 20), and the final learning curve with the optimized model.

In the analysis of the UFC dataset, the default model's learning curve revealed significant variance between the training and validation sets. Through parameter tuning, primarily focusing on adjusting the **ccp_alpha** to between 0.003-0.004 and maintaining a tree depth between 3-5, this variance

was successfully reduced. Grid search further highlighted the efficacy of the **gini** criterion with these parameters, outperforming the **entropy** method. Post-tuning, especially after examining over 2000 training instances, the variance diminished substantially and the model's accuracy improved from 0.583 to 0.596. Conversely, for the Wine dataset, despite its commendable initial performance, similar parameter tuning did not lead to marked shifts in accuracy. This observation underscores the notion that the chosen hyperparameters had a diminished impact on the wine dataset, likely attributed to its consistent and inherent chemical and physical properties.

3.2 Neural Network

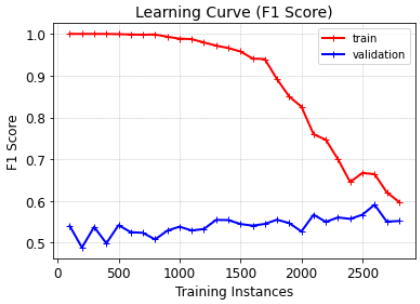
The **MLPClassifier** from **sklearn** was utilized for implementing the Neural Network. The methodology for this classifier mirrored the earlier approach. The hyperparameters subjected to tuning were:

- **hidden_layer_sizes**: This parameter determines the number of neurons in the hidden layers. Each value in the tuple represents the size of a hidden layer.
- **alpha**: Regularization parameter, also known as L2 penalty. It helps to combat overfitting by adding a penalty to the loss function.
- **max_iter**: The maximum number of iterations for the solver to converge.
- **beta_1**: Parameter for the gradient descent optimizer, specifically used for computing running averages of gradient and its square.

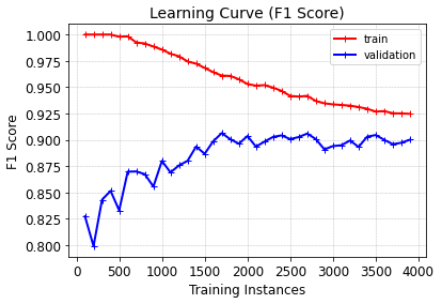
For analysis, complexity curves were plotted against **alpha** and **hidden_layer_sizes**. Figure 2 encapsulates the initial learning curve, the complexity curve, and the post-optimization learning curve. The results indicated that the **hidden_layer_sizes** had a significant influence on model performance.

For the UFC dataset, optimization of these parameters yielded a considerable improvement. While the initial model exhibited signs of overfitting, the tuned model presented a more generalized outcome. The UFC dataset saw an enhancement in accuracy from 0.582 to 0.596.

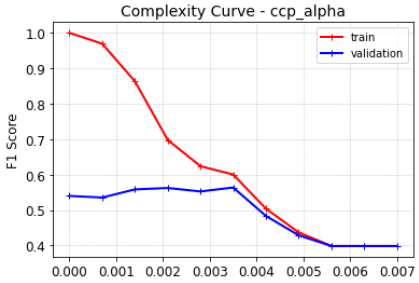
Conversely, the Wine dataset, though experiencing an improvement, did not show as pronounced a shift as the UFC dataset. Its accuracy was bolstered slightly from 0.924 to 0.931. This subtler improvement reiterates the inherent stability and consistency of the wine dataset, making it less susceptible to dramatic shifts from hyperparameter tuning.



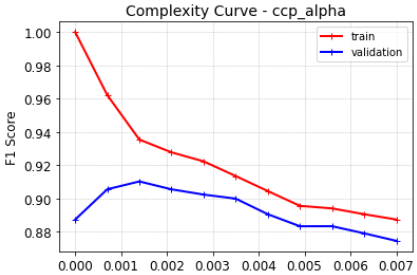
(a) UFC initial learning curve



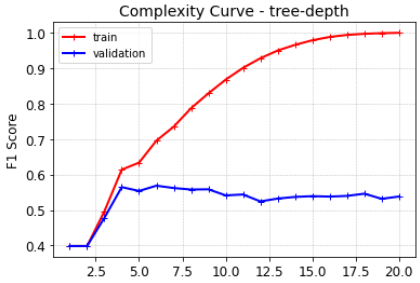
(e) Wine initial learning curve



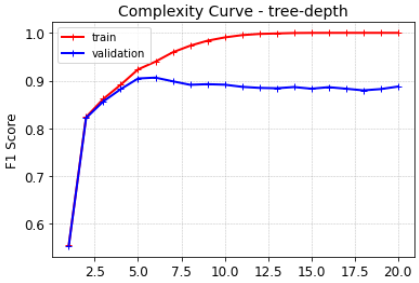
(b) UFC complexity: ccp_alpha



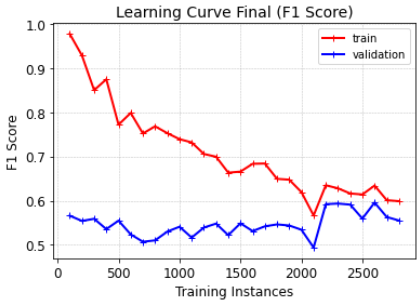
(f) Wine complexity: ccp_alpha



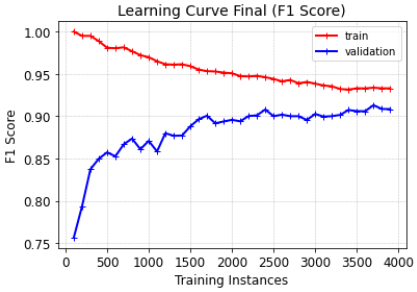
(c) UFC complexity: tree depth



(g) Wine complexity: tree depth

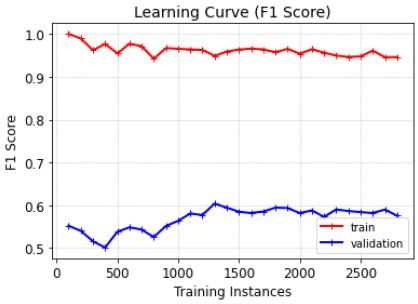


(d) UFC final learning curve

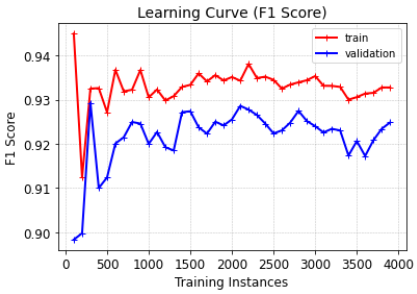


(h) Wine final learning curve

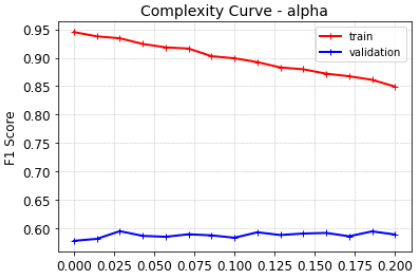
Figure 1 — Decision Tree Learning and Validation Curves.



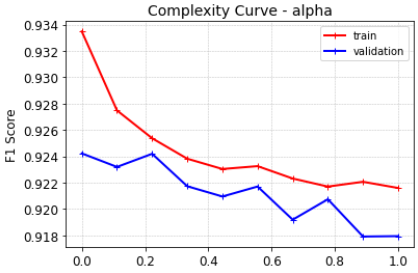
(a) UFC initial learning curve



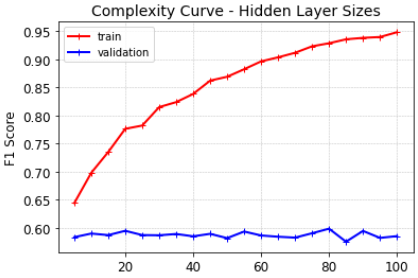
(e) Wine initial learning curve



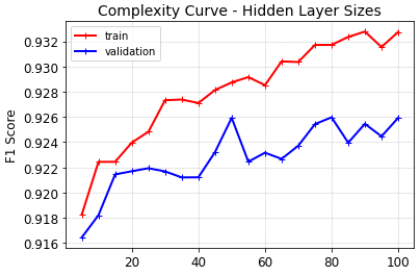
(b) UFC complexity: alpha



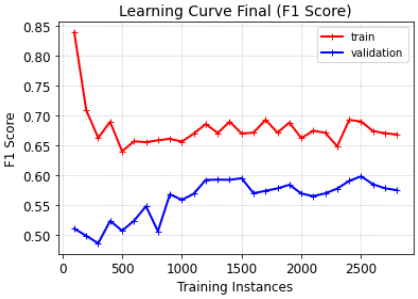
(f) Wine complexity: alpha



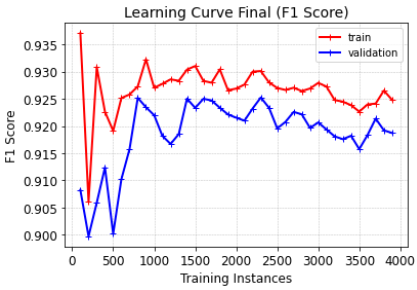
(c) UFC complexity: hidden layer size



(g) Wine complexity: hidden layer size



(d) UFC final learning curve



(h) Wine final learning curve

Figure 2 — Neural Network Learning and Validation Curves.

3.3 Boosting

The **AdaBoostClassifier** from **sklearn** was harnessed for the boosting algorithm, with the base learner set as a default tree model. The hyperparameters tuned for this classifier included:

- **base_estimator_max_depth**: Maximum depth of the individual estimator (tree).
- **n_estimators**: Specifies the number of estimators at which boosting is terminated.
- **criterion**: Measures the quality of a split.
- **learning_rate**: Governs the contribution of each classifier, shrinking the outcome of each tree before adding them up.

Complexity curves were drawn in relation to **learning_rate** and **n_estimators**, as captured in Figure 3, which also presents the learning curve.

The **learning_rate** emerged as a particularly sensitive hyperparameter. Any value above 0.2 resulted in a noticeable increase in bias for both the UFC and Wine datasets. For the UFC dataset, a higher **n_estimators** value tended to amplify overfitting.

Upon tuning, a modest improvement was achieved for the UFC dataset, while the accuracy enhancement for the Wine dataset was subtle. However, it's noteworthy that hyperparameter optimization altered the Wine dataset's learning curve in a manner more aligned with desired outcomes. Crucially, the optimized model managed to reduce variance across both datasets, signaling a more consistent and reliable model performance.

3.4 Support Vector Machine

The Support Vector Machine (SVM) was implemented using the **SVC** module from **sklearn**. The hyperparameters targeted for tuning in this classifier included:

- **Kernel**: The function used to map the data into a higher dimensional space. Different functions can achieve varied complexities of decision boundaries.
- **C**: The regularization parameter, which helps in determining the trade-off between achieving a low margin and ensuring that the data points lie on the correct side of the margin.
- **Gamma**: Governs the shape of the decision boundary. In essence, it's a parameter for the kernel, specifically for 'rbf', 'poly', and 'sigmoid'.

For the UFC dataset, the linear kernel was found to be more suitable, while for the Wine dataset, the rbf kernel was chosen. Complexity curves were drawn in relation to **C** and **gamma**, and these are displayed in Figure 4.

The SVM was observed to be quite sensitive to parameter tuning. A stark contrast was evident between the behavior exhibited by the initial and final learning curves, with the latter presenting a more harmonized pattern. Interestingly, the UFC dataset exhibited a discernible sensitivity primarily to the kernel method, rather than to other parameters. This could possibly be attributed to the nature of the data in the UFC dataset, wherein a linear decision boundary might be adequate to capture the underlying patterns without requiring further complexities.

On the other hand, the Wine dataset exhibited more varied behavior in response to hyperparameter adjustments. Notably, the training set's behavior underwent significant modification due to tuning. Naturally, the validation set also witnessed improvements. In its entirety, the SVM showcased commendable performance for both datasets, underlining its capacity as a robust classifier.

3.5 K Nearest Mean

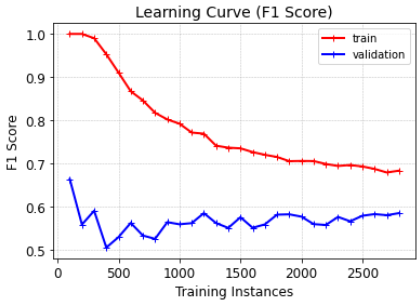
The **KNeighborsClassifier** from **sklearn** was employed, which is a staple for distance-based classification. The parameters targeted for tuning in this classifier encompassed:

- **Weights**: Determines the weight function used in prediction. Options like "uniform" assign uniform weights to all neighbors, while "distance" assigns weights proportional to the inverse of the distance from the query point.
- **n_neighbors**: Represents the number of neighbors to use for queries. It can significantly affect the boundaries around classes.
- **Leaf_size**: Affects the structure of the search tree, which is used for querying, and can impact the speed and memory required to store the classifier.

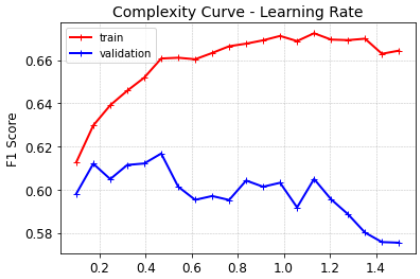
The complexity curves, visualizing the effects of **n_neighbors** and **leaf_size**, along with the learning curves are illustrated in Figure 5.

Naturally, the model is highly sensitive to the **n_neighbors** parameter. The UFC dataset demonstrates optimal performance with a smaller neighbor count of 10, while the Wine dataset favors a larger count, clocking in at 30. Uniform weights seemed to be the optimal choice for both datasets. A potential reason for this could be that the data distributions in both datasets don't require the nuanced weightings that the "distance" option would provide. Instead, a simple majority voting, which is what "uniform" facilitates, suffices.

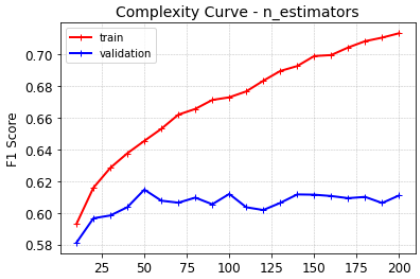
In terms of performance, while tuning managed to reduce the variance between training and validation sets, there wasn't a significant enhancement in prediction accuracy. This suggests that KNN might not be the most effective classifier for these particular datasets, possibly because it struggles to delineate the intricate boundaries or the underlying patterns inherent in the data.



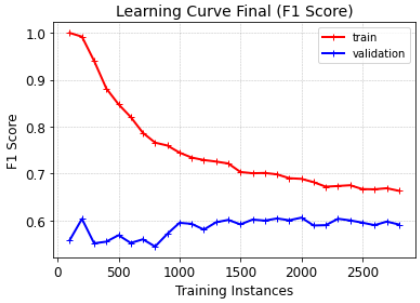
(a) UFC initial learning curve



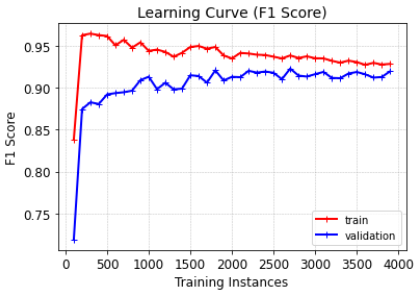
(b) UFC complexity: learning rate



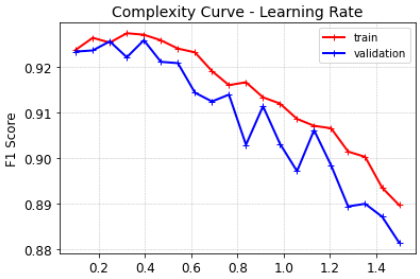
(c) UFC complexity: n_estimator



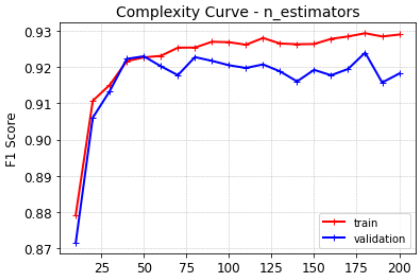
(d) UFC final learning curve



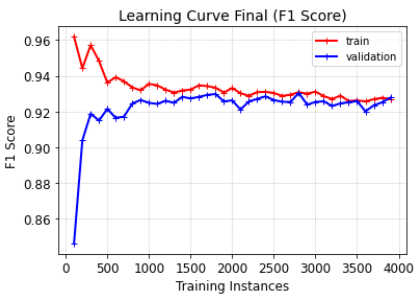
(e) Wine initial learning curve



(f) Wine complexity: learning rate

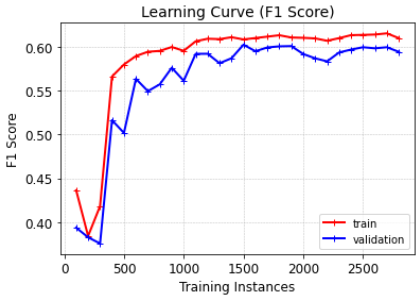


(g) Wine complexity: n_estimator

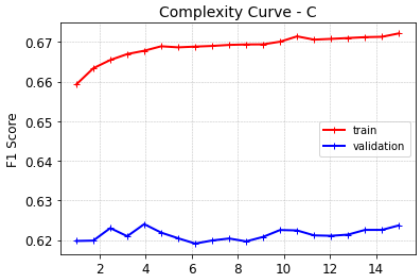


(h) Wine final learning curve

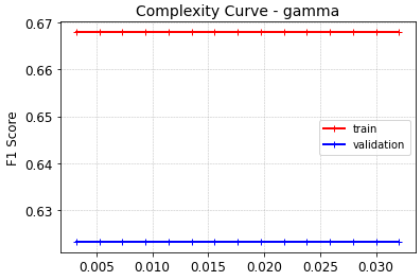
Figure 3 — Boosting Learning and Validation Curves.



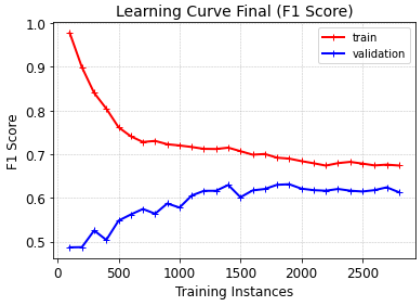
(a) UFC initial learning curve



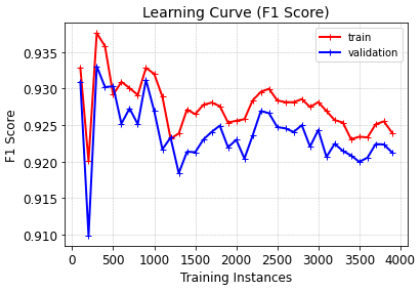
(b) UFC complexity: C



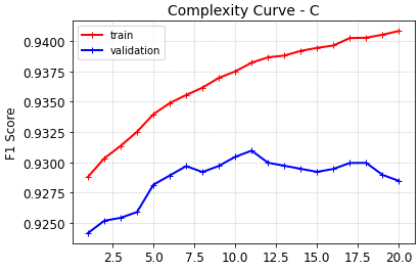
(c) UFC complexity: gamma



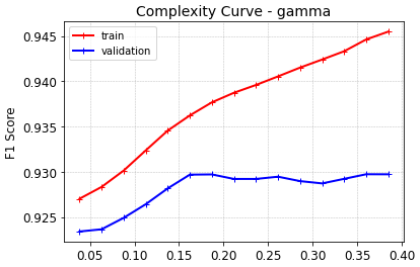
(d) UFC final learning curve



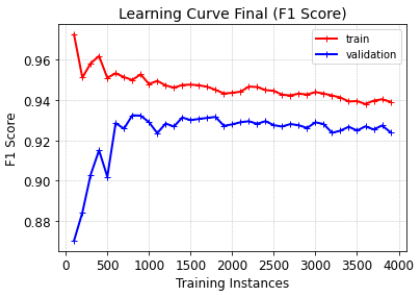
(e) Wine initial learning curve



(f) Wine complexity: C

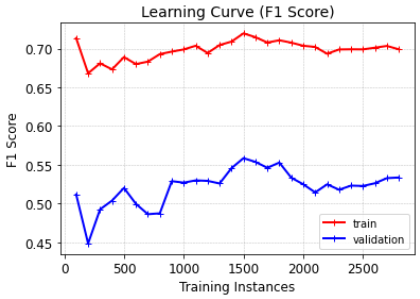


(g) Wine complexity: gamma

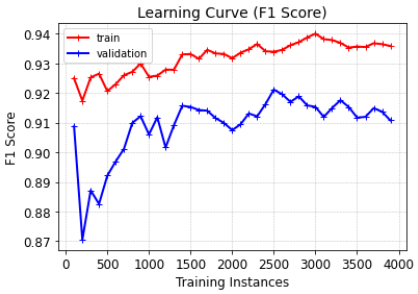


(h) Wine final learning curve

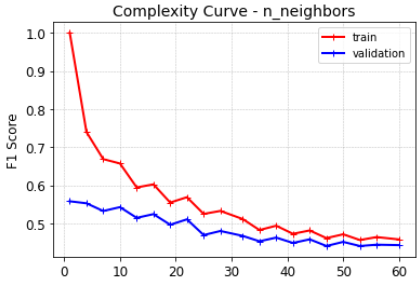
Figure 4 — Support Vector Machine (SVM) Learning and Validation Curves.



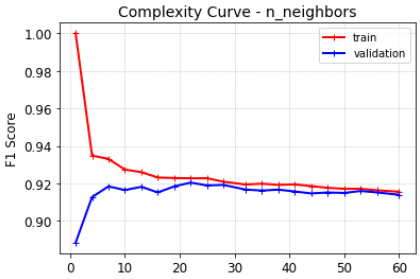
(a) UFC initial learning curve



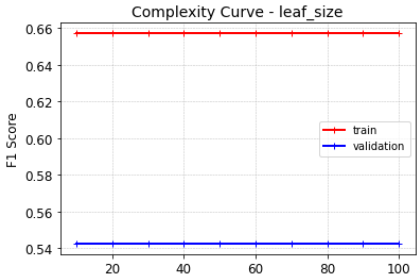
(e) Wine initial learning curve



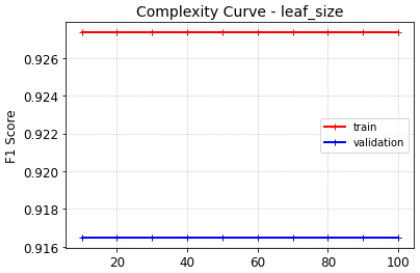
(b) UFC complexity: n_neighbors



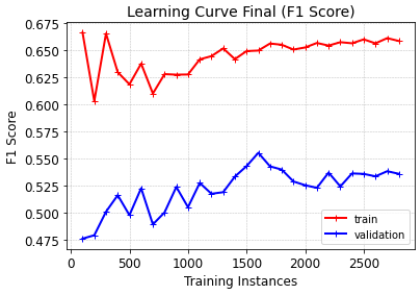
(f) Wine complexity: n_neighbors



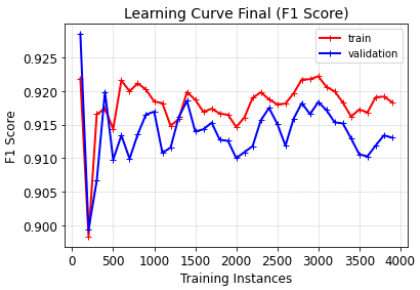
(c) UFC complexity: leaf size



(g) Wine complexity: leaf size



(d) UFC final learning curve



(h) Wine final learning curve

Figure 5 — K-Nearest Neighbor (KNN) Learning and Validation Curves.

4 SUMMARY

Table 1 summarizes the final F1 scores for the optimized models. Across the UFC and Wine datasets, model performances showed relative consistency. The Support Vector Machine (SVM) stood out, delivering top F1 scores of 0.621 for UFC and 0.935 for Wine. Its superiority can be attributed to its ability to find optimal hyperplanes in high-dimensional spaces and its flexible kernel functions, allowing it to adapt effectively to different data structures.

Boosting, using a Decision Tree base model, also performed impressively, scoring 0.618 for UFC and 0.935 for Wine. Its edge over the standalone Decision Tree (which registered 0.596 for UFC and 0.904 for Wine) highlights the strength of ensemble methods. While singular models like Decision Trees can be prone to overfitting, Boosting aggregates the power of multiple weak learners, iteratively refining predictions and achieving better generalization.

Overall, while the models exhibited comparable performances, the subtle distinctions in their behaviors and scores hint at the intricate interplay of algorithmic design, data characteristics, and hyperparameter tuning in determining the final outcomes.

Table 1 — Prediction Accuracy (F1 Scores) Summary.

UFC Dataset		
Model	**Class**	**F1 Score (marco)**
1. Decision Tree	Blue	0.432
	Red	0.761
	Total	0.596
2. Neural Network	Blue	0.411
	Red	0.782
	Total	0.597
3. Boosting	Blue	0.554
	Red	0.682
	Total	0.618
4. SVM	Blue	0.567
	Red	0.675
	Total	0.621
5. KNN	Blue	0.369
	Red	0.727
	Total	0.548

Wine Dataset		
Model	**Class**	**F1 Score (weighted)**
1. Decision Tree	A	0.913
	B	0.905
	C	0.893
	Total	0.904
2. Neural Network	A	0.926
	B	0.934
	C	0.932
	Total	0.931
3. Boosting	A	0.933
	B	0.939
	C	0.931
	Total	0.935
4. SVM	A	0.933
	B	0.940
	C	0.930
	Total	0.935
5. KNN	A	0.933
	B	0.931
	C	0.919
	Total	0.928