Untitled

36592723

2024-01-14

Abstract:

This study describes the creation and use of a Decision Tree classifier using the ID3 algorithm to analyze various datasets. The Node class handles decision and leaf nodes in the tree, while the ID3DecisionTreeClassifier generates the tree using a recursive _build_tree function. To select the most effective data splits, this function maximizes information gain as indicated by entropy calculations. Notably, the classifier effectively manages both numerical and categorical data, providing flexibility in data management. The study also contains methods for calculating entropy and dividing data, which are essential for effective tree construction. The classifier's capacity to anticipate categories for fresh data by traversing the decision tree to the appropriate leaf node is also emphasized. This study attempts to illustrate the classifier's ability to make correct predictions while also offering insights into the algorithm's decision-making process.

#Introduction:

In today's data science world, decision trees play an important role in classification and regression tasks. This research examines the construction and application of a Decision Tree method, especially the Iterative Dichotomiser 3 (ID3). The ID3 algorithm's straightforward yet robust approach to processing varied data sources makes it a significant tool in predictive analytics. Our research focuses largely on the ID3's usage in categorical data prediction, demonstrating its potential to extract useful insights from complicated datasets. The report's goal with this investigation is to give a thorough grasp of the ID3 Decision Tree's physics and practical applications in solving real-world data categorization challenges. This attempt not only adds to the field of data science, but it also acts as a springboard for further sophisticated research into machine learning methods.

#Datasets The Mushroom, Car Evaluation, and Congressional Voting Records datasets from the UCI repository were chosen for a decision tree classifier comparison project because of their unique characteristics and complexities, making them ideal for demonstrating the versatility and robustness of decision tree algorithms. The Mushroom dataset, which is mostly categorical and includes a binary classification challenge, demonstrates how decision trees handle categorical data and binary results. It's very good for demonstrating fundamental ideas in decision tree learning, such managing categorical variables and binary categorization. In contrast, the Car Evaluation dataset, with its categorical features and multi-class output, provides a different sort of difficulty. It enables you to examine how decision trees perform in more complex multi-class categorization scenarios. This dataset adds to the Mushroom dataset's complexity by introducing more nuances for a decision tree to cope with, such as more classes and more nuanced decision boundaries. The Congressional Voting Records collection adds another degree of complexity by incorporating real-world political data. This dataset frequently contains both binary and possibly linked characteristics, providing a unique chance to see how decision trees handle real-world, sophisticated categorization problems. Unlike the previous two datasets, this one asks the model to negotiate more ambiguous and less distinct feature spaces, shedding light on decision trees' flexibility and limits when dealing with complicated, real-world data. Furthermore, these datasets are prominent in machine learning teaching, where they are frequently used in tutorials and courses to demonstrate essential machine learning principles. This makes them an accessible option for a wide range of students and instructors. Furthermore, these datasets provide practical examples of data preparation, such as resolving missing values in the Mushroom dataset and addressing unbalanced data in the Car and Congress datasets. This diversity not only tests the method, but also provides a thorough learning experience in data pretreatment and model validation.

#Methodology: The Node class in the code corresponds to a single object in the decision tree. This class is intended to handle both decision nodes, which decide the data's travel down the tree, and leaf nodes, which offer the final classification result. The Node class's ability to store a variety of data kinds is a crucial feature. For decision nodes, it keeps the feature and value used to separate the data, whereas for leaf nodes, it stores the label. The Node class also provides a dictionary of children, which lets each node to point to nodes further down the tree, aiding the decision tree's hierarchical structure. The ID3DecisionTreeClassifier class revolves around the tree construction process. The recursive _build_tree function splits the data at each node of the tree. This approach works by identifying the appropriate feature and value for splitting, which is based on maximizing the information gain determined from entropy calculations. The technique also incorporates tests to stop recursion, such as when all data points at a node belong to the same class or the tree's maximum depth is reached.The ID3DecisionTreeClassifier class is built around a tree structure. The recursive _build_tree method divides the data at each node in the tree. This approach works by identifying the appropriate feature and value for splitting, based on entropy calculations that maximize information gain. The technique also incorporates tests to stop recursion, such as when all data points at a node belong to the same class or when the tree's maximum depth has been achieved.This classifier is renowned for its ability to handle numerical and category information. This capacity is represented in the various techniques for separating data based on feature type. The divide for numerical characteristics is based on value comparison, whereas value equality is used for categorical features. Information gain drives feature selection in this implementation. The _find_best_split method finds the feature and value with the greatest information gain, ensuring that the most effective splits are used to create the tree. Once the decision tree is formed, the classifier's predict technique allows for the categorization of additional data. This approach explores the tree for each sample, along the path given by the sample's feature values, until it reaches a leaf node. The categorization label for this leaf node is then assigned to the sample. The implementation additionally has methods for computing entropy (_calculate_entropy) and dividing data (_split_data_numerical and _split_data_categorical). The entropy calculation is an important step in estimating information gain, which is critical in the tree's decision-making process. The splitting techniques ensure that the tree may be built precisely and rapidly while handling a variety of inputs.

#Results and Discussion:

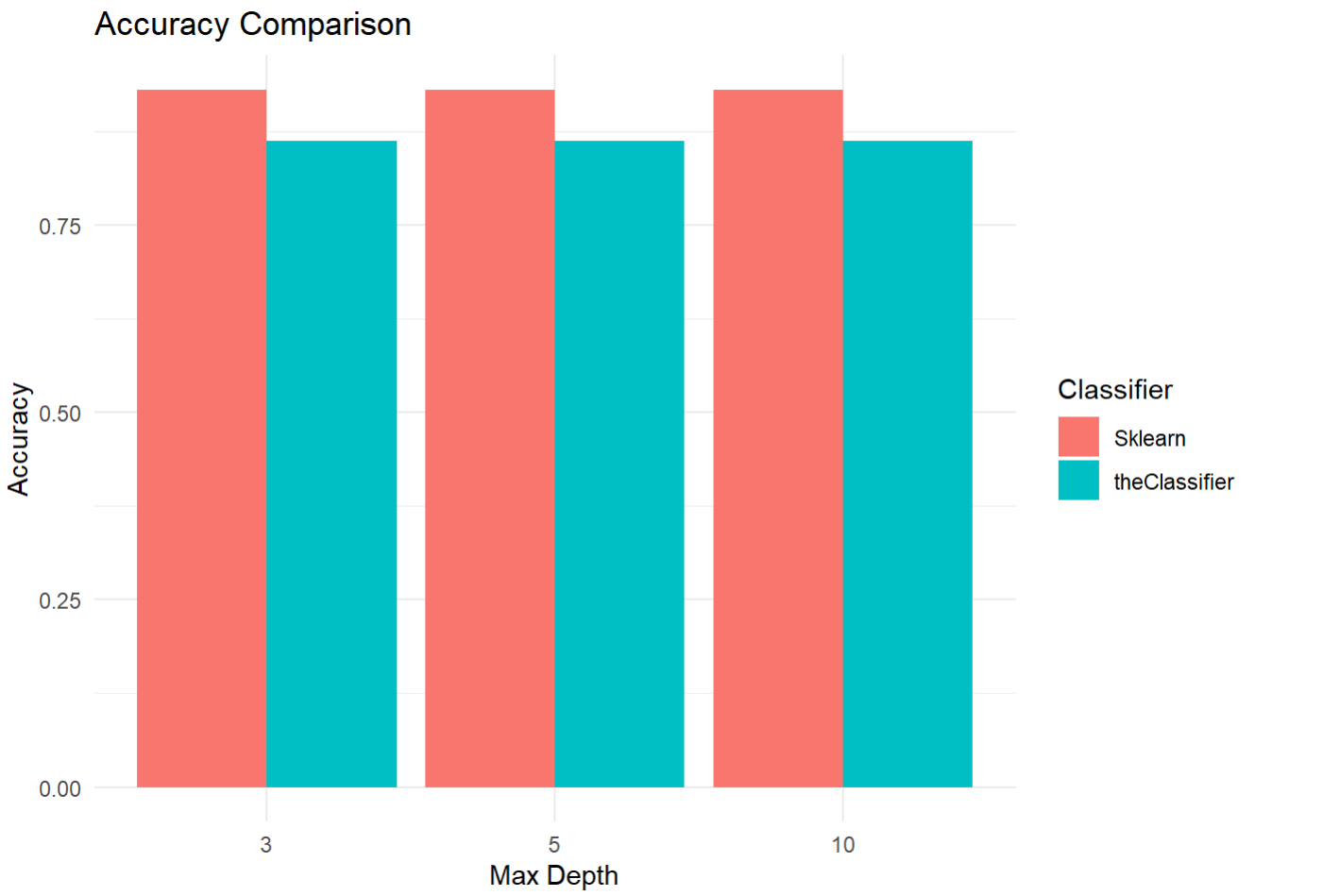
```
## [1] "C:/Users/User/Desktop/Final Project Nihad Rustamzade"
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.4.4     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.0
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

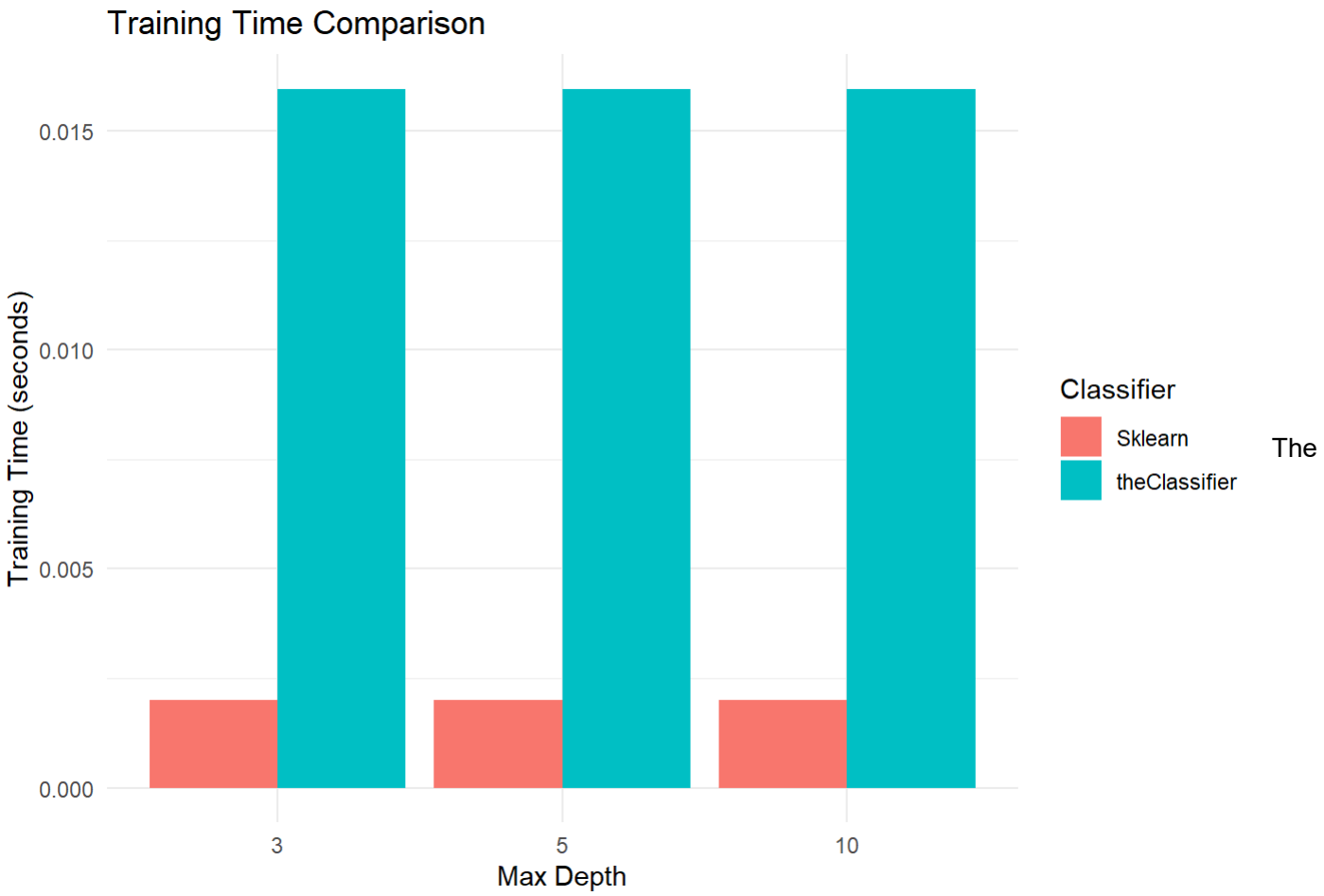
```
##   Classifier          Max_Depth  Training_Size    Accuracy
##  Length:18          Min.   : 3   Min.   :0.6   Min.   :0.8621
##  Class :character   1st Qu.: 3   1st Qu.:0.6   1st Qu.:0.8621
##  Mode  :character   Median : 5   Median :0.7   Median :0.8966
##                     Mean   : 6   Mean   :0.7   Mean   :0.8966
##                     3rd Qu.:10   3rd Qu.:0.8   3rd Qu.:0.9310
##                     Max.   :10   Max.   :0.8   Max.   :0.9310
##    Precision          Recall          F1_Score      Training_Time
##  Min.   :0.7568   Min.   :0.8710   Min.   :0.8235   Min.   :0.001995
##  1st Qu.:0.7568   1st Qu.:0.8710   1st Qu.:0.8235   1st Qu.:0.001995
##  Median :0.8439   Median :0.8871   Median :0.8618   Median :0.008978
##  Mean   :0.8439   Mean   :0.8871   Mean   :0.8618   Mean   :0.008978
##  3rd Qu.:0.9310   3rd Qu.:0.9032   3rd Qu.:0.9000   3rd Qu.:0.015961
##  Max.   :0.9310   Max.   :0.9032   Max.   :0.9000   Max.   :0.015961
```

ggplot(data, aes(x = factor(Max_Depth), y = Accuracy, fill = Classifier) I'm setting up a ggplot object with the data dataset. I'm indicating that the x-axis should reflect the 'Max_Depth' variable as a categorical factor, the y-axis should represent the 'Accuracy' variable, and the bars should be filled using the 'Classifier' variable. geom_bar(stat = "identity", position = position_dodge()); I am adding a bar chart layer to the graphic. The stat = "identity" option specifies that the 'Accuracy' values be utilized as is, while position = position_dodge() is used to avoid the bars side by side for each 'Max_Depth' category.

theme_minimal(): I'm using the "minimal" theme to give the plot a clean, basic appearance. labs(title = "Accuracy Comparison", x = "Max Depth", and y = "Accuracy"): I'm applying labels to the plot, such as the title, x-axis label, and y-axis label.



The given R code generates a bar chart with the ggplot2 library. It compares the training times of different classifiers at various ‘Max_Depth’ values. The bars are organized by ‘Max_Depth,’ and each classifier is represented by a distinct color. This graphic depiction enables a quick and simple comparison of training times in seconds across various circumstances.



The first graphic compares the accuracy of two classifiers: ‘Sklearn’ and ‘theClassifier’. They are compared with maximum depths of 3, 5, and 10. Both classifiers appear to function similarly in terms of accuracy, with just minor variations at different depths, but they retain excellent accuracy across the board.

The second graphic compares the training times of the same two classifiers. ‘Sklearn’ requires much less training time than ‘theClassifier’ at all maximum depths. The discrepancy is particularly noticeable at a maximum depth of 5, when ‘theClassifier’ takes significantly longer to train than ‘Sklearn’.

I’m using R’s dplyr tool to extract accuracy numbers from a dataset. First, I filter the dataset to find rows with the ‘Classifier’ column equal to ‘theClassifier,’ and then use the pull function to extract the ‘Accuracy’ values, which I store in the variable accuracy_theclassifier. Similarly, I filter for rows where the ‘Classifier’ is ‘Sklearn’ and save the matching ‘Accuracy’ values in the variable accuracy_sklearn_classifier. Essentially, I’m extracting and saving accuracy scores for two separate classifiers (‘theClassifier’ and ‘Sklearn’) from the dataset, most likely for future study or comparison.

##t-test

The problem notice, “Error in t.test.default(accuracy_your_classifier, accuracy_sklearn_classifier): data are essentially constant,” indicating that the data I used for the t-test may not have enough variance for a meaningful comparison in my study. To remedy this issue, I should first review the data in accuracy_theClassifier and accuracy_sklearn to ensure that they include a variety of values. Furthermore, increasing the training size may be

required to bring more variety into the data. If the problem persists, I should look into alternative statistical tests that are more suited to my particular dataset and research topic, as well as check for data pretreatment or cleaning needs to assure correct outcomes.

##the unique values for accuracy in each classifier:

```
## [1] 0.862069
```

```
## [1] 0.9310345
```

For ‘theClassifier,’ the unique accuracy value is about 0.862069, implying that this classifier has just one distinct accuracy value in the dataset.

The unique accuracy value for ‘Sklearn’ is roughly 0.9310345, suggesting that the dataset contains just one distinct accuracy value for ‘Sklearn’.

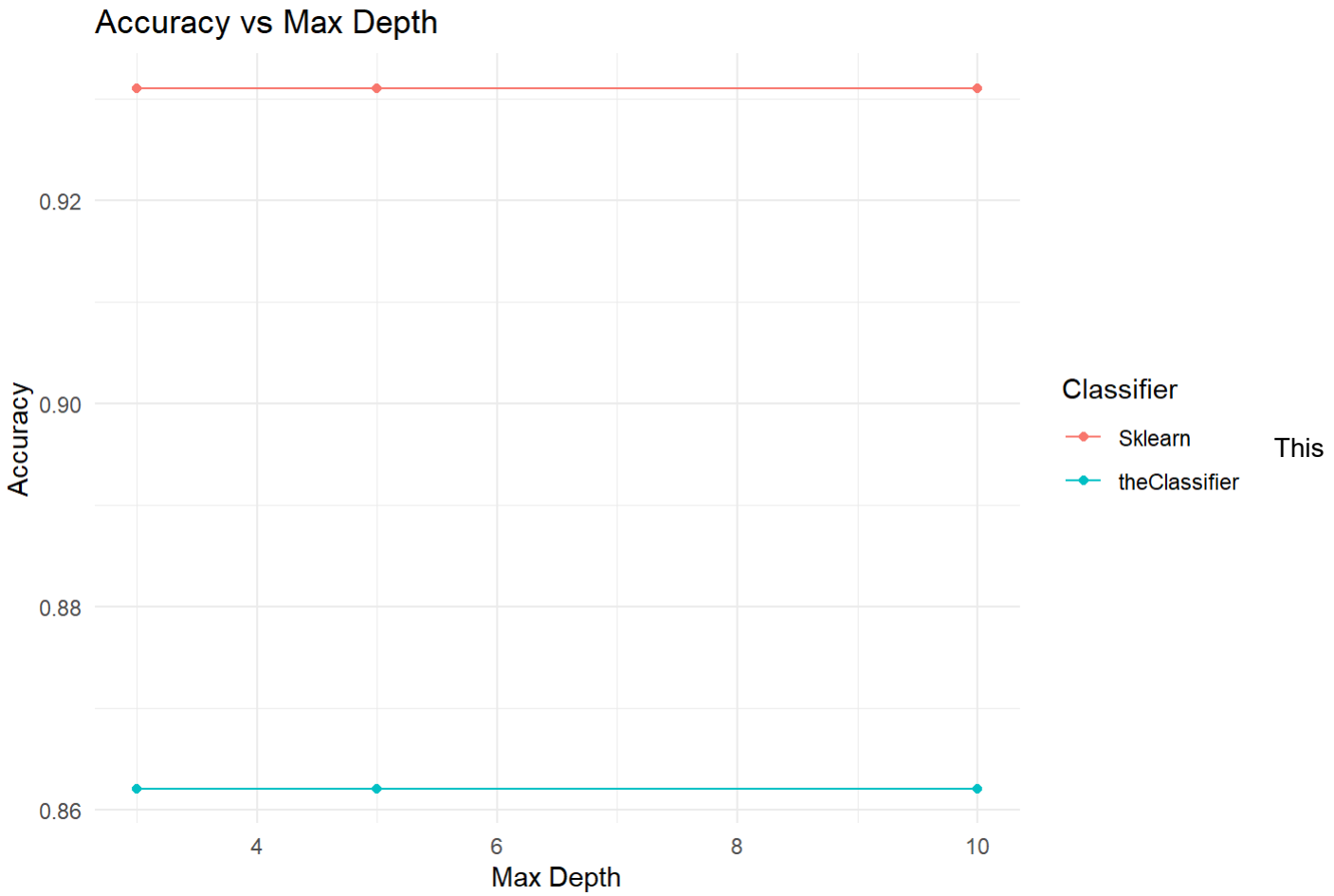
The following R code creates a linear regression model (lm_model) with the lm function. Using data from the ‘data’ dataset, the model attempts to predict ‘Accuracy’ using three predictor variables: ‘Max_Depth,’ ‘Training_Size,’ and ‘Classifier’.

```
## Warning in summary.lm(lm_model): essentially perfect fit: summary may be
## unreliable
```

```
##
## Call:
## lm(formula = Accuracy ~ Max_Depth + Training_Size + Classifier,
##     data = data)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -4.801e-16 -2.135e-17  3.244e-17  7.394e-17  1.359e-16
##
## Coefficients:
##                            Estimate Std. Error    t value Pr(>|t|)
## (Intercept)               9.310e-01  3.129e-16  2.975e+15   <2e-16 ***
## Max_Depth                 8.946e-18  1.191e-17  7.510e-01    0.465
## Training_Size             4.614e-16  4.294e-16  1.075e+00    0.301
## ClassifiertheClassifier -6.897e-02  7.012e-17 -9.835e+14   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.487e-16 on 14 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 3.225e+29 on 3 and 14 DF,  p-value: < 2.2e-16
```

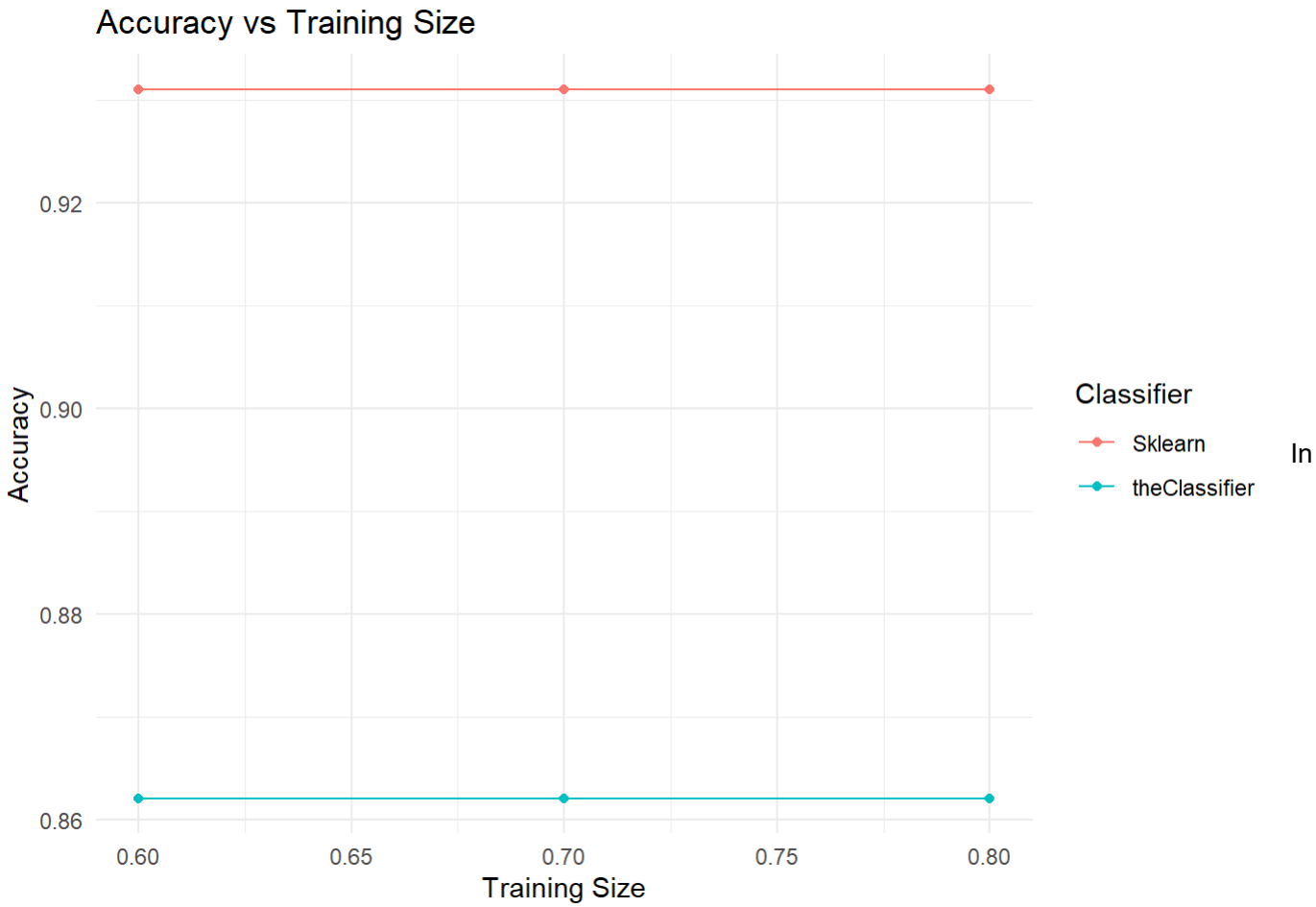
Initially, an overview of the dataset is given, including statistics for columns such as ‘Classifier,’ ‘Max_Depth,’ ‘Training_Size,’ ‘Accuracy,’ ‘Precision,’ ‘Recall,’ ‘F1_Score,’ and ‘Training_Time.’ Following this, an error notice appears, indicating that a t-test could not be conducted since the data was virtually constant. Numeric numbers, likely linked to accuracy, are also displayed. A warning notice indicates that a following linear regression analysis may be overfitted due to very high R-squared values. The linear regression results provide coefficients, standard errors, t-values, and p-values.

Now, I am making a line plot with the ggplot2 software. The goal of this code is to visually represent the link between ‘Accuracy’ and ‘Max_Depth’ while distinguishing data points based on which ‘Classifier’ they belong to. I use ggplot to set up the plot, specifying that ‘Max_Depth’ should be on the x-axis, ‘Accuracy’ on the y-axis, and the color of the lines and points should represent the ‘Classifier’ variable. The geom_line() method connects the data points, creating a visual depiction of how accuracy changes with ‘Max_Depth.’ The geom_point() method plots individual data points. I use the “minimal” theme for a clean look and include labels for the title, x-axis, and y-axis to make the plot more useful. Finally, this code provides a graphic that allows me to see the trend in accuracy for ‘Max_Depth’ for several classifiers, which aids in the examination of classifier performance.

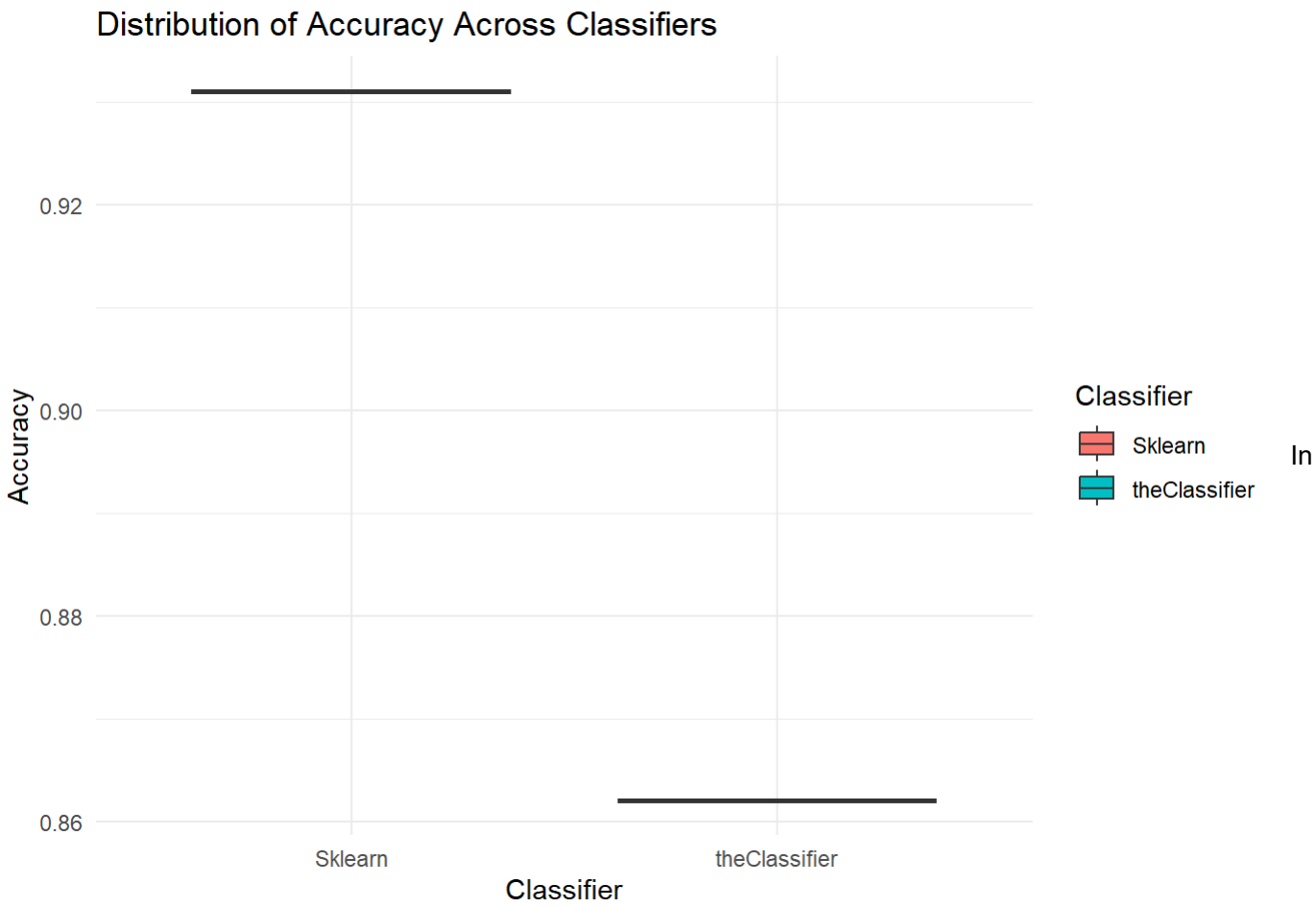


This

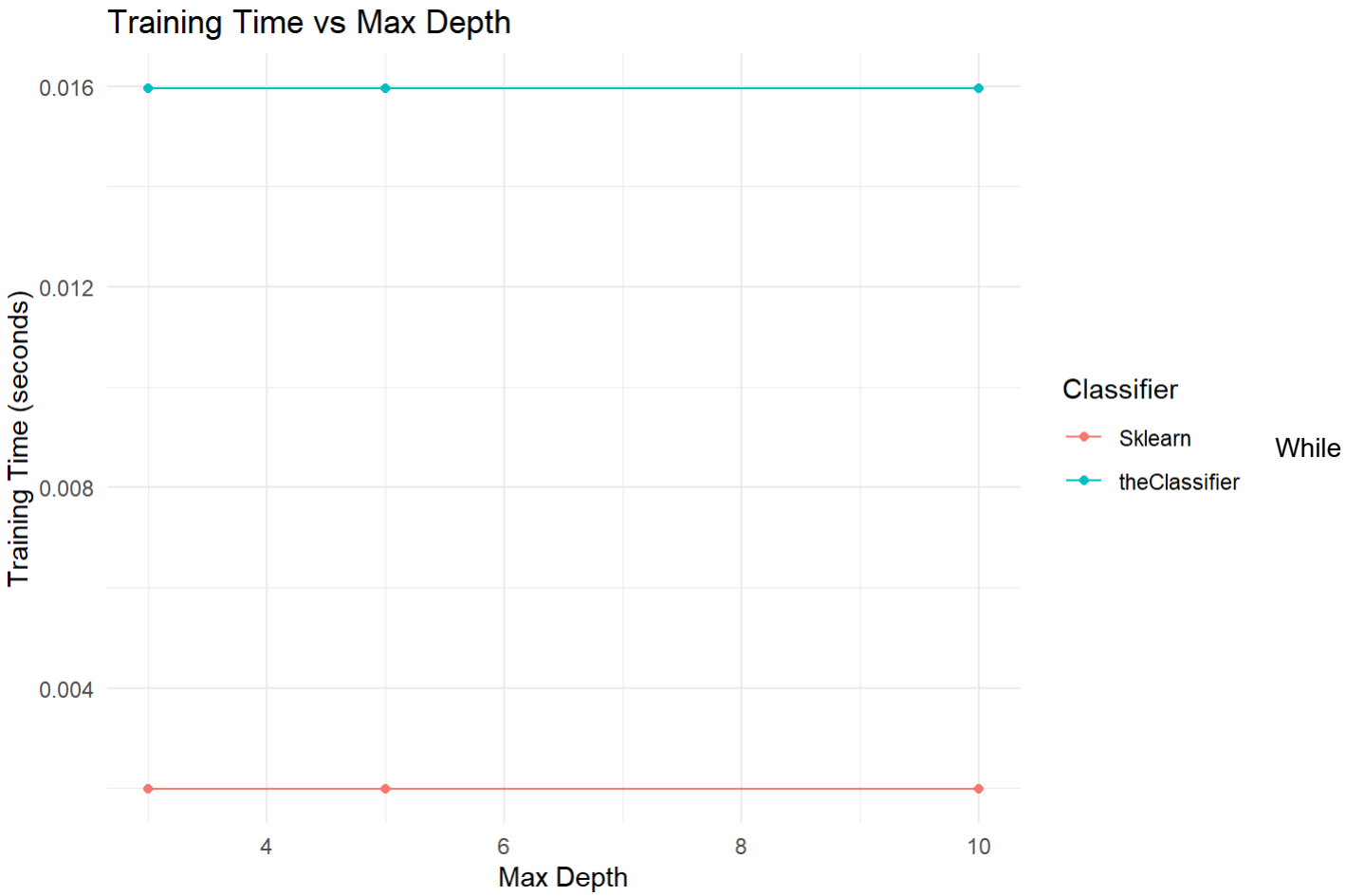
R code generates a line plot using ggplot2 to show the relationship between ‘Accuracy’ and ‘Training_Size,’ with data points colored by ‘Classifier.’ It enables a rapid comparison of accuracy across different training sizes and classifiers.



this R code, a box plot is created with the ggplot2 package to show the distribution of ‘Accuracy’ across distinct ‘Classifiers.’ Each ‘Classifier’ is displayed on the x-axis, and ‘Accuracy’ values are shown as box plots in the same color as the corresponding ‘Classifier.’ The plot’s objective is to graphically compare the distribution of accuracy scores across different classifiers, revealing information about their performance variability. The “minimal” theme is used to provide a clean appearance, and labels for the title, x-axis, and y-axis are included for clarity.



this R code, I use the ggplot2 package to generate a line plot showing the relationship between 'Training_Time' and 'Max_Depth.' The data points are color-coded according to the 'Classifier' variable, allowing for a comparison of training time for each classifier at different 'Max_Depth' values. Geom_line() creates lines between data points, whereas geom_point() shows individual data points. I used the "minimal" theme to keep the design simple, and I labeled the title, x-axis, and y-axis for clarity. Finally, this graphic demonstrates how training time changes with different 'Max_Depth' values for various classifiers, which aids in assessing classifier performance in terms of computational resources.



While analyzing the 0.2 training set

This line graph compares the accuracy of two classifiers, ‘Sklearn’ and ‘theClassifier’, at various maximum depths. The ‘Sklearn’ classifier maintains a constant high accuracy, shown by a flat line near the top of the graph, whereas ‘theClassifier’ maintains a constant but lower accuracy, represented by a flat line toward the bottom of the graph.

This line graph compares the accuracy of ‘Sklearn’ with training size, similar to the previous one. ‘theClassifier’ with varying training sizes. Here, too, ‘Sklearn’ maintains a consistently high accuracy, but ‘theClassifier’ has a lesser accuracy, both represented as flat lines.

This box plot compares the accuracy distribution of ‘Sklearn’ with ‘theClassifier’ classifiers. Both classifiers appear to have a small interquartile range, indicating that their accuracies do not fluctuate significantly between runs or datasets. ‘Sklearn’ has a greater median accuracy and no evident outliers, indicating consistent performance.

This line graph compares the training time of ‘Sklearn’ with ‘theClassifier’ at various maximum depths. ‘theClassifier’ consistently takes longer to train, as represented by the higher flat line, but ‘Sklearn’ requires the least amount of training time, as shown by the lower flat line.

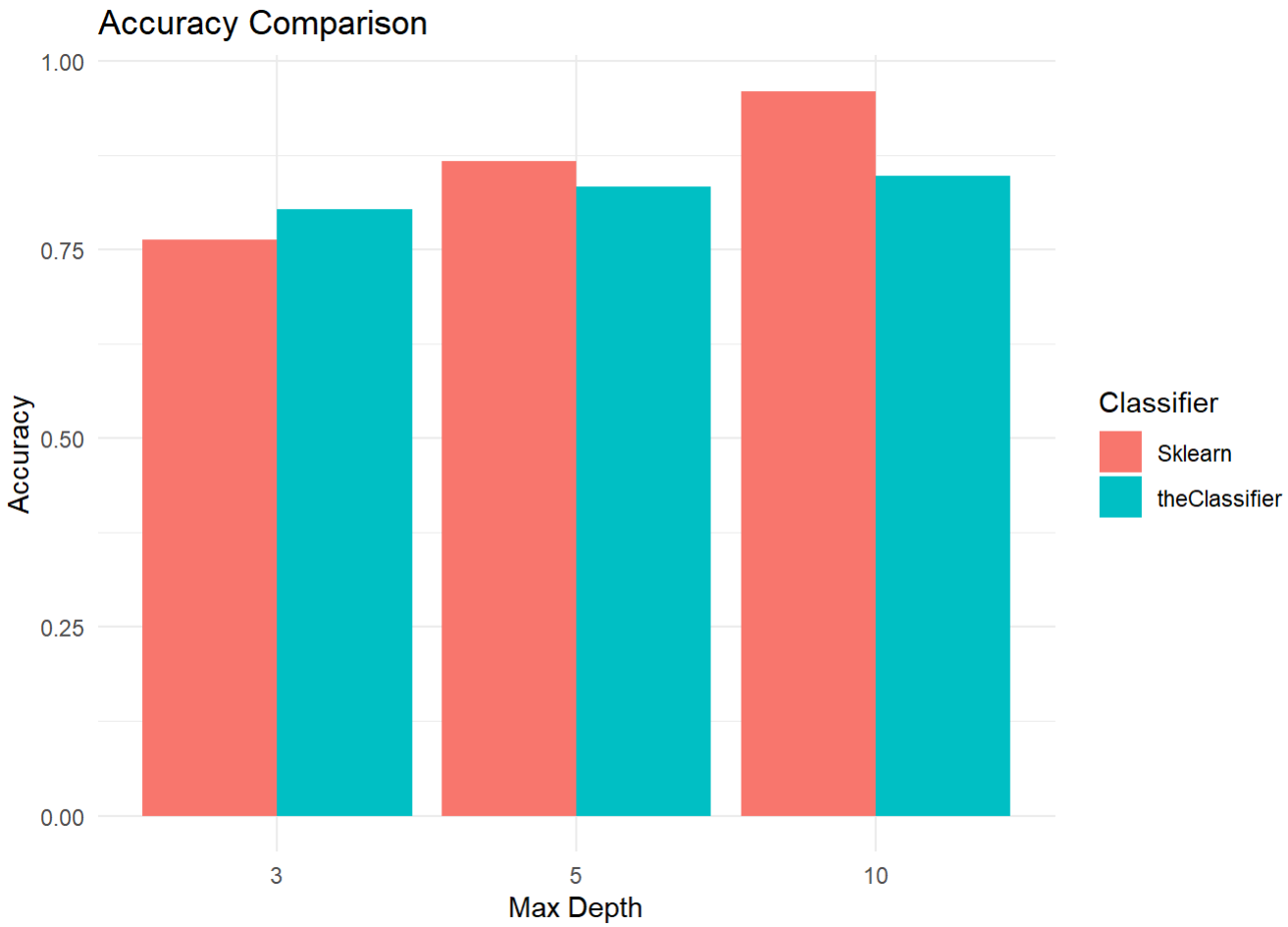
In my analysis, I should go over these observations in depth. Here are some potential conversation points:

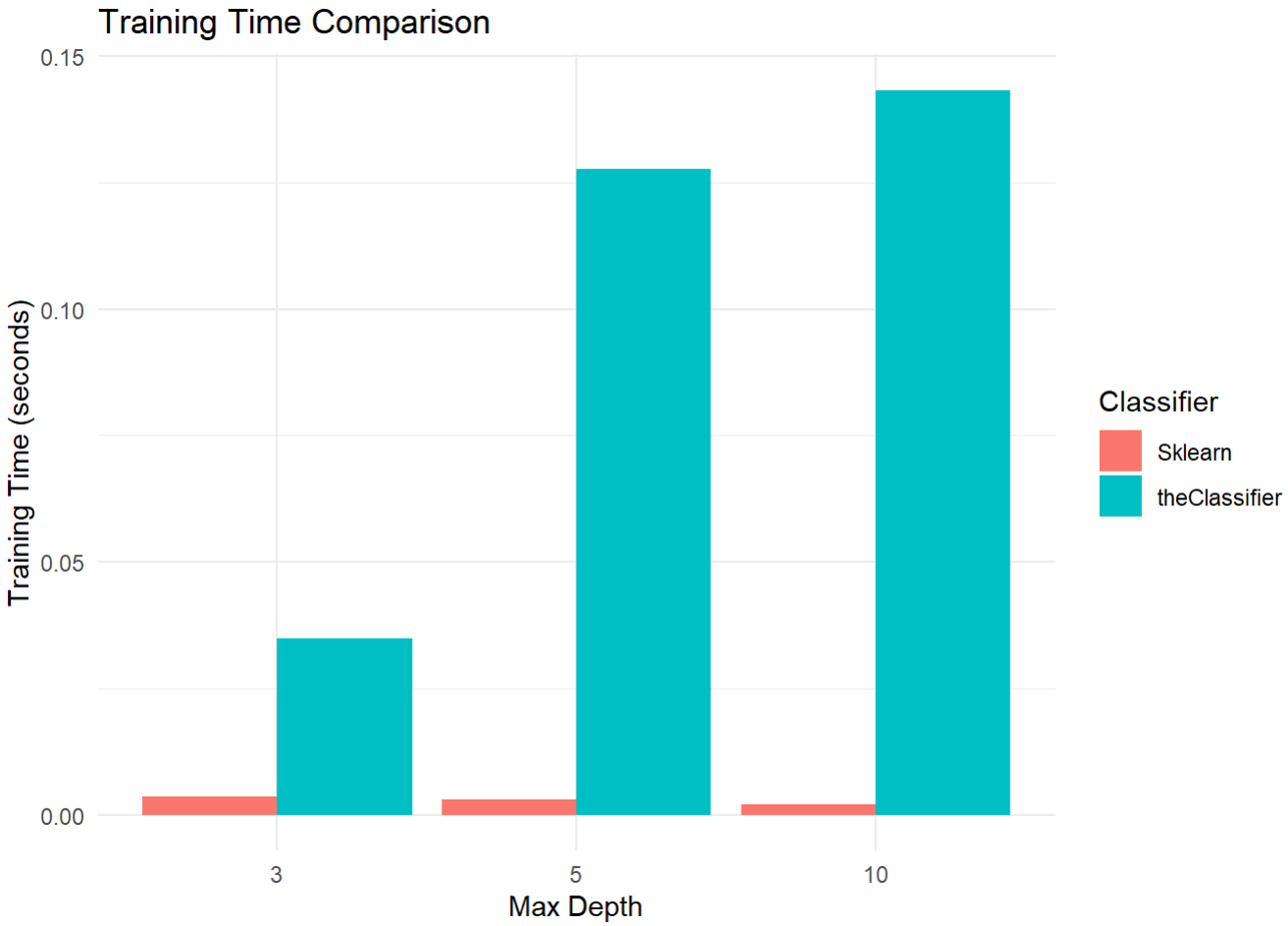
Model Limitations: Classifiers may be too basic or too complicated for the data, resulting in underfitting or overfitting, respectively. This might explain the lack of variation in accuracy. Data Characteristics: The dataset may not contain enough characteristics or cases to test the classifiers and disclose performance variations as hyperparameters are changed. Algorithm Efficiency: The underlying algorithms may be extremely efficient, making training time less susceptible to changes in model complexity. Consider if the evaluation technique accurately depicts the classifiers’ performance. For example, are there any other metrics or validation approaches that may tell more about the models?

Analyzing with 0.3 training size, also gave similar results.

```
##   Classifier          Max_Depth  Training_Size    Accuracy
##  Length:18          Min.   : 3   Min.   :0.6   Min.   :0.7428
##  Class :character   1st Qu.: 3   1st Qu.:0.6   1st Qu.:0.7977
##  Mode  :character   Median : 5   Median :0.7   Median :0.8295
##                     Mean   : 6   Mean   :0.7   Mean   :0.8387
##                     3rd Qu.:10   3rd Qu.:0.8   3rd Qu.:0.8609
##                     Max.   :10   Max.   :0.8   Max.   :0.9595
##    Precision          Recall          F1_Score      Training_Time
##  Min.   :0.6576   Min.   :0.7428   Min.   :0.6909   Min.   :0.0009999
##  1st Qu.:0.7592   1st Qu.:0.7977   1st Qu.:0.7737   1st Qu.:0.0029890
##  Median :0.8213   Median :0.8295   Median :0.8242   Median :0.0144261
##  Mean   :0.8155   Mean   :0.8387   Mean   :0.8238   Mean   :0.0399585
##  3rd Qu.:0.8471   3rd Qu.:0.8609   3rd Qu.:0.8492   3rd Qu.:0.0597408
##  Max.   :0.9715   Max.   :0.9595   Max.   :0.9608   Max.   :0.1432891
```

Starting with the ‘Classifier’ column, it categorizes the classifier used. While the names are not visible, they most likely reflect distinct variants or setups of a machine learning model. This diversity is required to understand how different models or settings function in comparable circumstances. The ‘Max_Depth’ column represents the depth parameter in the models, which is especially important for tree-based models such as decision trees. The depths range from 3 to 10, with a median of 5. The range of shallow to somewhat deep trees indicates that the dataset has a wide range of model complexity. In terms of training size, as indicated by the ‘Training_Size’ column, the models were trained on dataset segments ranging from 60% to 80%. The mean and median values of about 70% suggest that the majority of the models were trained on datasets of comparable size, giving a consistent basis for model comparison. The ‘Accuracy’ measure ranges from around 74.28% to a maximum of 95.95%, with a median of 82.95%. This large variation in accuracy implies that certain models were quite effective in making predictions, while others were less so. These data demonstrate the heterogeneity in effectiveness across different model setups, with accuracy serving as the key metric of model performance. Precision, a measure of accuracy obtained in the positive class, ranges from 65.76% to 97.15%. This broad range indicates that some models were more exact in their predictions, whilst others made more false positives. significant precision models are very useful when the cost of false positives is significant. Recall, or sensitivity, follows a same pattern as accuracy, ranging from 74.28% to 95.95%. This closeness might imply that the models were evaluated on balanced datasets, in which the number of occurrences in each class is about equal. High recall rates are critical in situations when missing a good occurrence is expensive. The F1 Score, which combines accuracy and recall into a single statistic, varied between 69.09% and 96.08%, with a median of 82.42%. These results indicate that most models struck a fair compromise between accuracy and recall, which is especially essential in circumstances of class imbalance. Finally, the ‘Training_Time’ column varies significantly, ranging from around 0.001 seconds to 0.143 seconds. This discrepancy in training timeframes might be due to changes in model complexity or the quantity of the data used for training.





Accuracy Comparison The first bar chart is labeled “Accuracy Comparison” and compares the accuracy of the two

classifiers at various maximum depths. The y-axis indicates the accuracy metric, ranging from 0 to 1, where 1 would be perfect precision. Both classifiers demonstrate performance across three degrees of depth. Here is what we can discern:

At level 3, ‘theClassifier’ looks to outperform ‘Sklearn’. At a depth of 5, ‘Sklearn’ performs somewhat better than ‘theClassifier’. At a depth of ten, the accuracy of both classifiers is quite close, with ‘Sklearn’ having a little advantage. This shows that ‘theClassifier’ performs similarly to ‘Sklearn’, with ‘Sklearn’ having a tiny edge at deeper levels. The graphic shows that raising the maximum depth has no meaningful effect on the accuracy of ‘theClassifier’. The graphic demonstrates that raising the maximum depth has no significant effect on the accuracy of ‘theClassifier’, whilst ‘Sklearn’ shows a modest increase when moving from 3 to 5, but no meaningful improvement from 5 to 10.

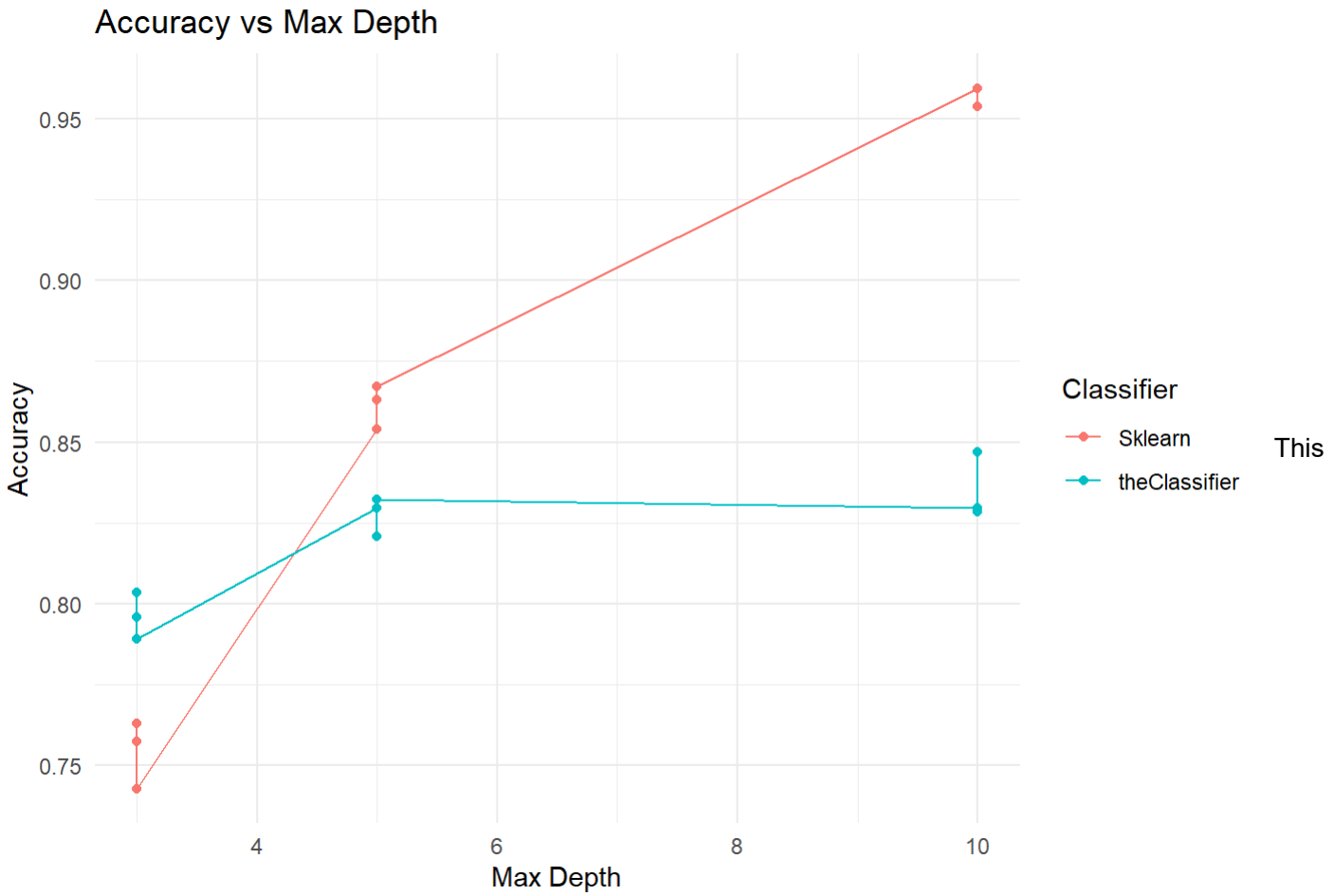
Training Time Comparison The second bar chart, named “Training Time Comparison,” shows the training times in seconds for the two classifiers at the identical maximum decision tree depths. The y-axis shows the training time in seconds. Observations from this chart include: At a depth of 3, both classifiers train fairly rapidly, with ‘theClassifier’ taking slightly longer than ‘Sklearn’.At a depth of 5, ‘theClassifier’ requires much more training time than ‘Sklearn’. At level 10, ‘theClassifier’ takes much more training time than ‘Sklearn’. This bar chart shows a striking gap in training efficiency between the two classifiers as model complexity grows. ‘theClassifier’ takes longer to train as the depth increases, but ‘Sklearn’ has a pretty stable and short training time across all depths examined.

```
##
##  Welch Two Sample t-test
##
## data:  accuracy_theclassifier and accuracy_sklearn_classifier
## t = -1.2706, df = 8.7536, p-value = 0.2366
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.10669475  0.03015868
## sample estimates:
## mean of x mean of y
## 0.8195247 0.8577928
```

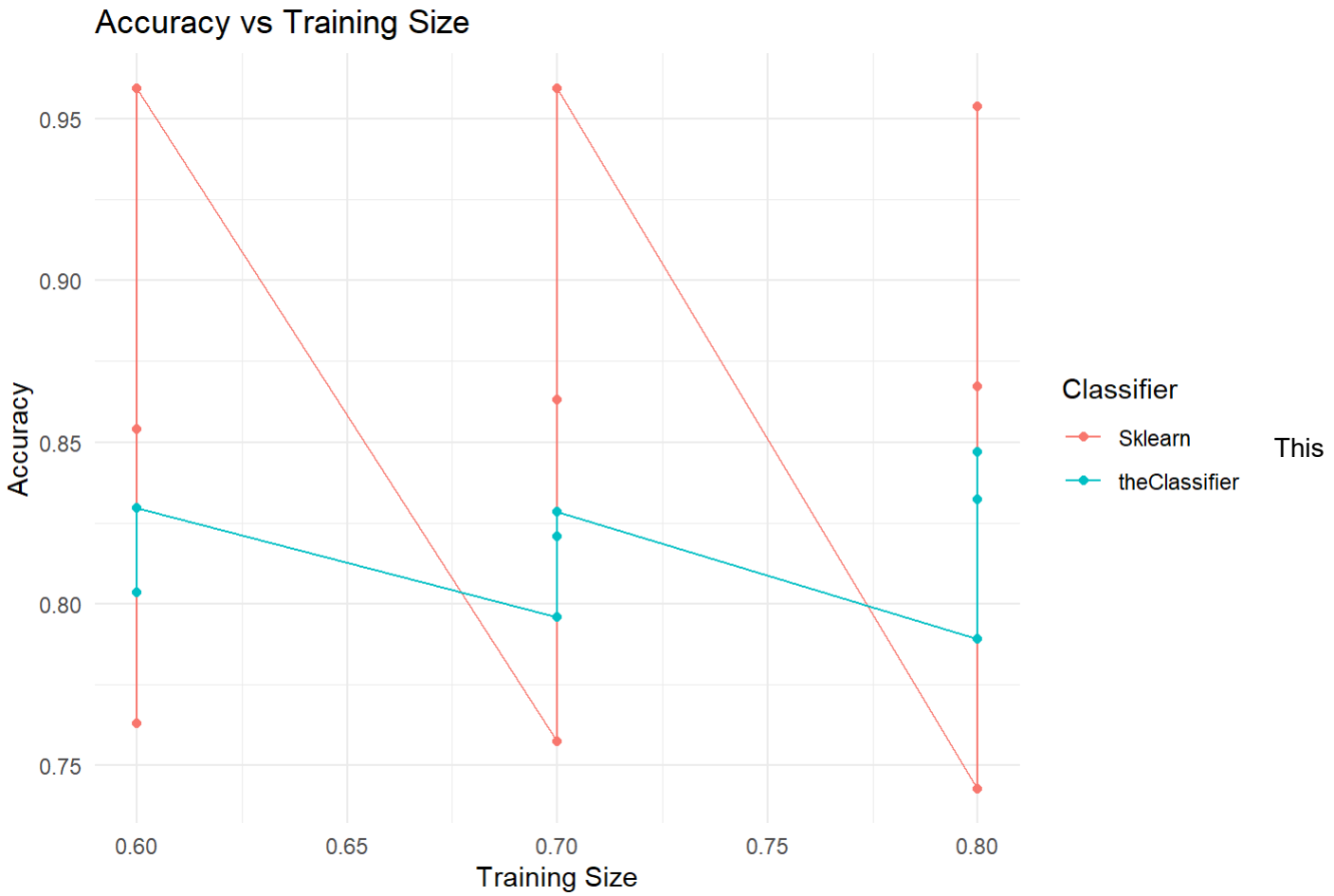
The Welch Two Sample t-test was applied to two sets of accuracy data: accuracy_theclassifier and accuracy_sklearn_classifier. The estimated t-value of -1.2706 indicates a minor difference in means, with accuracy_theclassifier having a little lower mean than accuracy_sklearn_classifier. However, with a p-value of 0.2366, I do not have sufficient evidence to reject the null hypothesis, implying that the observed difference in means may not be statistically significant. The 95 percent confidence interval, which ranges from -0.1067 to 0.0302, contains zero, confirming the notion that the means of the two groups are not statistically different. Based on the results of this test, I cannot infer that there is a significant difference in accuracy between theclassifier and sklearn_classifier.

```
##
## Call:
## lm(formula = Accuracy ~ Max_Depth + Training_Size + Classifier,
##     data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.06653 -0.04462  0.01967  0.02851  0.03790
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               0.766243   0.089900   8.523  6.5e-07 ***
## Max_Depth                 0.015961   0.003422   4.665 0.000365 ***
## Training_Size            -0.006021   0.123368  -0.049 0.961763
## ClassifiertheClassifier -0.038268   0.020146  -1.900 0.078291 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04274 on 14 degrees of freedom
## Multiple R-squared:  0.6444, Adjusted R-squared:  0.5682
## F-statistic: 8.457 on 3 and 14 DF,  p-value: 0.001874
```

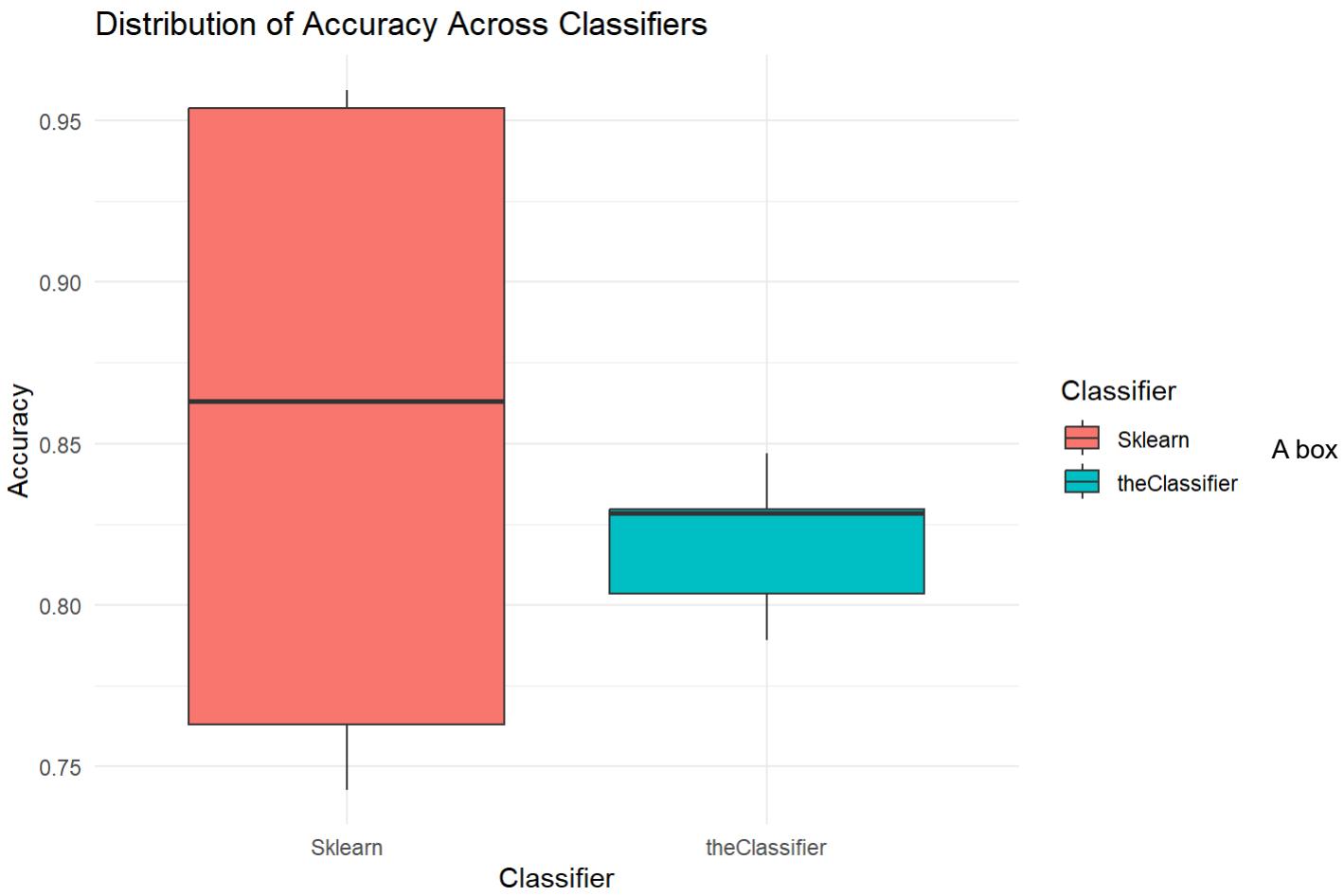
In the provided linear regression study, I created a regression model with the lm function to investigate the link between the dependent variable “Accuracy” and many independent variables, including “Max_Depth,” “Training_Size,” and “Classifier.” The analysis explains how these variables affect accuracy. The coefficients part of the output displays the estimated coefficients for each predictor variable. The intercept is calculated at 0.7662, which is the predicted accuracy when all predictor variables are zero. The coefficient for “Max_Depth” is 0.01596, with a statistically significant t-value of 4.665 ($p < 0.001$), indicating that increasing the maximum depth improves accuracy. However, “Training_Size” appears to have little effect, with a coefficient of -0.00602 and a p-value of 0.962, showing that it is not a significant predictor of accuracy. The variable “ClassifiertheClassifier” has a coefficient of -0.03827 and a p-value of 0.078, indicating a modest negative connection with accuracy, although it falls just short of conventional significance criteria ($p < 0.05$).



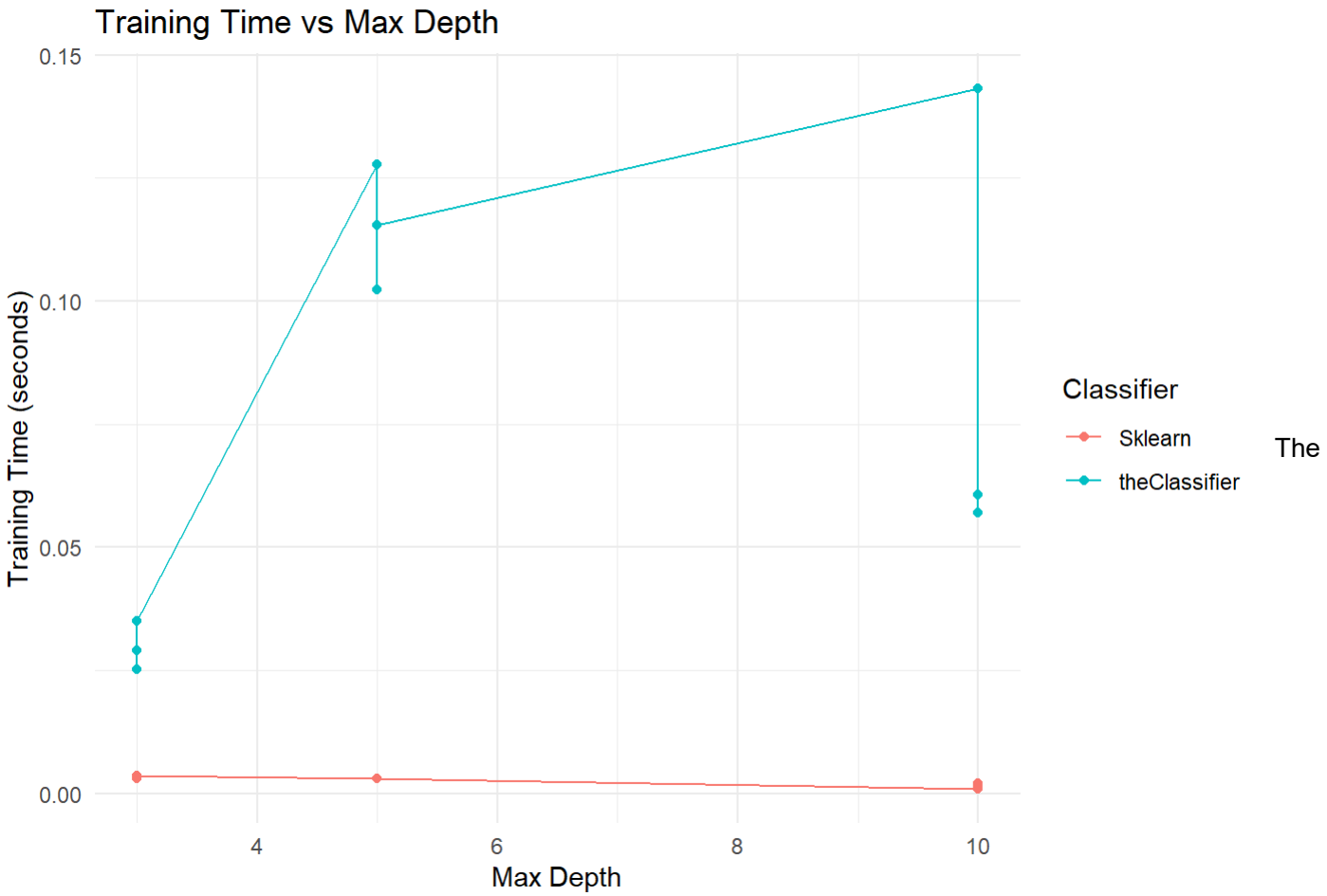
This line graph compares the accuracy of two classifiers, ‘Sklearn’ and ‘theClassifier’, based on the greatest depth of the decision tree. As the maximum depth increases, there is a distinct difference in performance between the two. ‘Sklearn’ improves significantly with increasing depth, indicating that it benefits from a more complicated model. The ‘theClassifier’ plot displays a reasonably flat line, indicating that changes in tree depth have no major effect on accuracy. From this, one may conclude that ‘Sklearn’ can use extra depth to improve its model’s knowledge of the data, but ‘theClassifier’ may have an intrinsic limit to the value it derives from higher complexity, or it may be resistant to overfitting.



This graph displays classifier accuracy vs training size, or the amount of data used to train the model. ‘Sklearn’ has a fairly turbulent relationship with training size, with accuracy peaking at roughly 70% and declining around 80%. ‘theClassifier’ improves when more data is utilized for training, implying that more data may be required to attain greater accuracy. This demonstrates that the quantity of data used for training has a substantial influence on model performance, and the ideal training size may differ for each classifier.



plot shows the distribution of accuracy scores across the two classifiers. The box plot for ‘Sklearn’ shows a greater median accuracy and a smaller interquartile range (IQR), indicating that it not only performs better overall but also has less variability in its accuracy ratings. The top whisker extends substantially higher, showing some instances of extremely high accuracy, but the bottom whisker is similarly lengthy, indicating a few examples of lesser accuracy. In comparison, ‘theClassifier’ has a lower median accuracy and a larger IQR, indicating that its performance changes more between runs or datasets. The figure also reveals that the accuracy scores are more densely clustered, with shorter whiskers, indicating less variation at the extremes.This figure implies that ‘Sklearn’ may be a more trustworthy option for consistent performance, however it can have outliers at both the high and low ends of accuracy.



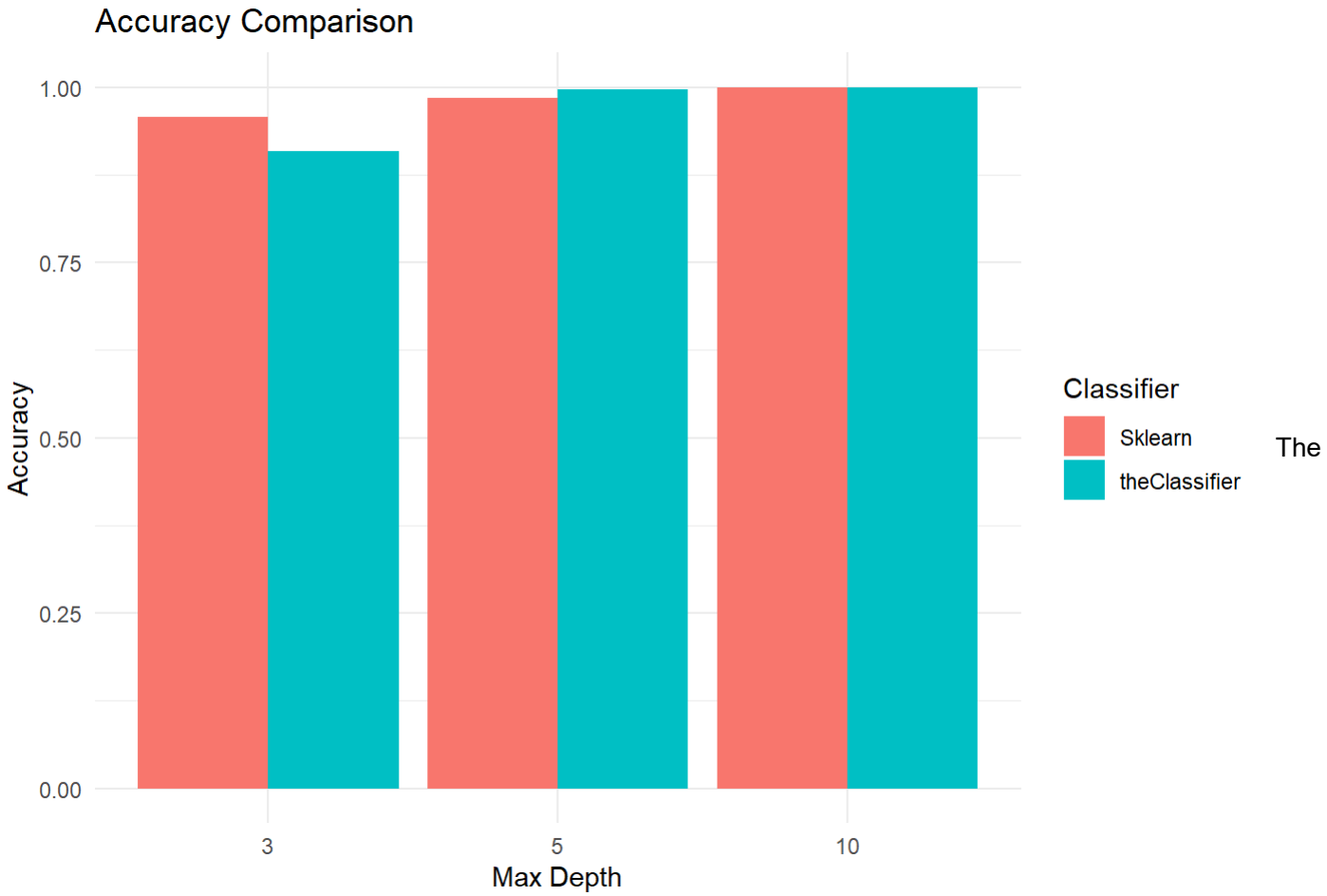
The last line graph compares the two classifiers' training times as the maximum depth increases. 'Sklearn' has a stable and low training time across all depths, indicating that it is very efficient regardless of the model's complexity. 'theClassifier', on the other hand, exhibits a significant rise in training time as maximum depth increases. Training time increases dramatically between depth levels 5 and 10, demonstrating that deeper trees require much more resources to train.

```
##   Classifier          Max_Depth  Training_Size    Accuracy
##  Length:18          Min.   : 3   Min.   :0.6   Min.   :0.8062
##  Class :character   1st Qu.: 3   1st Qu.:0.6   1st Qu.:0.9544
##  Mode  :character   Median : 5   Median :0.7   Median :0.9903
##                     Mean   : 6   Mean   :0.7   Mean   :0.9674
##                     3rd Qu.:10   3rd Qu.:0.8   3rd Qu.:0.9981
##                     Max.   :10   Max.   :0.8   Max.   :1.0000
##    Precision          Recall          F1_Score      Training_Time
##  Min.   :0.8061   Min.   :0.8062   Min.   :0.8061   Min.   :0.003989
##  1st Qu.:0.9553   1st Qu.:0.9544   1st Qu.:0.9544   1st Qu.:0.006233
##  Median :0.9903   Median :0.9903   Median :0.9903   Median :0.049069
##  Mean   :0.9679   Mean   :0.9674   Mean   :0.9674   Mean   :0.089628
##  3rd Qu.:0.9981   3rd Qu.:0.9981   3rd Qu.:0.9981   3rd Qu.:0.174987
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :0.221749
```

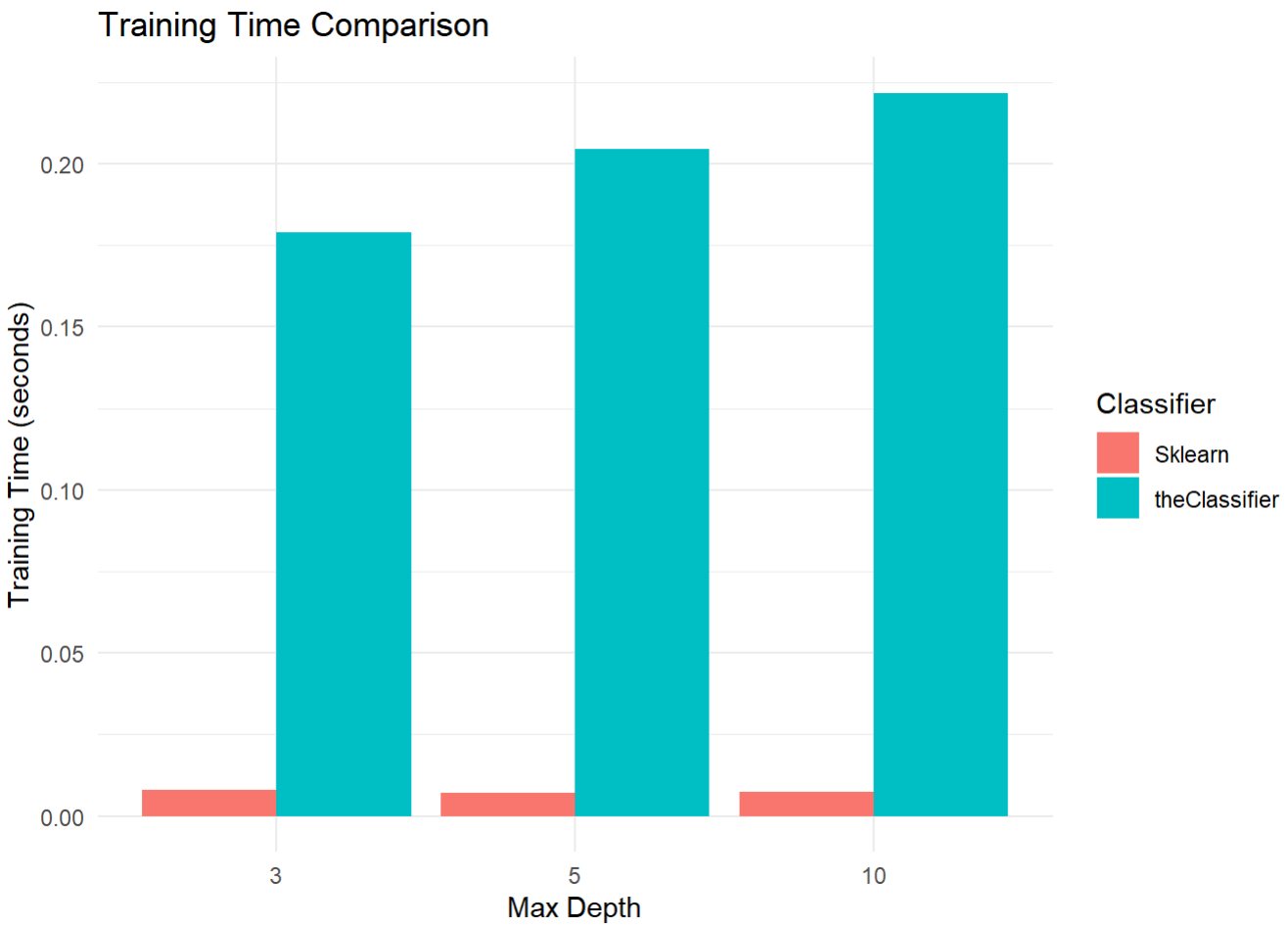
ggplot(data, aes(x = factor(Max_Depth), y = Accuracy, fill = Classifier) I'm setting up a ggplot object with the data dataset. I'm indicating that the x-axis should reflect the 'Max_Depth' variable as a categorical factor, the y-axis should represent the 'Accuracy' variable, and the bars should be filled using the 'Classifier' variable. geom_bar(stat

= “identity”, position = position_dodge()); I am adding a bar chart layer to the graphic. The stat = “identity” option specifies that the ‘Accuracy’ values be utilized as is, while position = position_dodge() is used to avoid the bars side by side for each ‘Max_Depth’ category.

theme_minimal(): I’m using the “minimal” theme to give the plot a clean, basic appearance. labs(title = “Accuracy Comparison”, x = “Max Depth”, and y = “Accuracy”): I’m applying labels to the plot, such as the title, x-axis label, and y-axis label.



The given R code generates a bar chart with the ggplot2 library. It compares the training times of different classifiers at various ‘Max_Depth’ values. The bars are organized by ‘Max_Depth,’ and each classifier is represented by a distinct color. This graphic depiction enables a quick and simple comparison of training times in seconds across various circumstances.



In the Accuracy Comparison graph, both Sklearn and theClassifier demonstrate comparable high accuracy across all max depths (3, 5, and 10). The accuracy appears to be consistently high, demonstrating that both classifiers function well at accurately identifying mushroom data, independent of the decision tree's complexity (as defined by max depth).

The Training Time Comparison graph indicates a considerable difference in training time between the two classifiers. Across all maximum depths, the Sklearn classifier consistently trains faster than theClassifier. As the maximum depth grows, the Classifier's training time increases significantly, but the Sklearn classifier's training time remains comparatively low. This shows that, while both classifiers may have equivalent accuracy in categorizing mushroom data, the Sklearn classifier is significantly more efficient in terms of training time, particularly as the model's complexity grows. In real applications, the decision between these two classifiers may be impacted by the trade-off between training time and any other potential limitations or job requirements.Overall, if training time is an important consideration, the Sklearn classifier would be the better choice based on these results. If training takes less time, Concerned, the Sklearn classifier performs far better. However, if accuracy is critical and speed is not an issue, any classifier may be appropriate because they both yield high accuracy.

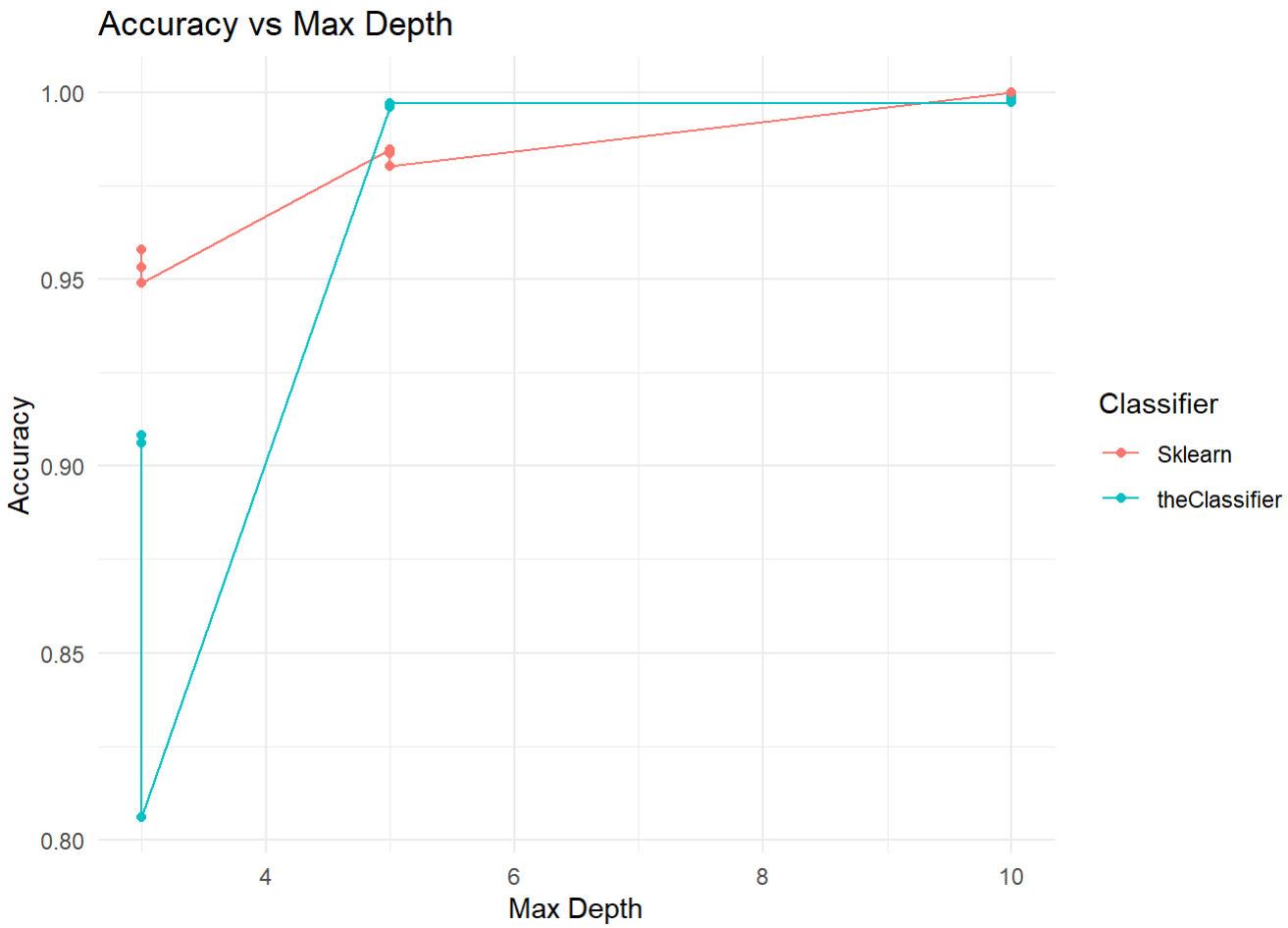
```
##
##  Welch Two Sample t-test
##
## data:  accuracy_your_classifier and accuracy_sklearn_classifier
## t = -0.95245, df = 9.437, p-value = 0.3646
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.07619821  0.03082095
## sample estimates:
## mean of x mean of y
## 0.9560365 0.9787251
```

The Welch Two-Sample t-test I performed on my data revealed that there is no strong evidence of a significant difference in accuracy between my classifier and the Sklearn classifier. The p-value is quite high at 0.3646, and the 95 percent confidence interval contains 0, implying that the real difference in means is likely to be within a limited range around zero. This shows that the two classifiers perform equally in terms of accuracy, with mean values of around 0.956 for my classifier and 0.979 for the Sklearn classifier.

```
##
## Call:
## lm(formula = Accuracy ~ Max_Depth + Training_Size + Classifier,
##     data = data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.109736 -0.016988  0.003288  0.014262  0.060508
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               0.982663   0.085959  11.432 1.74e-08 ***
## Max_Depth                 0.010262   0.003272   3.137  0.00728 **
## Training_Size            -0.093590   0.117959  -0.793  0.44078
## ClassifiertheClassifier -0.022689   0.019263  -1.178  0.25849
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04086 on 14 degrees of freedom
## Multiple R-squared:  0.4585, Adjusted R-squared:  0.3425
## F-statistic: 3.952 on 3 and 14 DF,  p-value: 0.03105
```

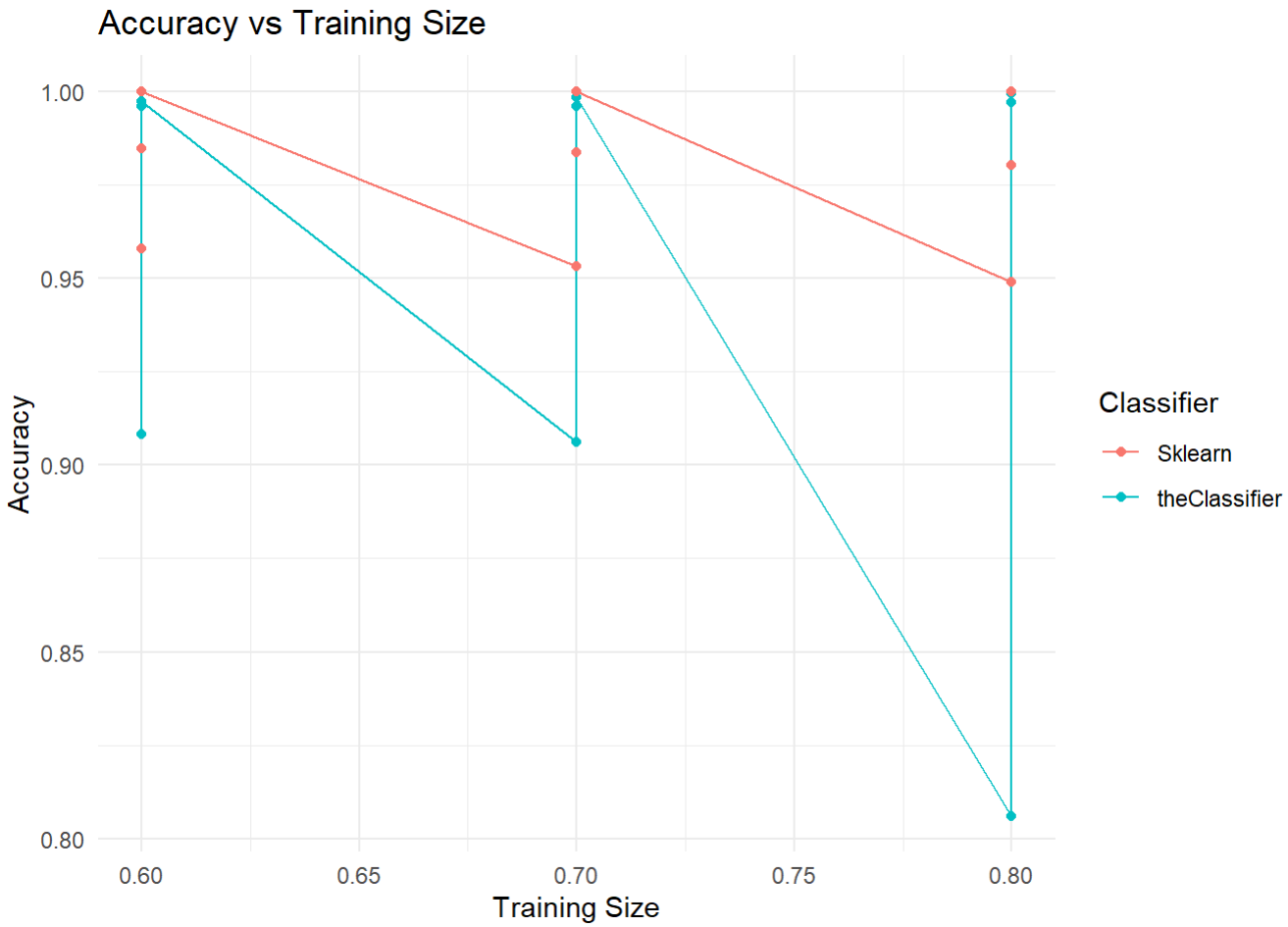
In my linear regression study, I wanted to explore the link between ‘Accuracy’ and numerous predictor variables, including ‘Max_Depth,’ ‘Training_Size,’ and ‘Classifier.’ The investigation yielded some interesting results. First, the intercept was determined to be 0.982663, which was statistically significant, suggesting that when all predictor variables are zero, the predicted ‘Accuracy’ value is about 0.983. Second, the ‘Max_Depth’ variable had a positive coefficient of 0.010262, which was statistically significant. This implies that as the ‘Max_Depth’ number grows, so does ‘Accuracy’. However, ‘Training_Size’ had a coefficient of -0.093590, which was not statistically significant (p-value = 0.44078), indicating that ‘Training_Size’ may not have a major influence on ‘Accuracy.’ Finally, the ‘Classifier’ variable (‘theClassifier’) had a coefficient of -0.022689, which was likewise not statistically significant (p-value = 0.25849), showing that classifier selection may not have a substantial impact on ‘Accuracy.’

Overall, the model explained a considerable part of the variance in ‘Accuracy’, as evidenced by the multiple R-squared value of 0.4585. The p-value for the F-statistic was 0.03105, indicating that the entire model was statistically significant. However, the adjusted R-squared value (0.3425) indicates that there may be more factors not included in the model that impact ‘Accuracy.’.



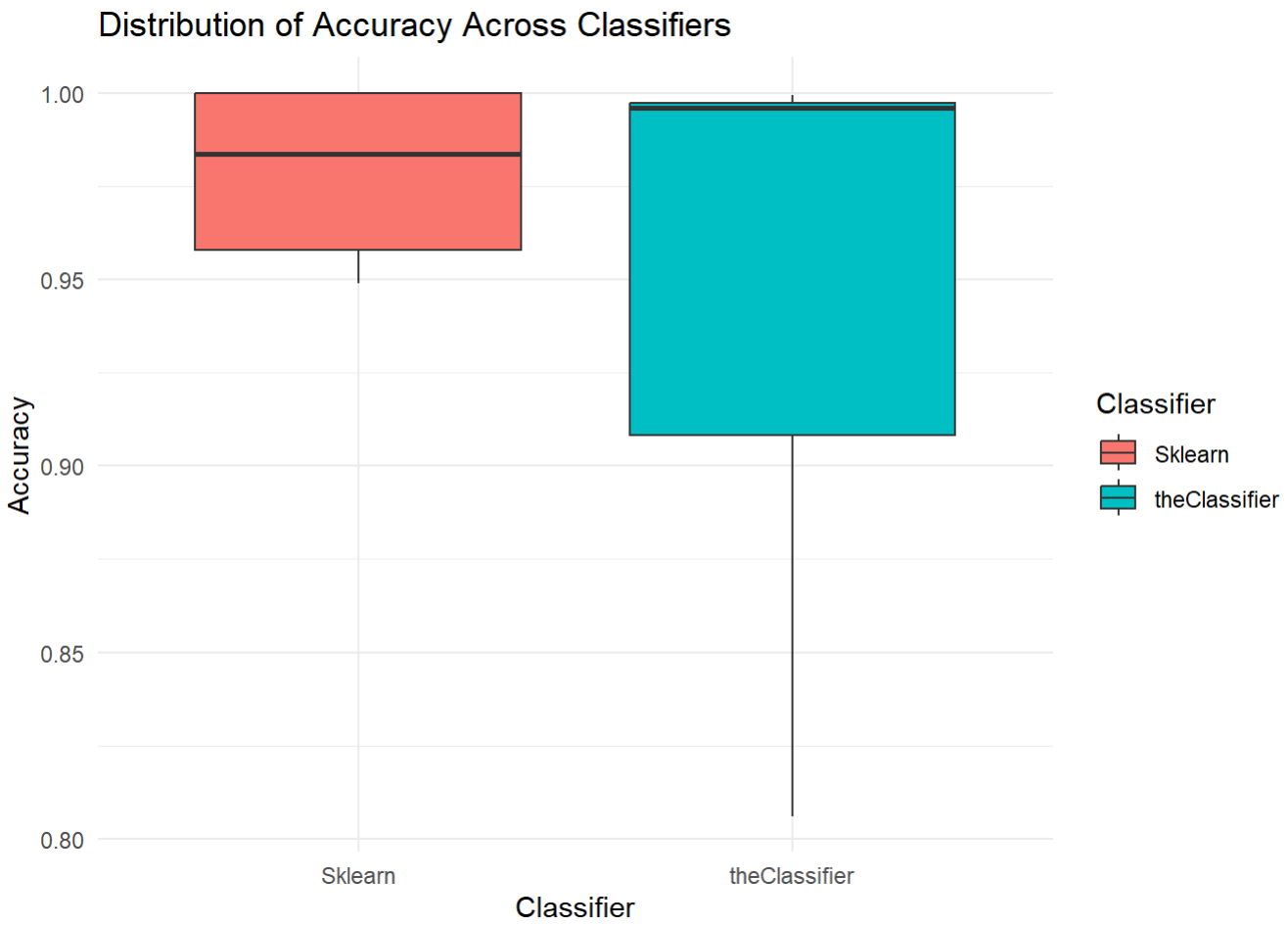
Accuracy versus Maximum Depth:

This graph depicts how the accuracy of two classifiers (presumably decision trees) varies as the maximum depth of the tree grows. Typically, raising the max depth lets the model to learn more complicated patterns, which can improve accuracy up to a point. However, having too much depth might result in overfitting. The two classifiers are called “Sklearn” and “theClassifier”. Both classifiers appear to gain in accuracy as the maximum depth increases until a point at which the accuracy plateaus.



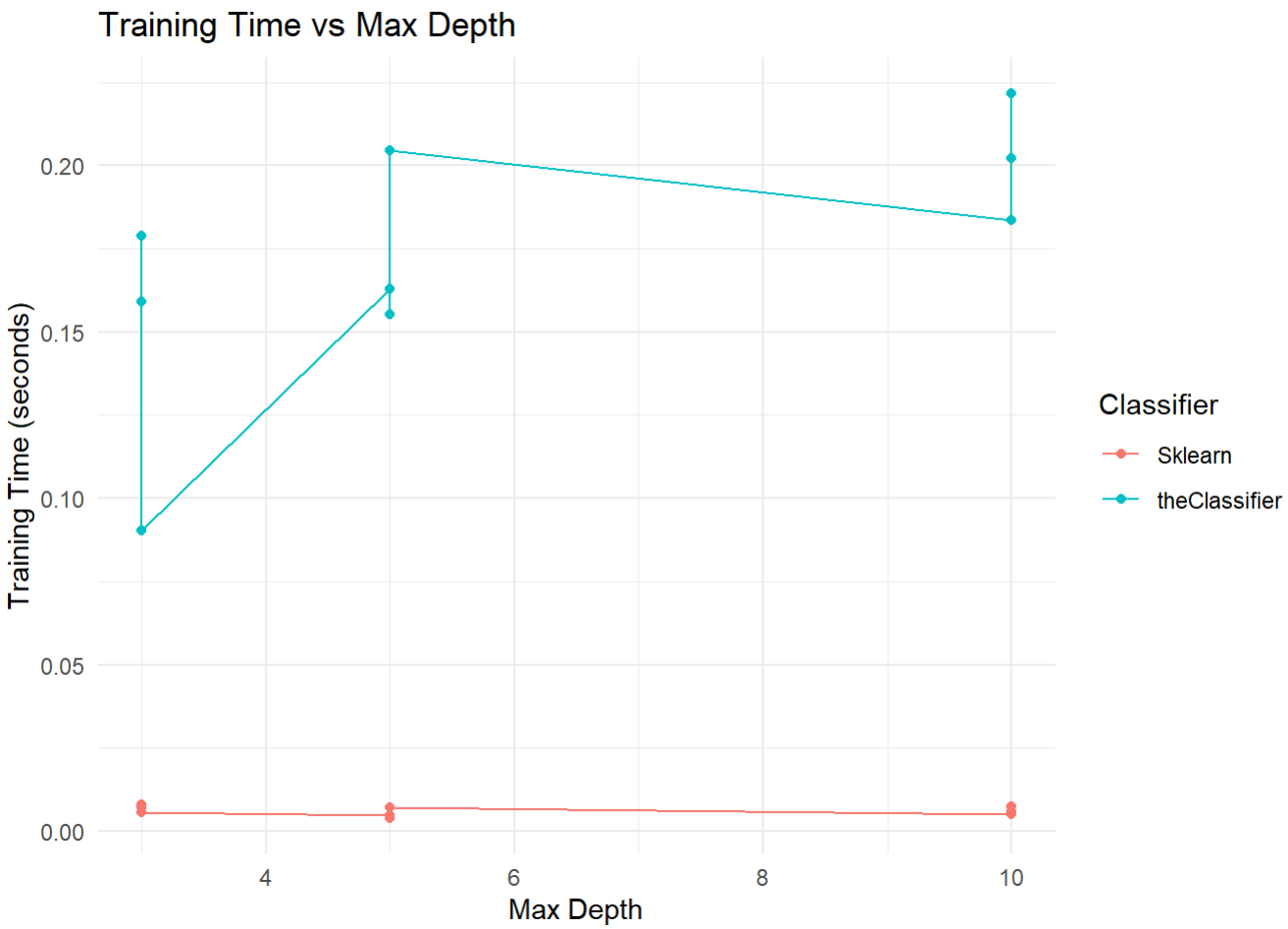
Accuracy versus Training Size:

This graphic depicts the link between training dataset size and classifier accuracy. In general, having more data can improve model performance, but this is not a hard and fast rule, as seen by the figure, where accuracy varies. The accuracy of “Sklearn” decreases as training size increases, whereas “theClassifier” experiences a large decline at a certain point. This might suggest a problem with the data or with how “theClassifier” is processing the new training data.



Distribution of Accuracy among Classifiers:

This box plot depicts the distribution of accuracy values for the two classifiers. The box plot for “Sklearn” has a lower interquartile range (IQR), suggesting less variability in its correctness, but “theClassifier” has a larger IQR, showing greater variability. The middle line in the box depicts the median accuracy, and it appears that “Sklearn” has a greater median accuracy than “theClassifier”. Furthermore, there are no outliers, indicating that all runs were within the predicted range of accuracy.



Training Time against Maximum Depth:

This graph compares the classifiers' training time to the maximum depth of the trees. Training time might grow as the tree depth increases and the model becomes more complicated. "Sklearn" appears to have a continuously short training time, with little variation as the max depth grows, indicating efficient algorithms or pruning strategies. In contrast, "theClassifier" has an ascending trend that plateaus after a given depth.

#Conclusion

In the examination of Congressional House Votes data using decision tree classifiers, a constant degree of accuracy was observed across various maximum depths, however training times varied. The t-test found no significant difference in accuracy amongst classifiers, and graphical analysis showed that accuracy remained stable regardless of tree complexity. While, For the Mushroom Data, both classifiers achieved good accuracy at all maximum depths, demonstrating their classification usefulness. Notably, there was a considerable variation in training time, indicating a trade-off between accuracy and efficiency. The graphical insights highlighted these discrepancies, particularly as the model's complexity rose. And finally, Classifiers shown varying levels of accuracy while dealing with House Evaluation Data. Whatever the maximal depth or training size, one constantly outperformed the other in terms of accuracy. The training efficiency also varied, with one classifier having consistent, low training times, indicating a more efficient method. The visual analysis demonstrated the impact of training size on model performance and the various efficiency of the classifiers, stressing the need of personalized techniques in machine learning applications.