COSC 757 Assignment 2: Classification

Daphnc McWilliams
Department of Computer Science
Towson University
Towson, MD
dmcwil2@students.towson.edu

***Abstract*—The purpose of this paper is to compare the results of the decision tree, Naïve Bayes, and logistic regression classification methods when applied to UCI's Mushroom Data Set.**

I. Introduction

A. *The Data Set*

This assignment uses the Mushroom Data Set that was contributed to UCI's Machine Learning Repository in 1987. This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushroom from the Agaricus and Lepiota families. These descriptions were drawn from *The Audubon Society Field Guide to North American Mushrooms* (1981).

The field guide classifies each species of mushroom is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. For the purposes of this data set, mushrooms of this last category ("unknown edibility") are lumped in with the definitely poisonous ones. The field guide states that there's no simple rule for determining the edibility of a mushroom (for example, like the "leaflets three, let it be" rule for Poisonous Oak and Ivy), hence the usefulness of a reliable classification model built from this dataset.

There are 8,124 individual rows in this data set. Each mushroom in this data set is classified as either edible or poisonous and has various other characteristics that help in determining the edibility of the mushroom.

B. *Variables*

There are 22 variables in this data set in addition to a class variable that categorizes each sample as either edible ("e") or poisonous ("p"). They are listed below in Table 1 along with their values.

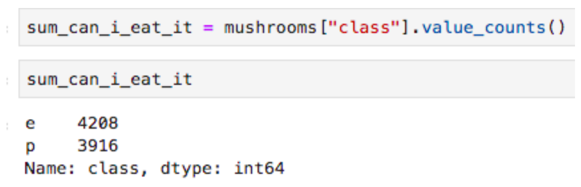
TABLE I. MUSHROOM DATA SET VARIABLES

1. class	poisonous = p, edible = e
2. cap-shape	bell = b, conical = c, convex = x, flat = f, knobbed = k, sunken = s
3. cap-surface	fibrous = f, grooves = g, scaly = y, smooth = s
4. cap-color	brown = n, buff = b, cinnamon = c, gray = g, green = r, pink = p, purple = u, red = e, white = w, yellow = y
5. bruises?	bruises = t, no = f
6. odor	almond = a, anise = l, creosote = c, fishy = y, foul = f, musty = m, none = n, pungent = p, spicy = s
7. gill-attachmnet	attached = a, descending = d, free = f, notched = n
8. gill-spacing	close = c, crowded = w, distant = d
9. gill-size	broad = b, narrow = n
10. gill-color	black = k, brown = n ,buff = b ,chocolate = h, gray = g, green = r, orange = o, pink = p, purple = u, red = e, white = w, yellow = y
11. stalk- shape	enlarging = e, tapering = t
12. stalk-root	bulbous = b, club = c, cup = u, equal = e, rhizomorphs = z, rooted = r, missing = ?
13. stalk-surface-above-ring	fibrous = f, scaly = y, silky = k, smooth = s
14. stalk-surface-below-ring	fibrous = f, scaly = y, silky = k, smooth = s
15. stalk-color-above-ring	brown = n, buff = b, cinnamon = c, gray = g, orange = o, pink = p, red = e, white = w, yellow = y
16. stalk-color-below-ring	brown = n, buff = b, cinnamon = c, gray = g, orange = o, pink = p, red = e, white = w, yellow = y
17. veil-type	Partial = p, universal = u
18. veil-color	brown = n, orange = o, white = w, yellow = y
19. ring-number	none = n, one = o, two = t
20. ring-type	cobwebby = c, evanescent = e, flaring = f, large = l, none = n, pendant = p, sheathing = s, zone = z
21. spore-pint-color	black = k, brown = n, buff = b, chocolate = h, green = r, orange = o, purple = u, white = w, yellow =y
22. population	abundant = a, clustered = c, numerous = n, scattered = s, several = v, solitary =y
23. habitat	grasses = g, leaves = l, meadows = m, paths = p, urban = u, waste = w, woods = d

All variables in this data set are categorical. Additionally, there are no missing or null values, making data preprocessing a simple task.

C. *Exploratory Data Analysis*

This is a large data set. Some preliminary data visualization was conducted to gain insights into the data set and make it easier to ask the right questions moving forward. Exploratory data analysis was performed using Python's pandas, numpy, and matplotlib libraries in Jupyter Lab. Fig. 1 below shows the proportion of poisonous to edible mushrooms in the target variable. As shown, there are 4,208 edible mushrooms (51.8%) against 3,916 poisonous mushrooms (48.2%) in the data set. There is fairly even balance between the two classes.



```
sum_can_i_eat_it = mushrooms["class"].value_counts()
```

```
sum_can_i_eat_it
```

```
e    4208
p    3916
Name: class, dtype: int64
```

Fig. 1. Edible vs. Poisonous Mushrooms

Some simple data visualization revealed which variables might be used as reliable criteria in determining whether a mushroom is edible or poisonous and might play a significant role in creating decision trees later on. The normalized bar graphs for these variables vs. the target variable are shown below.

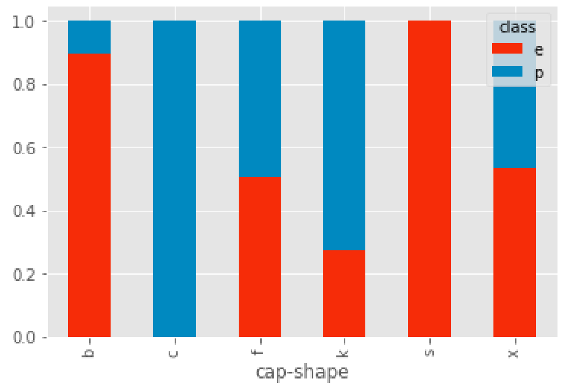


Fig. 2. Cap Shape vs. Target Variable

As shown in Fig. 2, it can be safely assumed that mushrooms with a sunken cap shape are edible, and that those with a conical cap shape are definitely poisonous. Other cap shapes cannot be taken on their own as reliable indicators of the mushroom's edibility.

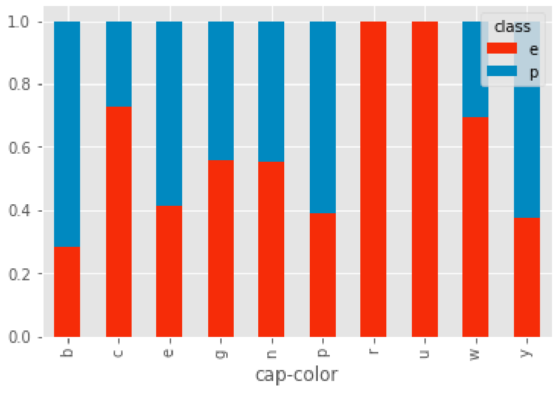


Fig. 3. Cap Color vs. Target Variable

Fig. 3 indicates that mushrooms with green or purple caps are safe to eat. Mushrooms with any other cap color should be approached with caution.

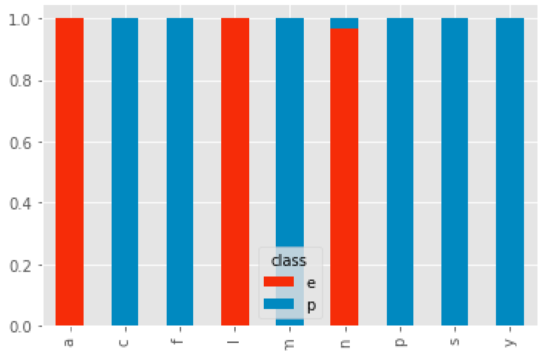


Fig. 4. Odor vs. Target Variable

As Fig. 4 shows, of the variables in this data set, odor is the most reliable indicator of edibility. Mushrooms with an almond or anise-like smell can safely be eaten. All other scents are better avoided. A mushroom with no scent should be carefully checked to see if it fits other criteria for edibility.

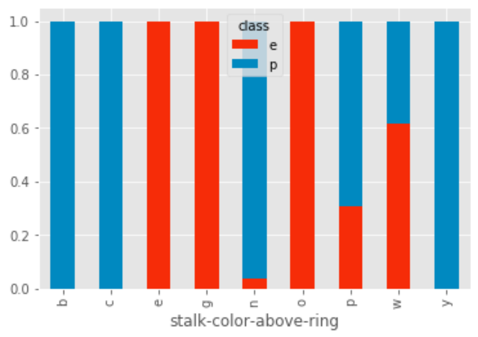


Fig. 5. Stalk Color Above Ring vs. Target Variable

As shown in Fig. 5, stalk color above the ring is another fairly good indicator of edibility. Brown, cinnamon, and yellow stalk colors should be avoided, while mushrooms with red, grey, and orange clored stalks above the ring are safe to eat. The visualization for stalk color below the ring showed the exact same results, so a separate diagram for that variable hasn't been included in this report.

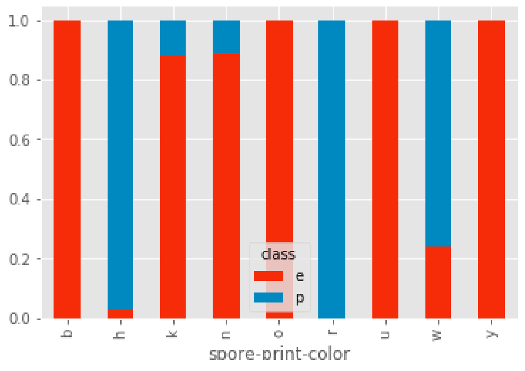


Fig. 6. Spore Print Color

Lastly, as can be seen in Fig. 6, mushrooms with brown, orange, yellow, and purple sport prints are safe to eat. Mushrooms with green spore prints are definitely not edible, while the remaining colors should be cross-checked against other edibility criteria.

II. Methodology

Classification was undertaken using the decision tree, Naïve Bayes, and logistic regression methods. RapidMiner was used to perform the various classification methods. Data was partitioned into testing and training sets using the holdout method (70% of the data set was partitioned for training, and 30% was partitioned for testing).

III. Experimental Set-up and Results

A. Decision Trees

A decision tree classifies given input into one of its possible classes using a flow-chart structure. The idea behind decision trees is that the features of a data set can be used to create yes/no questions to continually split the data set until every data point belonging to each class is isolated and labeled. The ideal tree is the smallest tree possible (i.e., with the fewest splits) that can accurately classify all data points.

The basic decision tree algorithm uses a greedy, non-backtracking approach to construct a tree in a recursive, divide-and-conquer manner. A greedy approach makes locally optimal decision to the pick the feature that should be used in each split, instead of trying to make the best overall decision. It focuses only on the node at hand, and what's best for that node in particular instead of exploring all possibilities.

On every split, the algorithm tries to partition the data set into the smallest subsets possible. A number of methods can be used to determine the best attribute for partitioning the data at each level; three popular methods, which are also attempted in this report, are information gain (ID3), gain ratio (C4.5), and the Gini Index (CART).

Information gain involves calculating the entropy of every attribute in the data set. In the context of information gain, entropy refers to how much variance the data has; it is the degree of uncertainty, impurity, or disorder of a random variable. The original set is then split into subsets using the attribute for which the resulting entropy after splitting is minimum, or information gain is maximum.

Information gain is biased towards attributes with a large number of values; however, this problem can be overcome by using gain ratio as a partitioning method. Gain ration is used to normalize of information gain of an attribute against how much entropy that attribute has. The attribute with the maximum gain ratio is selected as the splitting attribute.

Lastly, the Gini Index is the measure of variance across the different classes. It calculates the probability of a specific randomly selected feature that is classified incorrectly. The Gini Index ranges between 0 and 1, where 0 represents purity of the classification and 1 denotes random distribution of elements among various classes. The attribute with the lowest Gini Index value (i.e., that provides the largest reduction in impurity) is chosen to split the node.

A decision tree was created from the mushrooms data set using each of these methods, and the results are detailed below. A maximum tree depth of 10 was selected, and pruning and prepruning were applied to avoid overfitting.

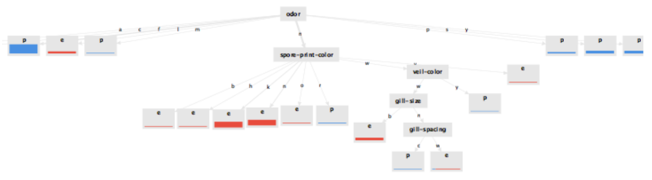
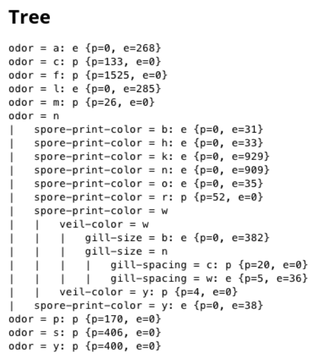


Fig. 7. Gain Ratio Decision Tree

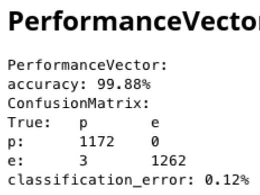


Tree

```
odor = a: e {p=0, e=268}
odor = c: p {p=133, e=0}
odor = f: p {p=1525, e=0}
odor = l: e {p=0, e=285}
odor = m: p {p=26, e=0}
odor = n
|   spore-print-color = b: e {p=0, e=31}
|   spore-print-color = h: e {p=0, e=33}
|   spore-print-color = k: e {p=0, e=929}
|   spore-print-color = n: e {p=0, e=909}
|   spore-print-color = o: e {p=0, e=35}
|   spore-print-color = r: p {p=52, e=0}
|   spore-print-color = w
|   |   veil-color = w
|   |   |   gill-size = b: e {p=0, e=382}
|   |   |   gill-size = n
|   |   |   |   gill-spacing = c: p {p=20, e=0}
|   |   |   |   gill-spacing = w: e {p=5, e=36}
|   |   veil-color = y: p {p=4, e=0}
|   spore-print-color = y: e {p=0, e=38}
odor = p: p {p=170, e=0}
odor = s: p {p=406, e=0}
odor = y: p {p=400, e=0}
```

Fig. 8. Description of the Gain Ratio Decision Tree

Figs. 7 and 8 show the results of creating a decision tree with gain ratio as a partitioning method. Odor was taken as the initial splitting attribute, then spore print color followed by veil color, gill size, and gill spacing.



PerformanceVector

```
PerformanceVector:
accuracy: 99.88%
ConfusionMatrix:
True:   p       e
p:      1172    0
e:      3       1262
classification_error: 0.12%
```

Fig. 9. Performance of Gain Ratio Decision Tree

As can be seen in Fig. 9, the decision tree with gain ratio partitioning worked very well as a classificaiton method on the mushrooms data set. The confusion matrix shows that all poisonous mushrooms were correctly classified, and only three edible mushrooms were incorrectly classified as poisonous, with an overall accuracy rate of 99.88%. The sensitivity (true positive recognition rate for poisonous) was 100% (1172/1172). The specificity (true negative recognition rate for not poisonous, or edible) was 99.76% (1262/1265). The precision (exactness, or % of tuples correctly labeled as positive/poisonous) was 99.74% (1172/1175), and recall (completeness, or % of positive tuples labeled as positive) was 100% (1172/1172).

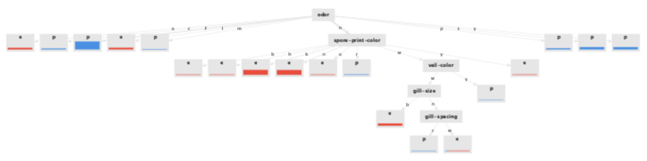
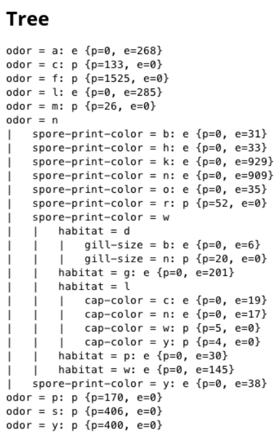


Fig. 10. Information Gain Decision Tree

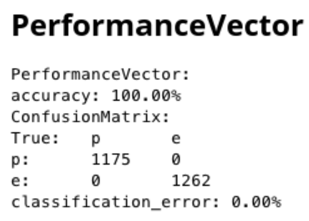


Tree

```
odor = a: e {p=0, e=268}
odor = c: p {p=133, e=0}
odor = f: p {p=1525, e=0}
odor = l: e {p=0, e=285}
odor = m: p {p=26, e=0}
odor = n
|   spore-print-color = b: e {p=0, e=31}
|   spore-print-color = h: e {p=0, e=33}
|   spore-print-color = k: e {p=0, e=929}
|   spore-print-color = n: e {p=0, e=909}
|   spore-print-color = o: e {p=0, e=35}
|   spore-print-color = r: p {p=52, e=0}
|   spore-print-color = w
|   |   habitat = d
|   |   |   gill-size = b: e {p=0, e=6}
|   |   |   gill-size = n: p {p=20, e=0}
|   |   habitat = g: e {p=0, e=201}
|   |   habitat = l
|   |   |   cap-color = c: e {p=0, e=19}
|   |   |   cap-color = n: e {p=0, e=17}
|   |   |   cap-color = w: p {p=5, e=0}
|   |   |   cap-color = y: p {p=4, e=0}
|   |   habitat = p: e {p=0, e=30}
|   |   habitat = w: e {p=0, e=145}
|   spore-print-color = y: e {p=0, e=38}
odor = p: p {p=170, e=0}
odor = s: p {p=406, e=0}
odor = y: p {p=400, e=0}
```

Fig. 11. Description of Information Gain Decision Tree

Figs. 10 and 11 show the results of creating a decision tree with information gain as a partitioning method. Once again, odor was taken as the initial partitioning attribute, followed by spore print color, habitat, and finally either gill size or cap color.



PerformanceVector

```
PerformanceVector:
accuracy: 100.00%
ConfusionMatrix:
True:   p       e
p:      1175    0
e:      0       1262
classification_error: 0.00%
```

Fig. 12. Performance of Information Gain Decision Tree

It's clear from Fig. 12 that information gain worked very well as a partitioning method when creating decision trees from the mushroom data set. There were no classification errors, and the overall accuracy was 100% as a result. The sensitivity, specificity, precision, and recall were all 100%.

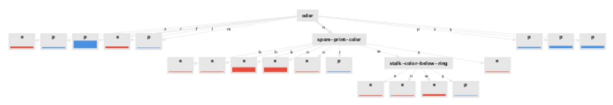
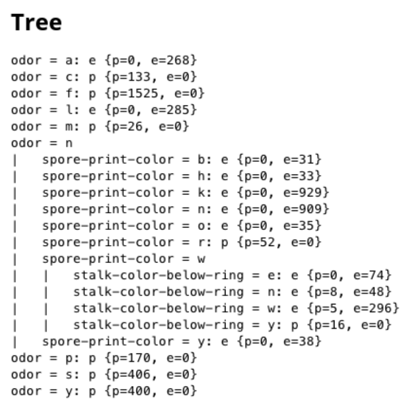


Fig. 13. Gini Index Decision Tree

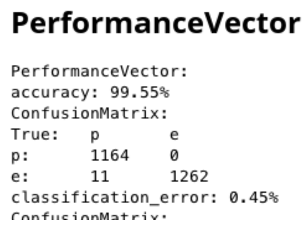


Tree

```
odor = a: e {p=0, e=268}
odor = c: p {p=133, e=0}
odor = f: p {p=1525, e=0}
odor = l: e {p=0, e=285}
odor = m: p {p=26, e=0}
odor = n
|   spore-print-color = b: e {p=0, e=31}
|   spore-print-color = h: e {p=0, e=33}
|   spore-print-color = k: e {p=0, e=929}
|   spore-print-color = n: e {p=0, e=909}
|   spore-print-color = o: e {p=0, e=35}
|   spore-print-color = r: p {p=52, e=0}
|   spore-print-color = w
|   |   stalk-color-below-ring = e: e {p=0, e=74}
|   |   stalk-color-below-ring = n: e {p=8, e=48}
|   |   stalk-color-below-ring = w: e {p=5, e=296}
|   |   stalk-color-below-ring = y: p {p=16, e=0}
|   spore-print-color = y: e {p=0, e=38}
odor = p: p {p=170, e=0}
odor = s: p {p=406, e=0}
odor = y: p {p=400, e=0}
```

Fig. 14. Description of Gini Index Decision Tree

Like in the previous two decision trees, the Gini Index tree begins splitting the data with the odor attribute. It then uses spore print color and stalk color below ring to partition the remaining results.



PerformanceVector

```
PerformanceVector:
accuracy: 99.55%
ConfusionMatrix:
True:   p       e
p:      1164    0
e:      11      1262
classification_error: 0.45%
ConfusionMatrix:
```

Fig. 15. Performance of Gini Index Decision Tree

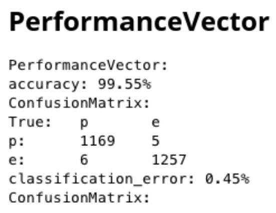
Fig. 15 shows the overall performance of a decision tree created with Gini Index partitioning. This tree also performed very well on the mushroom data set, with an accuracy rate of 99.55%, but it was the weakest of the three methods attempted. As can be seen in the confusion matrix, this model classified all of the poisonous mushrooms correctly, but incorrectly labeled 11 edible mushrooms as poisonous. The sensitivity was 100% (1164/1164). The specificity was 99.14% (1262/1273). The precision was 99.06% (1164/1175), and recall was 100% (1164/1164).

Overall, decision trees worked well as a classification method for this dataset. Odor was chosen as the first splitting attribute for each type of decision tree, which was expected given the results of the data visualizations in section II. Spore print color was also identified as a potentially significant variable in the exploratory data analysis in section II, and it was used as the second splitting attribute in each of the three decision trees.

B. *Naïve Bayes*

The second classification method attempted on this data set was Naïve Bayes. Naïve Bayes is a probabilistic classifier based on Bayes theorem. It assumes strong independence between features, in other words, that the presence of a feature in a class is unrelated to the presence of any other feature. It also assumes that all predictors have an equal effect on the outcome.

Bayes theorem provides a way of calculating the probability from the prior probability of a class (i.e., target) given a predictor (i.e., attribute), the prior probability of the predictor, and the probability of the predictor given class.



PerformanceVector

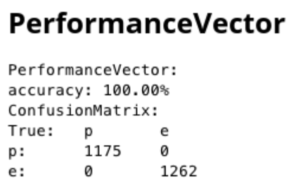
```
PerformanceVector:
accuracy: 99.55%
ConfusionMatrix:
True:   p       e
p:      1169    5
e:      6       1257
classification_error: 0.45%
ConfusionMatrix:
```

Fig. 16. Results of Naïve Bayes Classification

Fig. 16 shows the results of using a Naïve Bayes model to classify the mushroom data. This model performed fairly well, with an accuracy rate of 99.55%, but is the only classification model so far to misidentify poisonous mushrooms as edible (5 out of 1262), which is concerning. It also misidentified six edible mushrooms as poisonous. The sensitivity was 100% (1169/1174). The specificity was 99.52% (1257/1263). The precision was 99.48% (1169/1175), and recall was 99.57% (1169/1174).

C. *Logistic Regression*

The final classification method attempted on this data set was logistic regression. Logistic regression is a statistical analysis method to predict binary outcomes such as "yes" or "no" (or in this case, "edible" or "poisonous") based on prior observations of a data set. In contrast to linear regression, which predicts continuous values, logistic regression is used when the data of the target value is categorical in nature. Logistic regression seemed like a good fit for this dataset, because the target variable is categorical binary and all other variables are categorical as well.


PerformanceVector

```
PerformanceVector:
accuracy: 100.00%
ConfusionMatrix:
True:   p       e
p:      1175    0
e:      0       1262
```

Fig. 17. Performance of Logistic Regression

Fig. 17 shows the overall results of using logistic regression as a classification model on this data set. It ranks next to the information gain decision tree as the most accurate classification model for this data set, and yielded no classification errors. The sensitivity, specificity, precision, and recall were all 100%.

IV. Conclusion

I used RapidMiner for the first time in completing this assignment and was somewhat surprised by the high accuracy rates for each of the models I used on this data set, leading me to believe that I've overlooked something or done something incorrectly. As mentioned earlier, the class variable is well balanced between edible and poisonous mushrooms (51.8 and 48.2%, respectively), so it seems unlikely that the high accuracy is due to a class imbalance problem. Overall, I found this data set relatively easy to work with compared to the forest fires data set I used in Assignment 1.

It was very easy to generate visuals for the decision trees using RapidMiner; however, I was not able to get it to generate all of the performance evaluation metrics I would have liked for each model, and had to resort to calculating sensitivity, specificity, recall, and precision by hand. This might have had something to do with the version of RapidMiner that was compatible with my computer. Given more time, it might have been worthwhile to compare the results of using RapidMiner to create classification models vs. Python's scikitlearn library.

References

[1] UCI Machine Learning Repository. (n.d.). *Mushrooms data set* [Data set]. https://archive.ics.uci.edu/ml/datasets/Mushroom

[2] Bento, C. (2021, June 28). *Decision tree classifier explained in rail life: Picking a vacation destination.* Towards Data Science. https://towardsdatascience.com/decision-tree-classifier-explained-in-real-life-picking-a-vacation-destination-6226b2b60575

[3] Nelson, D. (2019, May 10). *Overview of classifciation methods in Python with scikit learn.* StackAbuse. https://stackabuse.com/overview-of-classification-methods-in-python-with-scikit-learn/

[4] Ghandi, R. (2018, May 4). *Naïve Bayes classifier*. Towards Data Science. https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

[5] Kambria. (2019, July 9). *Logistic Regression for Machien Learning.* https://kambria.io/blog/logistic-regression-for-machine-learning/

[6] RapidMiner Documentation (n.d.). *Decision tree.* https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/trees/parallel_decision_tree.html

[7] RapidMinder Documentation (n.d.). *Naïve Bayers.* https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/bayesian/naive_bayes.html

[8] RapidMiner Documentation. (n.d.). *Logistic regression.* https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/logistic_regression/logistic_regression.html