Report: Custom Decision Tree Classifier Implementation

36523348

2024-01-10

Abstract

This research delves into the comprehensive evaluation of a custom decision tree classifier, contrasting its outcomes with the widely-used SKlearn library implementation. Through extensive testing across diverse datasets with varied parameters, it aims to (i) assess the performance of a custom implementation of a decision tree classifier, with the SKlearn implementation, (ii) evaluate result metrics encompassing accuracy, precision, recall, F1-score, Mathew's correlation, and prediction similarity, and computational metrics including training time and memory utilization.

Comparative analysis reveals marginal differences, approximately within a 1% range, between the outcomes of the custom and SKlearn classifiers. Notably, prediction similarity stands at around 98%, indicating closely aligned performances. A detailed exploration of the impact of input parameters reveals that sample size, depth and minimum samples criteria have some impact on evaluation metrics, but a greater impact on training time and memory utilization. The robustness of findings is supported by statistical analyses throughout the study, informing the significance of observed changes. Lastly, this study brings focus to optimal configurations for custom classifiers, highlighting the delicate balance between capturing dataset nuances and preventing over fitting.

1. Introduction

Machine learning, especially the development and assessment of decision tree classifiers, has undergone substantial progress. The focus on optimizing the accuracy and efficiency of classification algorithms has become paramount within this evolving field. Review of the existing literature reveals a predominant reliance on established machine learning libraries, particularly emphasizing the widely-adopted SKlearn implementation ("Sklearn.tree.DecisionTreeClassifier" n.d.) as the benchmark for decision tree classifiers. However, not much literature exists focusing on the intricate aspects and performance associated with custom decision tree classifiers. This study explores the cross examination of custom implementations, questioning whether these alternatives can offer comparable accuracy and efficiency to widely-adopted libraries like SKlearn.

In this study, the first goal is to construct a custom decision tree classifier that is comparable in some aspects to the SKlearn implementation. Next is to perform a detailed evaluation of the custom decision tree classifier's performance in comparison to the widely-adopted SKlearn implementation. This evaluation seeks to provide a nuanced understanding of the strengths and weaknesses of this custom classifier across various datasets, and comparatively evaluate the performance against SKlearn. Lastly, we aim to identify paramaters contributing significantly to optimal configurations for classification, whilst addressing the nuances and limitations of the data.

2. Methodology

In this section, the implementation details of a custom Decision Tree Classifier are presented. Decision tree is a fundamental machine learning algorithm used for classification tasks. The code is written in Python, leveraging key libraries such as NumPy and Pandas.

Class Structure: Node and DecisionTree

The Decision Tree Classifier is implemented through two classes: `Node` and `DecisionTreeClassifier`.

Node Class: The `Node` class represents a node in the decision tree. It has attributes such as `feature_name`, `threshold`, `left`, `right`, `info_gain`, and `value`. These attributes define the structure of the decision tree node. The constructor initializes the node with the specified attributes, allowing for the definition of branching conditions and values for leaf nodes.

DecisionTreeClassifier Class: The `DecisionTreeClassifier` class serves as the main implementation of the decision tree algorithm. Key parameters such as the splitting method (`method`), minimum samples required for a split (`min_samples_split`), and maximum tree levels (`tree_levels`) can be set during initialization. The `fit` method is responsible for training the decision tree on a given dataset. It extracts column names, converts object-type columns to numeric or string types, and checks and stores categorical information for features. The `train` method recursively creates the decision tree by splitting nodes based on information gain until specified stopping conditions are met, namely the minimum splitting criteria (`min_samples_split`) and the levels of branching (`tree_levels`) provided as input. The `best_split` method finds the best split for the current dataset, and the `split` method efficiently divides the dataset based on a feature threshold, considering categorical or numeric data. Methods for calculating information gain (`information_gain`), entropy (`entropy`), and Gini index (`gini_index`) are provided to evaluate split quality. The `predict` method predicts the output for a new dataset, and the `make_prediction` method facilitates the prediction process for individual data points. Methods for printing the tree in console (`print_tree`), saving the tree to a varible and as a file (`save_tree_to_variable`) and (`print_tree_to_file`). Some other supporting methods for variable manipulation such as `convert_data_types` and `check_string_columns` methods.

Usage and Customization

The Decision Tree Classifier allows customization through parameters, enabling users to tailor the model according to their specific requirements. Users can specify the splitting method, minimum samples for a split, and the maximum levels of branching. It can handle datasets of all types, containing numeric and character variables and multiclass features.

Information Gain Calculation

Information gain is a key concept used in decision tree algorithms to quantify the effectiveness of a particular feature in splitting a dataset. It is implemented in the context of entropy (Shannon, n.d.) or Gini impurity (J. R. Quinlan 1986). The calculation of information gain with reference to the entropy-based and Gini-Index approach are detailed ahead, they were extended for multi-class problems.

Entropy: Entropy is a measure of impurity or disorder in a dataset. For a binary classification problem, the entropy of a dataset S is defined as:

$$\mathit{Entropy}(S) = -p_+ \cdot \log_2(p_+) - p_- \cdot \log_2(p_-)$$

where: - p_+ is the proportion of positive instances in S. - p_- is the proportion of negative instances in S. - $\log_2$ is the logarithm to the base 2.

Information Gain with Entropy: Information gain quantifies the reduction in entropy achieved by splitting a dataset based on a particular feature. It is calculated as the difference between the entropy of the parent dataset S and the weighted sum of entropies of the child datasets resulting from the split. The formula for information gain is as follows:

$$\text{Information Gain} = \text{Entropy}(S) - \left(\frac{|S_{\text{left}}|}{|S|} \cdot \text{Entropy}(S_{\text{left}}) + \frac{|S_{\text{right}}|}{|S|} \cdot \text{Entropy}(S_{\text{right}}) \right)$$

where: - S_{left} and S_{right} are the datasets resulting from the split. - $|S|$ is the total number of instances in the parent dataset. - $|S_{\text{left}}|$ and $|S_{\text{right}}|$ are the number of instances in the left and right child datasets, respectively, when divided with the total instances $|S|$, they form the weight of the respective subset.

Higher information gain indicates a more effective split, as it signifies a greater reduction in entropy and, consequently, increased purity or homogeneity in the resulting subsets. Decision tree algorithm implemented finds the feature and threshold that maximizes information gain during the tree-building process. This leads to the creation of subsets with more homogeneous class distributions, ultimately improving the accuracy of the model.

Gini Impurity

Gini impurity is an alternative measure of calculating impurity commonly used in decision tree algorithms. For custom classification tree, the Gini impurity of a dataset S is calculated as:

$$Gini(S) = 1 - \sum_{i=1}^{c} p_i^2$$

where: - c is the number of classes in the dataset. - p_i is the proportion of instances belonging to class i.

Information Gain with Gini Impurity

In the decision tree algorithm, information gain is also calculated using Gini impurity as the impurity measure. The formula for information gain with Gini impurity is given by:

$$\text{Information Gain (Gini)} = \text{Gini}(S) - \left(\frac{|S_{\text{left}}|}{|S|} \cdot \text{Gini}(S_{\text{left}}) + \frac{|S_{\text{right}}|}{|S|} \cdot \text{Gini}(S_{\text{right}}) \right)$$

where: - S_{left} and S_{right} are the datasets resulting from the split. - $|S|$ is the total number of instances in the parent dataset. - $|S_{\text{left}}|$ and $|S_{\text{right}}|$ are the number of instances in the left and right child datasets, respectively, when divided with the total instances $|S|$, they form the weight of the respective subset.

It quantifies the reduction in Gini impurity achieved by splitting the dataset based on a specific feature. Higher information gain with Gini impurity indicates a more effective split, leading to subsets with increased homogeneity.

Evaluation against SKlearn Decision Tree CLassifier

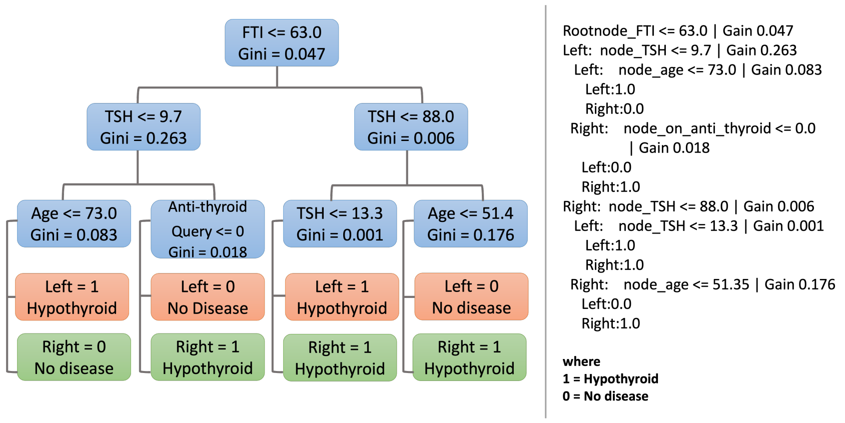
The performance of our custom decision tree implementation was assessed by comparing it to the pre-built decision tree classifier provided by the SKlearn library ("Sklearn.tree.DecisionTreeClassifier" n.d.). Two sets of results were obtained to facilitate this comparison:

Evaluation Metrics: Accuracy, F1-score, Precision, Recall, Matthews Correlation Coefficient, and Prediction Similarity. Prediction Similarity: This metric gauges the agreement between the predicted classes of our custom classifier and those generated by SKlearn.

Computational Metrics: Training Time Difference: Quantifies the variance in training times between our custom classifier and the SKlearn implementation. Memory Utilization: Examines the disparity in memory usage during the training phase between the two classifiers. These comprehensive sets of metrics provide a robust basis for comparing the effectiveness and efficiency of our custom decision tree implementation against the widely-used SKlearn library.

Basic Tree Structure: For comparison, Hypothyroid dataset was used to train a decision tree through both libraries, and plotted or printed for comparative visualization. Custom decision tree operated in a relatively similar manner forming similar root and leaf nodes (small error range, analyzed in detail in Results section). The decision tree at minimum samples = 3 and tree depth = 3, is shown in the figure below.

Limitations: While the custom decision tree implementation shares common parameters with SKlearn, such as the Information Gain method (Entropy or Gini Index), minimum samples for splitting, and maximum levels of branches, it has some limitations in flexibility, compared to the SKlearn library. Custom Implementation operates on a 'best fit' protocol, where it tests all possible splits to find the optimal one. Whereas, SKlearn Library offers greater flexibility with advanced parameters, including but not limited to pruning, random permutations for splitting, and complexity cost parameters. These



```
Rootnode_FTI <= 63.0 | Gain 0.047
Left:  node_TSH <= 9.7 | Gain 0.263
  Left:    node_age <= 73.0 | Gain 0.083
     Left:1.0
     Right:0.0
  Right:    node_on_anti_thyroid <= 0.0
                | Gain 0.018
    Left:0.0
    Right:1.0
Right:  node_TSH <= 88.0 | Gain 0.006
  Left:    node_TSH <= 13.3 | Gain 0.001
     Left:1.0
     Right:1.0
  Right:    node_age <= 51.35 | Gain 0.176
    Left:0.0
    Right:1.0

where
1 = Hypothyroid
0 = No disease
```

Figure 1: Hypothyroid data Decision Tree by Custom Classifier. (Left) Tree Structure (Right) Tree saved as output in text file

advanced features are not integrated into our custom implementation. Custom Implementation lacks certain advanced parameters available in the SKlearn Decision Tree classifier such as the fine-tuning of the decision tree model using pruning and complexity parameters, allowing for a more detailed and customized configuration. To ensure a fair comparative evaluation, the SKlearn classifier was utilized with a 'best split' setting and without additional parameters.

3. Results and Discussion

In this section, we comprehensively assess the implementation and outcomes of the custom decision tree classifier, in comparison to the decision tree classifier from the SKlearn library. The classifiers underwent testing across multiple datasets with varied parameters to ensure a thorough examination.

3.1. Evaluation summary

The table below provides a summary of evaluation metrics associated with each dataset, along with their respective attributes and hyperparameters. Four datasets were employed, each at different sizes: 25%, 50%, 75%, and 100% of samples. The tree branching depths were set at 2, 4, 8, and 16, and the minimum sample sizes for splitting were configured at 3, 15, 50, 100 and 500. The datasets used for evaluation are as follows:

1. Wine Quality Dataset: Represents a business-oriented objective of identifying high-quality wines for purchase. This dataset includes only quantitative features (features: 12, samples: ~6500) (Paulo Cortez 2009).
2. Customer Churn Dataset: Represents a business cycle approach to customer retention, encompassing both qualitative and quantitative features (features: 14, samples: ~3100) (Unknown 2020).
3. Hepatitis-C Dataset: Applied for a medical/biological context to predict disease outcomes based on test readings and other factors. It includes qualitative, categorical, and quantitative features (features: 29, samples: ~1400) (Sanaa Kamal 2017).
4. Hypothyroid Dataset: Employed for a medical/biological context in predicting disease outcomes based on test readings and other factors. It comprises qualitative, categorical, and quantitative features (features: 18, samples: ~3100) (R. Quinlan 1986).

Maintaining a variety in datasets aims to mitigate field-based evaluation bias, facilitating assessment based on differing complexities and sizes. Evaluation metrics encompassed accuracy, precision, recall, F1-score, Mathew's correlation, and prediction similarity of predicted classes on test data. Computational metrics assessed included the training time difference between the custom classifier and the SKlearn classifier, along with memory utilization. A summarized version of evaluation metrics for the results of the custom implementation of the decision tree are presented in Table-1.

Table 1: Summary of Custom Decision Tree Metrics (depth = 8, minimum samples for split = 3)

method	Dataset	Variables	Samples	Features	Accuracy	F1score
entropy	Wine	numeric	1624	12	0.535	0.496
gini	Wine	numeric	1624	12	0.531	0.491
entropy	CustomerChurn	mixed	787	14	0.903	0.888
gini	CustomerChurn	mixed	787	14	0.908	0.899
entropy	HCV	mixed	346	29	0.238	0.189
gini	HCV	mixed	346	29	0.240	0.198
entropy	Hypothyroid	mixed	790	18	0.974	0.975
gini	Hypothyroid	mixed	790	18	0.975	0.975

Results indicate that broadly, evaluation metrics are not dependent upon the variable types (quantitative or categorical), data set attributes of size or the information gain method utilized (gini index or entropy). Rather they represent the ability of the features contained withing to be appropriate enough to allow useful classification through decision trees. Each aspect has been investigated in more detail in later sections.

3.2. Detailed Comparison of Differences in Evaluation Results

A comprehensive evaluation of the disparity in results between the custom decision tree classifier and SKlearn library's classifier is compiled below. This analysis aims to assess the performance across various evaluation and computational metrics, including accuracy, precision, recall, F1-score, prediction similarity, Matthew's correlation, training time, and memory utilization. The differences between the custom decision tree and SKlearn decision tree classifiers are presented with respect to different groups of inputs, shedding light on performance variations based on input attributes and classifier parameters.

1. Tree Depth:: Tree depths were tested at four different levels, and the results are summarized in the table below. On average, a negligible difference in accuracy, precision, recall, F1-score and Matthew's correlation coefficient (MCC) is observed. The training time and memory increase with depth, with an average difference of 2.927seconds more and -0.091MiB less memory utilized by custom classifier. The time increase accumulates with growing complexity, reaching up to approximately 5 seconds. Prediction similarity decreases slightly with increase in depth.

The findings suggest that the depth has a minimal impact on the evaluation metrics of accuracy and similarity between the two classifiers. However, as a measure of complexity, increasing depth contributes to a continuous rise in training time and memory utilization. This aligns with expectations, as computational metrics are inherently influenced by classification complexity.

The observed decrease in prediction similarity with increased depth is anticipated to be a consequence of overfitting in classifiers. This decrease in generalizability is directly reflected in the altered results, indicating the need for cautious consideration of depth parameters to prevent overfitting and ensure robust model performance.

Table 2: Difference in Evaluation Metrics Across Tree Depths: Table highlighting differences in results, Negative values indicate comparativelyy lesser units by custom classifier, positive values indicates greater units taken by custom classifier

depth	Accuracy	Precision	Recall	F1score	MCC	Time_secs	Memory_MiB	Similarity
2	0.000	0.002	0.000	0.000	0.003	1.059	-0.553	0.999
4	0.001	0.002	0.001	0.001	0.003	2.015	-0.192	0.980
8	0.000	-0.001	0.000	-0.001	0.001	3.642	0.148	0.985
16	-0.001	-0.001	-0.001	-0.001	0.000	4.991	0.233	0.967

2. Minimum samples criteria:: Minimum samples required for branching were tested at 6 different levels and the results are summarized in the table below. The relation of complexity and overfitting with

Table 3: Average of Difference in metrics across Tree Depths: Table highlighting differences in results, Negative values indicate comparativelyy lesser units by custom classifier, positive values indicates greater units taken by custom classifier.

	Accuracy	Precision	Recall	F1score	MCC	Time_secs	Memory_MiB	Similarity
Average	0	0.001	0	0	0.002	2.927	-0.091	0.983

evaluation results is similar to that seen in tree depths. The time increase accumulates with growing complexity (decreasing minimum samples input), reaching up to approximately 4 seconds. Prediction similarity decreases slightly with increase in minimum samples criteria.

The findings suggest that sample size of branching has a minimal impact on the evaluation metrics of accuracy and similarity between the two classifiers. However, as a measure of complexity, decreasing criteria contributes to a continuous rise in training time and memory utilization. This aligns with expectations, as computational metrics are inherently influenced by classification complexity.

The observed decrease in prediction similarity with increased criteria is anticipated to be a consequence of overfitting in classifiers. This decrease in generalizability is directly reflected in the altered results.

Table 4: Difference in Evaluation Metrics Across Minimum Samples for Branching

min_samples	Accuracy	Precision	Recall	F1score	MCC	Time_secs	Memory_MiB	Similarity
3	0.000	0.000	0.000	0.000	-0.001	4.187	0.145	0.953
15	0.000	0.000	0.000	0.000	0.002	3.746	0.112	0.977
50	-0.001	-0.001	-0.001	-0.001	0.002	3.107	-0.278	0.986
100	0.001	0.002	0.001	0.001	0.004	2.701	0.276	0.992
200	0.001	0.002	0.001	0.000	0.002	2.281	-0.298	0.992
500	0.001	0.002	0.001	0.001	0.002	1.540	-0.503	0.996

Table 5: Average of Difference in Evaluation Metrics Across Minimum Samples for Branching

	Accuracy	Precision	Recall	F1score	MCC	Time_secs	Memory_MiB	Similarity
Average	0	0.001	0	0	0.002	2.927	-0.091	0.983

3.3. Detailed Exploration of Results

A meticulous examination of the results from the custom classifier is undertaken to discern patterns and trends arising from dataset attributes and classification parameters. This detailed exploration aims to unravel nuanced insights into the performance of the classifier under varying conditions.

1. Performance across Sample Sizes: The classifier was executed at four distinct sample sizes from each dataset (25%, 50%, 75%, and 100%). A subset of results for each size, across wine quality dataset is shown in the table below. Additionally, a graphical representation of the results is presented in Figure 2. Accuracy and F1-score are plotted from the evaluation metrics, considering F1-score is a harmonic mean of precision and recall, thus encapsulating both aspects. Time is plotted from the computational metrics. Together, the visual representation in the plots offers a scaled summary of the entire evaluation metrics panel, showcasing their variations with an increase in complexity introduced by sample size

While the evaluation metrics exhibit a general increasing trend with the expansion of the training data obtained from samples, a plateau or peak is observed, before falling as well. Time and memory display an anticipated increase with the augmented complexity introduced by dataset size.

The observed peaks in accuracy for the HCV and Hypothyroid datasets at 75% and 50% of sample sizes, respectively, may indicate an optimal training ratio. Beyond this point, overfitting could potentially occur, leading to a decline in the performance of the training model on the test set. Similarly, the

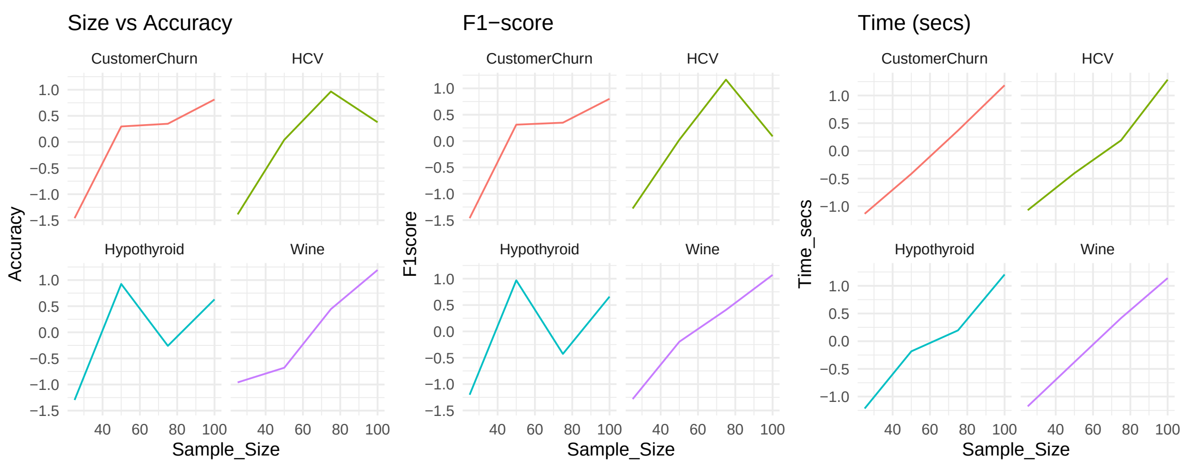


Figure 2: Size vs. Accuracy, F1score and Time by Dataset (Standardized)

plateau in the Customer Churn dataset at 50% to 75% exemplifies an optimal dataset size for achieving good performance, beyond which further increases in size may have reduced or diminishing impacts on accuracy.

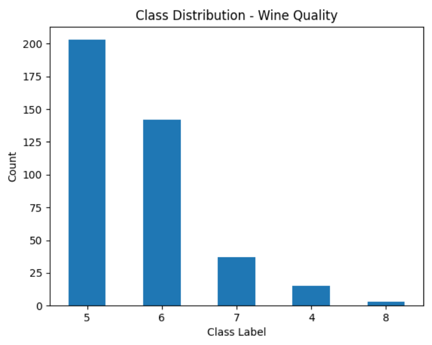
In practical applications, obtaining and managing large datasets can be challenging. Therefore, identifying a suitable amount of data along with an appropriate training-test ratio becomes crucial for achieving reasonable accuracy in real-world scenarios. These findings underscore the importance of carefully balancing dataset size and training ratios to optimize classifier performance.

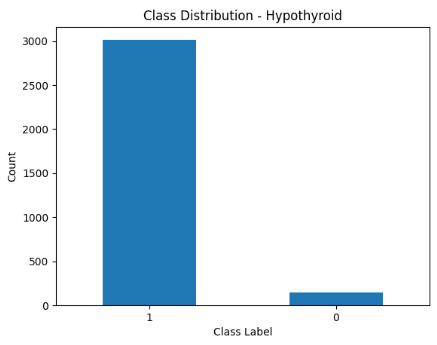
Table 6: Detailed Evaluation of Metrics Across Sample Sizes

Sample_size	Dataset	Features	Accuracy	F1score	MCC	Time	Memory	Similarity
25	Wine	12	0.523	0.472	0.189	2.163	53.499	0.983
50	Wine	12	0.526	0.490	0.206	3.895	55.263	0.983
75	Wine	12	0.538	0.500	0.198	5.621	60.128	0.984
100	Wine	12	0.546	0.511	0.204	7.192	58.873	0.987

2. Depth vs Minimum samples for splitting: The classifier was executed at six minimum samples criteria (3, 15, 50, 100, 200, 500), across different input attributes (sample size, infromation gain methodology and depths criteria). In this section we will focus on the depth vs minimum samples criteria for optimal classification setting, and it impacts the evaluation metrics. To gain insights, a graphical representation of the results is presented in the figures below. Accuracy and F1-score are plotted from the evaluation metrics (F1-score encapsulating precision and recall). Here we take MCC into account as well, and Time is plotted from the computational metrics. Together, they provide a visual summary of the evaluation metrics panel. MCC (Matthew Correlation Coefficient), also known as Phi Coefficient is a measure of how closely related two variables are. For multiclass and imbalanced class problems, MCC is proposed to be a better evaluation measure than accuracy, precision, recall or F1 score.

Wine Data Set: An intriguing trend is noted in the wine dataset, where a decline in accuracy is observed with increasing depths. We can observe the start contrast between an over-simplified model vs an overfit one. At the lowest depth of 2, minimum samples will produce no effect due to over simplification. And although it might retain an accuracy measure without change, the F1-score and MCC are low. Upon closer examination, this trend can be rationalized through an exploration of the dataset attributes. The decrease in accuracy, coupled with a simultaneous rise in the F1-score, is attributed to a shift in the balance between true positives (TP), false positives (FP), and false negatives (FN) in the classification results. This is because the weighted F1-score, considering both precision and recall, proves to be less sensitive to class imbalances compared to accuracy. Given the strong class imbalance in the wine quality dataset, as depicted in the accompanying figure, changes in classification performance are more prominently reflected in metrics like accuracy. A decrease in accuracy suggests that the model encounters challenges with the majority class, while the F1-score, with its weighted calculation that accounts for class imbalances, indicates that the model optimizes in terms of capturing the minority class, hence yielding a higher F1 score.





HCV Dataset: In the HCV dataset, although class labels are balanced, a similar discrepancy between accuracy and F1-score is observed. The overall low accuracy for this dataset (average = 0.24) implies that increasing the depth leads to an improved balance between TP, FP, and FN in the classification results, albeit at the cost of a slight decrease in accuracy. This phenomenon can be observed in the figures, where low complexity in parameters increases the accuracy but decreases F1-score, affecting the TP, FP and FN balance. It is essential to note that the standardization applied visually exaggerates the actual decline in accuracy (from 0.24 to 0.23), included to better observe small changes in data.

Customer Churn and Hypothyroid Dataset: For these datasets, an optimal complexity setting is identified, beyond which accuracy and F1-score either decline sharply or plateau. The trade off between complexity and results of clasification is clearly visibl. The appropriate classification conditions for these dataset will include a trade off in evaluation measures for generalization, towards better performance on external datasets. The robustness of our conclusion regarding the Hypothyroid dataset is further substantiated by observing the change in the time required for decision tree training. The time metric also plateaus at the parameter following 4, reinforcing our earlier conclusion that the model is likely overfit at this point. This plateau suggests that the model has fully branched and does not necessitate additional time to increase complexity. The synchronized behavior of both accuracy-related metrics and computational metrics highlights the importance of the chosen depth parameter in achieving a well-balanced and efficient model for the given dataset.

Time: The robustness of our conclusion regarding these datasets is corraborated by observing the change in the time required for decision tree training. The time metric also plateaus after the optimal complexity parameters, in the case of Hypothyroid dataset, reinforcing our earlier conclusion that the model is likely overfit at this point. In other datasets, the complexity directly affects the time utilized, creating a linear increase. The synchronized behavior of both accuracy-related metrics and computational metrics highlights the importance of the chosen depth parameter in achieving a well-balanced and efficient model for the given dataset.

3. Paired T-test for Assessing Significant Change

In the preceding sections, we observed the change of evaluation metrics in response to alterations in classification input attributes. Although adjustments in attributes such as depth, sample size, and minimum branching did induce some discernible changes, our focus now shifts to ascertaining the statistical significance of these variations within the sampled data.

To conduct a comprehensive analysis, we examined multiple datasets, summarizing the outcomes for each input attribute, including sample sizes, information gain methods, tree depths, and minimum samples for branching. Our evaluation encompassed changes in Accuracy, F1-score, and Time utilization. Employing a paired t-test adjusted at a 95% confidence interval was deemed appropriate, given that the datasets remained constant while only the input attributes were systematically modified for repeated testing. This established a paired basis for the test.

Surprisingly, none of the observed changes exhibited statistical significance in this paired data setting (number of significant tests = 0). A concise presentation of the results is provided in the table below.

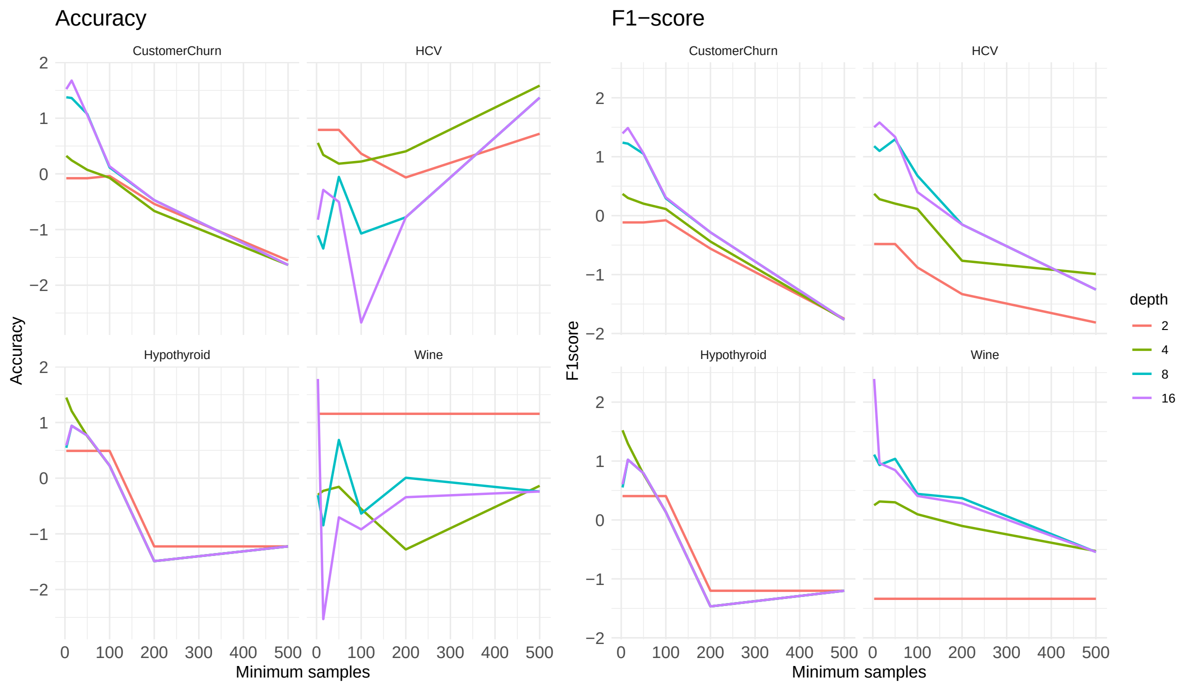


Figure 3: Minimum Samples vs. Accuracy, F1score and Time (Standardized)

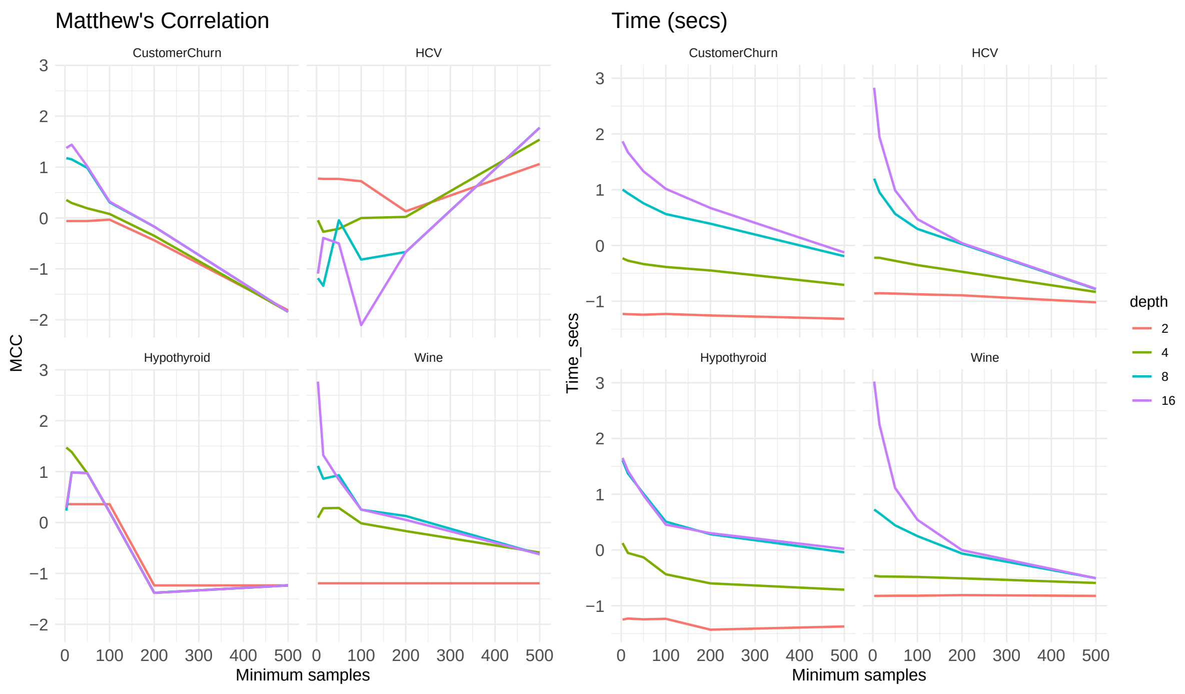


Figure 4: Minimum Samples vs. Accuracy, F1score and Time (Standardized)

Table 7: T-test results accross HCV Dataset for Accuracy per Sample size (Wine Quality Dataset)

feature	group1	group2	confidence	hypothesis	mean_group1	mean_group2
accuracy	25	50	0.95	Null hypothesis not rejected	0.523	0.526
accuracy	50	75	0.95	Null hypothesis not rejected	0.526	0.538
accuracy	75	100	0.95	Null hypothesis not rejected	0.538	0.546
f1score	25	50	0.95	Null hypothesis not rejected	0.472	0.490
f1score	50	75	0.95	Null hypothesis not rejected	0.490	0.500
f1score	75	100	0.95	Null hypothesis not rejected	0.500	0.511
time_diff	25	50	0.95	Null hypothesis not rejected	2.163	3.895
time_diff	50	75	0.95	Null hypothesis not rejected	3.895	5.621
time_diff	75	100	0.95	Null hypothesis not rejected	5.621	7.192

3.4. Repeated Measures ANOVA for Assessing Significant Change

In the previous analysis, paired T-test setting did not reveal statistical significance in the observed changes (number of significant tests = r significant_tests). One plausible explanation is that summarizing across a single feature, such as depth, might inadvertently incorporate confounding variations from other features like minimum samples or sample size. A more exhaustive approach, involving separation across groups in all input parameters and testing against the permutations created, would be too extensive.

An alternative strategy involves testing for significant changes in the paired-samples using Repeated Measures ANOVA. This approach simplifies the process by automating sample recognition and assessing changes across groups. For paired or repeated measures data, where each dataset is measured multiple times under different conditions, a Repeated Measures ANOVA is a suitable statistical method. Repeated Measures ANOVA is similar to a simple multivariate design, where the same participants are measured repeatedly, yet with variations in conditions for the same characteristic ("Repeated Measures ANOVA - Understanding a Repeated Measures ANOVA | Laerd Statistics" n.d.; "ANOVA Test: Definition, Types, Examples, SPSS" n.d.). This method offers a more comprehensive evaluation of the dataset, capturing complex interactions between different parameters and their collective impact on the observed changes.

Changes in Accuracy : Results detailed in the table below, indicate that the only significant parameter is Sample size (p-value 0.03) explaining 47% variance in accuracy data. While not significant enough, it is interesting to note minimum samples criteria can potentially explain around 12.1% of variance in the accuracy data. Although some parameters are not significant enough to be reliable, however, they give an estimate of variation that maybe explainable and will be explored later in 'Regression').

Table 8: Repeated Measures ANOVA across Input Parameters for Accuracy

Effect	ges	p	p<.05
Sample_size	47.02	0.03	*
min_samples	12.12	0.41	
depth:Sample_size	5.06	0.21	
min_samples:depth:Sample_size	1.64	0.33	

Changes in F1-score : Results detailed in the table below, indicate that the only significant parameter is Sample size (p-value 0.04) explaining 47.6% variance in F1-score data. While not significant enough, it is interesting to note minimum samples criteria and depth can potentially explain around 47.6% and 5.1% of variance in F1 data respectively. Although some parameters are not significant enough to be reliable, however, they give an estimate of variation that maybe explainable and will can be explored later.

Changes in Time : Results detailed in the table below, indicate that no attribute is a significant parameter for time changes. While not significant enough, it is useful to note that depth and sample size can potentially explain around 42.1% and 12.1% of variance in time data respectively. Although some parameters are not significant enough to be reliable, however, they give an estimate of variation that maybe explainable and will can be explored later.

Table 9: Repeated Measures ANOVA across Input Parameters for F1-score

Effect	ges	p	p<.05
Sample_size	47.60	0.04	*
min_samples	47.56	0.07	
depth	24.70	0.06	
depth:Sample_size	10.04	0.15	

Table 10: Repeated Measures ANOVA across Input Parameters for Time

Effect	ges	p	p<.05
depth	42.09	0.10	
Sample_size	37.53	0.06	
min_samples	17.70	0.11	
depth:Sample_size	15.39	0.10	

3.5. Explaining Relationships Through Regression

In the previous sections, we have focused on identifying input attributes having considerable impact on evaluation metrics. In this section, we try to identify the strength and direction of that impact. The main goal of regression analysis is to understand the relationship between the dependent variable (target) and one or more independent variables (features or predictors). The coefficients in the regression equation represent the strength and direction of the relationships between the variables. While we cannot look at all the relationships in details, lets focus on two representative datasets (i) Low accuracy data (Wine Quality) and (ii) High accuracy data (Customer Churn). These datasets were analyzed for the dependency of accuracy and F1-score on input attributes such as Sample sizes, Information gain methodology, minimum samples and tree depth inputs.

First, the influential or important features among all the input attributes were determined for accuracy and F1-score, through backward selection techniques. Metrics considered were RMSE, r squared and P-value. Once the attributes were selected, a general linear model was created and visualized.

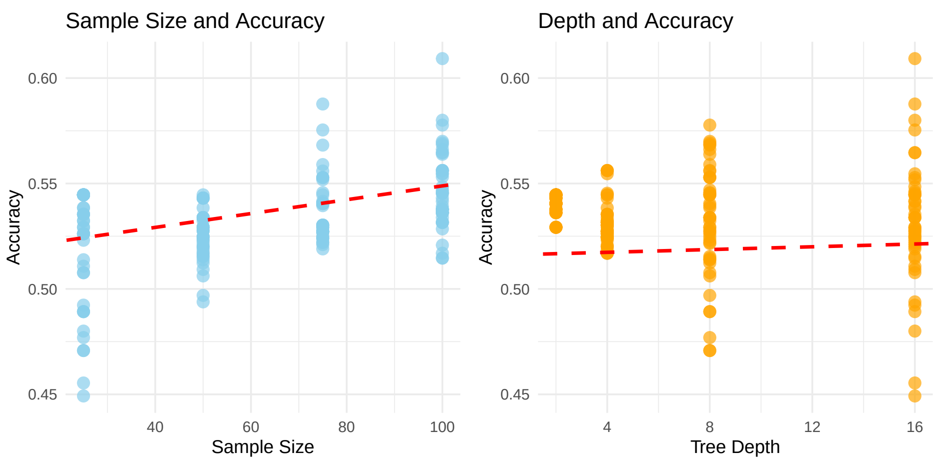


Figure 5: Depth and Sample size regressed on Accuracy

Regression analysis of Accuracy: Regression analysis was conducted to examine the impact of input attributes on model accuracy. Subset selection showed Sample size and depth as key covariates, although Information Gain method 'gini' was highlighted, the observed change in Root Mean Squared Error (RMSE) did not justify an increase in model complexity.

The resulting model accounted for approximately 21% of the variance in the data (Rsquared: 0.2070558), with an associated RMSE of 0.0184911. The relatively low variance explained can be attributed to two primary factors:

1. Small Sample Size: The limited size of the sample did not encompass enough variance to identify a robust trend.
2. Minimal Shift in Accuracy: In a broader context, the marginal change in accuracy, as evidenced in the T-test section, indicates that accuracy is not strongly influenced by the considered input attributes.

These factors collectively contribute to the modest impact of Sample Size and Depth on the model's accuracy. The observed coefficients (Sample size:3.282265×10^{-4} and Depth: $-4.2189242 \times 10^{-4}$) reveal that Sample Size has a positive impact on the unit change in accuracy, while Depth has a negative impact. Although this might initially appear counter intuitive, our earlier exploratory analysis of the data shed light on the fact that accuracy was an unweighted metric. In datasets characterized by imbalanced classes, accuracy tends to improve with a decrease in model complexity. Further the p-value indicates that the effect of sample size (p-value $1.1882556 \times 10^{-10}$) is significant and considerable, while depth does not significantly impact accuracy (p-value 0.0942485)

These results align with our prior expectations regarding the relationship within these datasets. The positive influence of Sample Size on accuracy is consistent with the notion that larger sample sizes contribute positively to model performance. Simultaneously, the negative impact of Depth suggests that reducing the complexity of the model enhances accuracy, particularly in situations with imbalanced class distributions.

Regression analysis of F1-score: Regression analysis was conducted to examine the impact of key input attributes on model F1-score of Customer Churn dataset. The key covariates identified through subset selection were Minimum samples, samples size and tree depth.

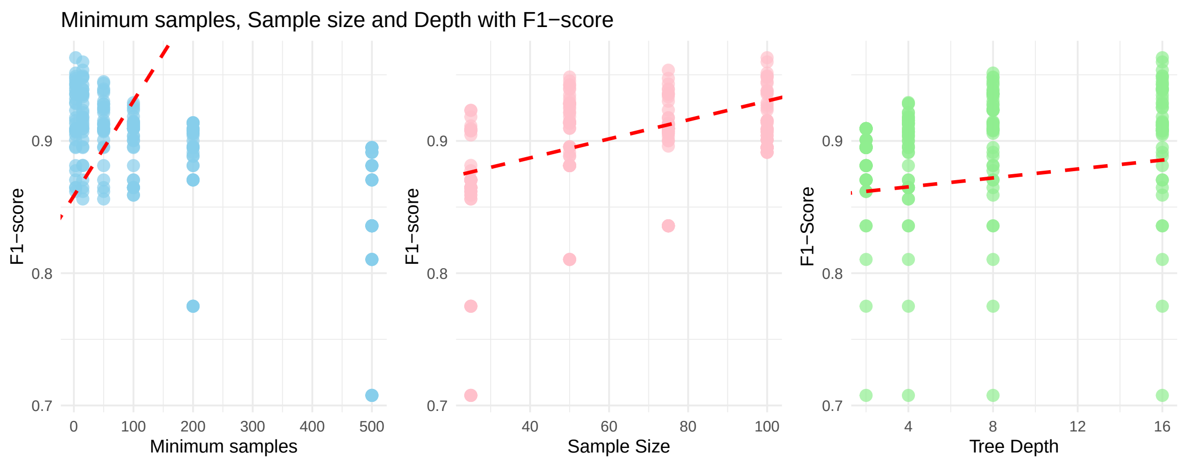


Figure 6: Depth and Sample size regressed on F1-score

The resulting model accounted for approximately 60% of the variance in the data (Rsquared: 0.5926566), with an associated RMSE of 0.0283934. Regression model performed comparatively better than the accuracy model. The lack of variance explained can be attributed to a limited sampling size which did not encompass enough variance to identify a robust trend.

The observed coefficients (Minimum samples:$-1.5276461 \times 10^{-4}$, Sample size:$7.1714674 \times 10^{-4}$ and Depth: 0.0016896) reveal that Minimum samples has a negative impact, where as sample size and Depth have a positive impact on the unit change in F1-score. They are significant with pvalues: $5.1445558 \times 10^{-27}$, $3.0908258 \times 10^{-18}$, 2.0115731×10^{-5}. These results align with our prior expectations regarding the relationship within these datasets. The positive influence of Sample Size and Depth on F1-score is consistent with the notion that an increase in complexity parameters will contribute positively to weighted model performance metrics. Simultaneously, the negative impact of Minimum samples suggests that reducing the complexity of the model increases generalization and costs us the gain in weighted evaluation metrics.

4. Conclusion

In this study, a comprehensive evaluation of a custom decision tree classifier was conducted, comparing its performance to the decision tree classifier from the SKlearn library across multiple datasets. The primary objective of this study was to assess the efficacy of the custom decision tree classifier and understand how its performance compares to the widely-used SKlearn decision tree classifier. The research addressed the challenge of achieving accurate and efficient classification across datasets of varying sizes and complexities.

The custom implementation demonstrated consistent performance across diverse datasets, showcasing that while the parameters of classification impact evaluation metrics to some extent, they were not significantly impacted by variable types (quantitative, binary or multiclass categorical), dataset attributes (size and features), or the information gain method used. Pivotal impact was held by the intrinsic properties of the dataset and the capacity of its feature to represent classification outcomes. The focus on F1-score highlighted the weighted impact of classes, and the balance between precision and recall estimates, offering a representative measure of the model's overall effectiveness. It was higlhighted at many instances that the accuracy metric is not appropriate in many conditions such as in imbalanced classes or multiclass problems, and metrics such as F1-score and Matthew's correlation may perform better in such cases.

A detailed analysis revealed marginal differences in outcomes between the custom and SKlearn classifiers. Despite these differences, the performance of the custom classifier closely aligned with the SKlearn classifier under comparable settings. While the custom classifier exhibited longer training times compared to SKlearn, the difference was consistent across datasets and did not significantly impact performance. The findings emphasize the importance of considering both evaluation and computational metrics when assessing classifier performance. As such, results support the viability of custom classifier for 'best fit' classification without pruning

Through a detailed comparison of custom classifier's results, it was established that increasing tree depth demonstrated a small impact on evaluation metrics but had a greater impact on computational properties, creating a continuous rise in training time and memory utilization, indicating a trade-off between complexity and computational efficiency. The examination of tree depths across datasets revealed nuanced trends, emphasizing the importance of selecting optimal depth values to prevent overfitting and achieve efficient model performance. A detailed exploration of the impact of depth and minimum samples on evaluation metrics provided insights into achieving a trade-off between complexity and generalization.

Summarily, this study contributes valuable insights into the performance of a custom decision tree classifier, offering a nuanced understanding of its strengths, limitations, and areas for improvement. Results highlight the critical role of model complexity in achieving balanced and efficient classification. Both underfitting and overfitting were observed, emphasizing the need for thoughtful parameter selection to achieve good metrics and to balance dataset nuances without overfitting. The trade-off in computational efficiency did not significantly impact overall performance, despite longer training times, the custom classifier remains a viable option for classification tasks. In future, exploring pruning techniques and ensemble methods such as random forests, could enhance the computational efficiency of the custom decision tree.

5. Appendix

Table 11: Detailed Evaluation of Metrics Across Tree Depths

depth	Dataset	Features	Accuracy	F1score	MCC	Time	Memory	Similarity
2	Wine	12	0.540	0.467	0.165	1.228	57.603	1.000
4	Wine	12	0.531	0.494	0.199	2.591	58.026	0.998
8	Wine	12	0.532	0.504	0.212	5.786	55.852	0.985
16	Wine	12	0.531	0.507	0.221	9.266	56.281	0.954
2	CustomerChurn	14	0.897	0.880	0.565	1.053	49.156	1.000
4	CustomerChurn	14	0.900	0.888	0.588	1.942	50.201	0.998
8	CustomerChurn	14	0.911	0.903	0.643	2.954	50.248	0.996
16	CustomerChurn	14	0.913	0.905	0.653	3.470	49.492	0.993
2	HCV	29	0.243	0.172	0.000	1.584	50.254	0.999
4	HCV	29	0.243	0.190	-0.006	2.874	50.277	0.929
8	HCV	29	0.235	0.205	-0.013	4.870	50.407	0.963
16	HCV	29	0.234	0.207	-0.015	6.264	49.607	0.928
2	Hypothyroid	18	0.974	0.974	0.816	0.373	54.175	0.998
4	Hypothyroid	18	0.975	0.976	0.832	0.652	52.998	0.996
8	Hypothyroid	18	0.974	0.975	0.821	0.960	53.055	0.994
16	Hypothyroid	18	0.974	0.975	0.822	0.965	53.211	0.994

6. References

"ANOVA Test: Definition, Types, Examples, SPSS." n.d. *Statistics How To*. Accessed January 14, 2024. https://www.statisticshowto.com/probability-and-statistics/hypothesis-testing/anova/.

Paulo Cortez, A. Cerdeira. 2009. "Wine Quality." UCI Machine Learning Repository. https://doi.org/10.24432/C56S3T.

Quinlan, J. R. 1986. "Induction of Decision Trees." *Machine Learning* 1 (1): 81–106. https://doi.org/10.1007/BF00116251.

Quinlan, Ross. 1986. "Thyroid Disease." UCI Machine Learning Repository. https://doi.org/10.24432/C5D010.

"Repeated Measures ANOVA - Understanding a Repeated Measures ANOVA | Laerd Statistics." n.d. Accessed January 14, 2024. https://statistics.laerd.com/statistical-guides/repeated-measures-anova-statistical-guide.php.

Sanaa Kamal, Mohamed ElEleimy. 2017. "Hepatitis C Virus (HCV) for Egyptian Patients." UCI Machine Learning Repository. https://doi.org/10.24432/C5989V.

Shannon, C E. n.d. "A Mathematical Theory of Communication."

"Sklearn.tree.DecisionTreeClassifier." n.d. *Scikit-Learn*. Accessed January 14, 2024. https://scikit-learn/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

Unknown. 2020. "Iranian Churn Dataset." UCI Machine Learning Repository. https://doi.org/10.24432/C5JW3Z.