Binary classification "Dataset Mushroom"

Realised by : Eya Haddad

Isra Moussaoui

The goal:

This project aims to create a binary classifier model able to classify whether the product 'mushroom' as 'poisonous' or 'edible' with the help of the dataset 'Mushroom' from the website 'UCI'.

About the dataset:

Subject Area : Biology

Data date : 26/04/1987

Data source : https://archive.ics.uci.edu/dataset/73/mushroom

Data set owner : Audubon Society

Dimension of the dataset : 22 variables

Dataset Characteristics : Multivariate

Number of features : 21 features

Number of records : 8124 records

Variable Name	Role	Type	Description
poisonous	Target	Categorical	edible=e, poisonous=p
cap-shape	Feature	Categorical	bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
cap-surface	Feature	Categorical	fibrous=f,grooves=g,scaly=y,smooth=s

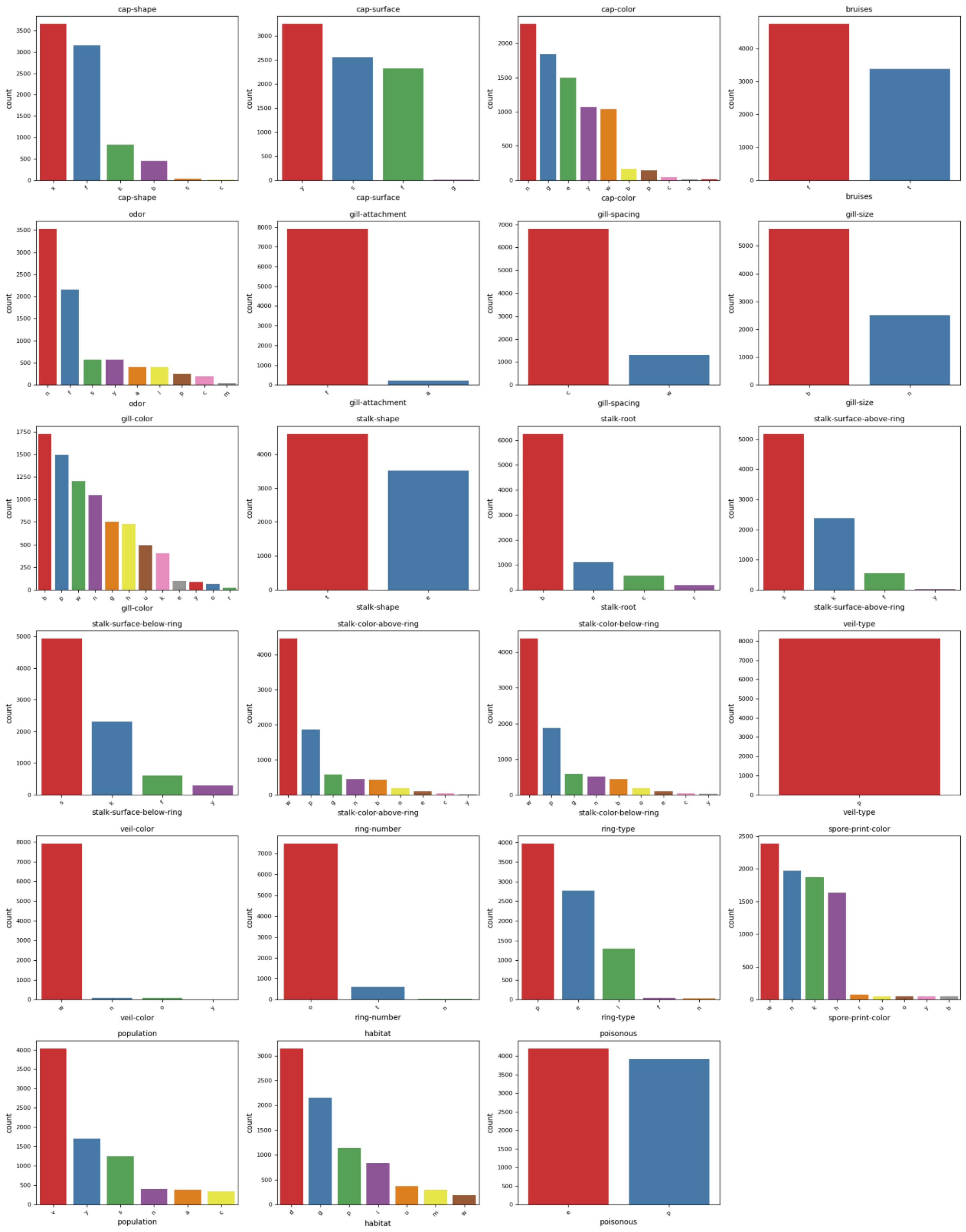
cap-color	Feature	Binary	brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p,purple=u,red=e,white=w,yellow=y
bruises	Feature	Categorical	bruises=t,no=f
odor	Feature	Categorical	almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s
gill-attachment	Feature	Categorical	attached=a,descending=d,free=f,notched=n
gill-spacing	Feature	Categorical	close=c,crowded=w,distant=d
gill-size	Feature	Categorical	broad=b,narrow=n
gill-color	Feature	Categorical	black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o,pink=p,purple=u,red=e, white=w,yellow=y
stalk-shape	Feature	Categorical	enlarging=e,tapering=t
stalk-root	Feature	Categorical	bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r,missing=?
stalk-surface-above-ring	Feature	Categorical	fibrous=f,scaly=y,silky=k,smooth=s
stalk-surface-below-ring	Feature	Categorical	fibrous=f,scaly=y,silky=k,smooth=s
stalk-color-above-ring	Feature	Categorical	brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
stalk-color-below-ring	Feature	Categorical	brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
veil-type	Feature	Binary	partial=p,universal=u
veil-color	Feature	Categorical	brown=n,orange=o,white=w,yellow=y

ring-number	Feature	Categorical	none=n,one=o,two=t
ring-type	Feature	Categorical	cobwebby=c,evanescent=e,flaring=f,large=l, none=n,pendant=p,sheathing=s,zone=z
spore-print-color	Feature	Categorical	black=k,brown=n,buff=b,chocolate=h,green=r, orange=o,purple=u,white=w,yellow=y
population	Feature	Categorical	abundant=a,clustered=c,numerous=n, scattered=s,several=v,solitary=y
habitat	Feature	Categorical	grasses=g,leaves=l,meadows=m,paths=p, urban=u,waste=w,woods=d

- ➔ All the features are categorical.
- ➔ ❌ 2480 missing values for the feature 'stalk-root'.
    - ◆ "stalk-root" is a categorical attribute so to solve this problem -> replace the missing value with the most frequent value of this attribute. We choose this method because the number of the missing values is important compared to the dataset so we can't delete them.

The data exploration :

Distribution of the categorical attributs of the dataset

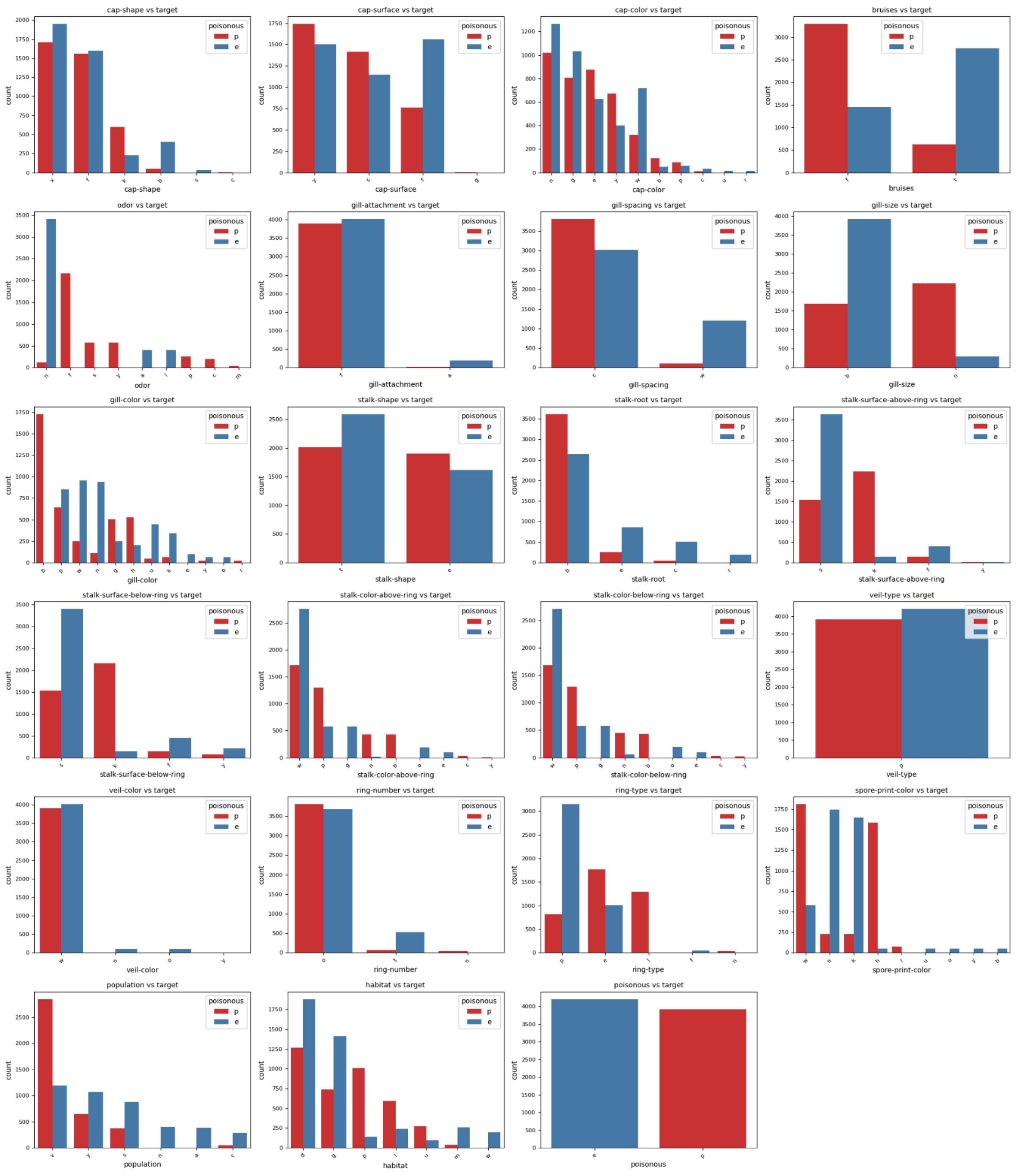


We have use the barplots for each variable to visualize their distribution :

- ➔ ❌ There is many attributes with rarely present values, their rate of presence is under 5% of the whole dataset size:
  - ◆ cap-shape with values 's' & 'c'.
  - ◆ cap-surface with value 'g'.
  - ◆ cap-color with values 'c' 'u & 'r'.
  - ◆ odor with value 'm'.
  - ◆ gill-attachement with value 'a'.
  - ◆ gill-color with values 'e' 'y' 'o' & 'r'.
  - ◆ stalk-root with value 'r'.
  - ◆ stalk-surface with value 'y'.
  - ◆ stalk-color with values 'o' 'e' c' & 'y'.
  - ◆ ❗ veil-types have only the value 'p', there's no 'e' (0%).
  - ◆ veil-color is mostly present (more than 90%) with the only value 'w', all the other possible values are rarely present under (10%).
  - ◆ ring-number with value 'n'.
  - ◆ ring-type with value 'f' & 'n'.
  - ◆ spore-print-color with values 'r' 'u' 'o' y' & 'b'.
- ➔ We can consider that those values doesn't have that much important for the classification decision.
- ➔ For the target variable 'poisonous', the distribution is quite equal with 48% class 'poisonous' and 52% class 'edible'.
  - ◆ ✅ We have a balanced dataset.

🚨 In our case, the class 'edible' is more important than 'poisonous' so the error rate have to be as small as possible in this class. In fact, if there is a poisonous mushroom that is classified as edible is more dangerous than the opposite situation when the mushroom is classified as poisonous or in reality is not.

Comparaison of all the features with the target values

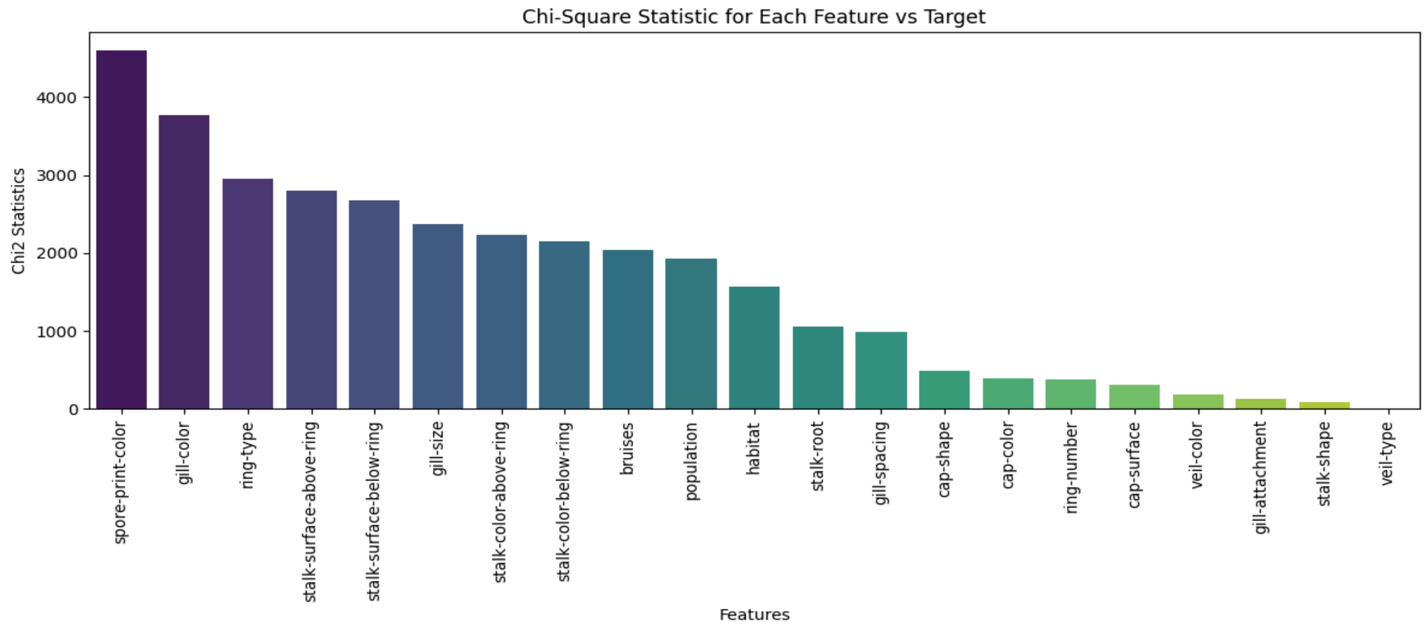


We have use the barplots to visualizes the relationship between mushroom attributes and the target whether the mushroom is edible (e) or poisonous (p) :

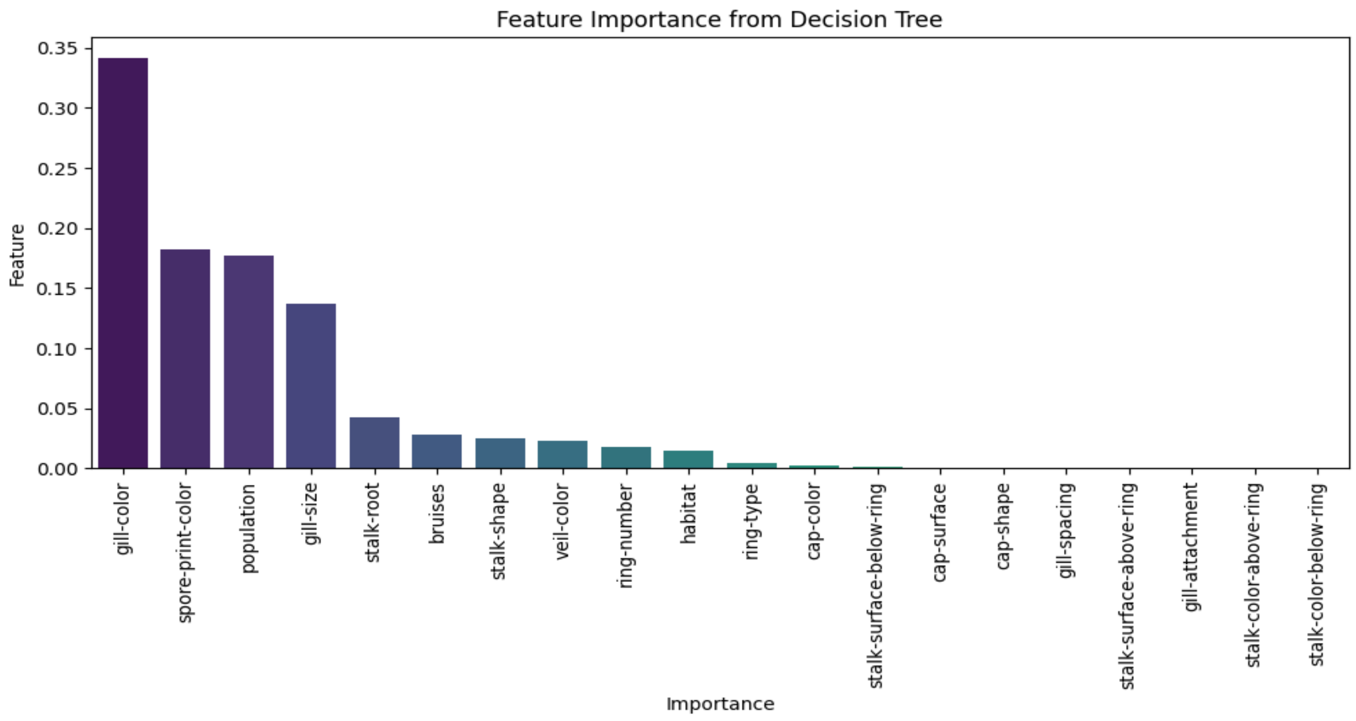
- ➔ Some variables are strong indicators for the predication :
    - ◆ 'odor' like f, s, y suggest poisonous mushrooms, and the n, a, l suggest edibility, especially n which dominates the edible category.
    - ◆ 'gill color' like b,e,r,o are strongly associated with one class.
    - ◆ 'stalk-color' like g, o, e for the edible class and c , y , b for the poisonous class.
    - ◆ 'veil-type' has the one and only value common in either class 'p', the other value 'e' is absolutely absent.
    - ◆ 'spore-print-color' for the values (u, o, y, b) are absolutely associated with class 'edible'.
    - ◆ same for the variable 'population' with the values a,c,n.
- ➔ We notice that the attributes 'odor', 'stalk-color' and 'spore-print-color' are very correlated with target ➡️ We can take them off to make the dataset more complex.

Feature selection :

- ➔ Since we are dealing with categorical data, the standard correlation matrix used for continuous variables is not appropriate. Instead, to assess whether two categorical variables are statistically related, we use the Chi-Square test of independence, which is specifically designed for this type of data.



- ➔ The only insignificant feature to drop is ‘veil-type’ because its ‘p-value’ is large (< 0.05) which means that this feature is absolutely not related to the target.
  - ◆ The ‘p-value’ tells us how "rare" the observed result is if we assume that there is no real relationship between the variables.
  - ◆ So we have kept the significant variables to our target that have a small p-value (< 0.05) → the link between the variable and the target is statistically significant.
- ➔ For the feature selection, we have also done the tree decision classifier because this model has the ability to automatically detect which variables have the most influence on target prediction.



- ➔ Label encoding : Because this model expects numerical inputs, not strings, we need to encode categorical data properly. However, decision trees have an advantage: they don’t treat these integers as continuous values. They only use them to split the data based on category equality.
  - ◆ So that we have used the ‘OrdinalEncoder’ that assigns a unique integer to each category per feature.
- ➔ Feature importance : The tree splits the data based on the features that provide the most Gini impurity reduction (each split reduces uncertainty about the target).

✅ Features used in early and frequent splits are typically more important, so we select them by applying a threshold (0.01) for the importance.

⚠️ The graphic visualisation of this Tree Decision is in the python file.

⛔ After the feature selection, we left with 10 important features that seem to be potentially helpful for the predication.

The supervised model KNN with parameters :

Step 1 : construction model

➔ K-Nearest Neighbors is a distance-based algorithm :

- It requires numerical input. That's why OneHotEncoding is the right choice for categorical features, especially when there's no meaningful order.
- As it allows all variables to be scaled equally, the data has to be standardized by removing the mean and dividing it by the standard deviation, resulting in reduced centered variables. (StandardScaler)

➔ The data split is based on the stratification to preserve class distribution (balanced dataset), 30% for the test set and 70% for the train set.

➔ Some metrics for the categorical data including in the gridsearch :

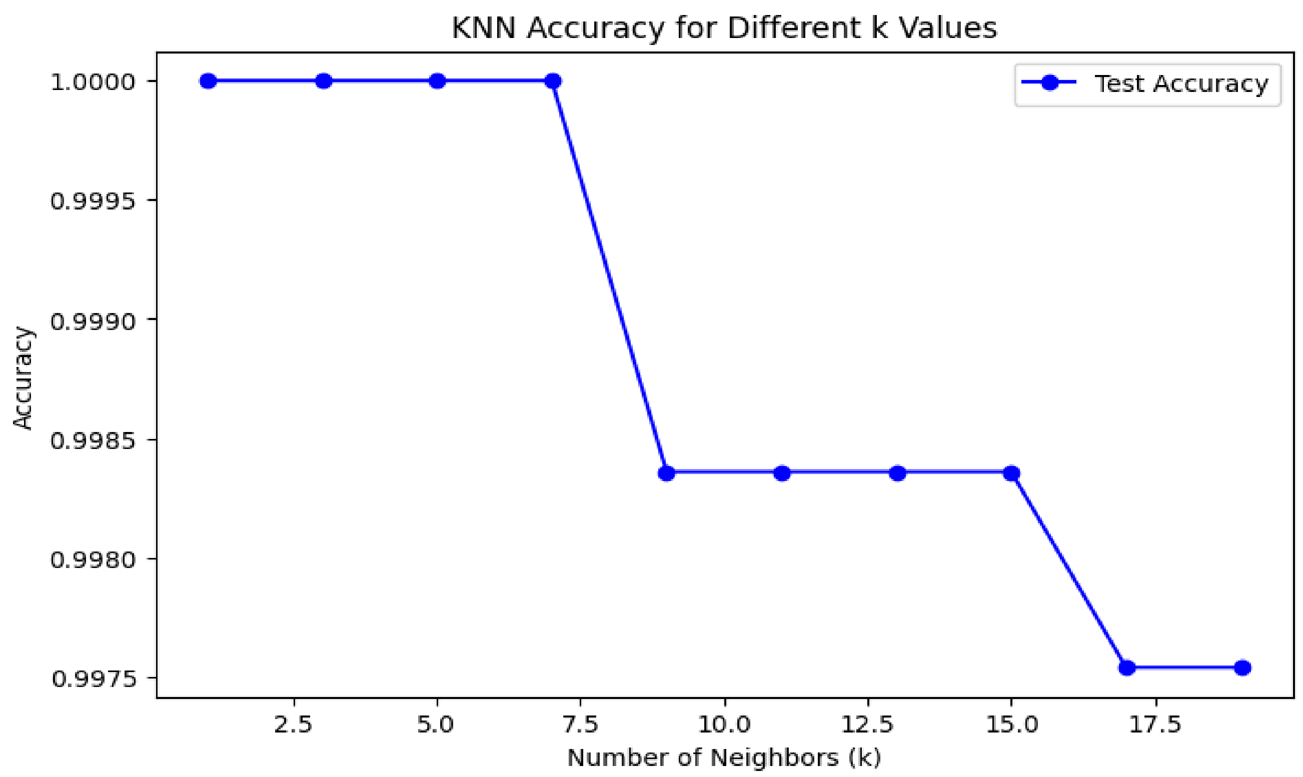
- Hamming Distance : Counts the number of different positions between two vectors. Perfect for binary variables.
- Jaccard Distance : measures the dissimilarity between two data sets by calculating the ratio of the intersection to the union of the sets.
- Rogerstanimoto Distance : is a variation of the Hamming distance, taking into account different types of attributes.

➔ Best Hyperparameters according to the accuracy of each model :

- metric : hamming
- number of neighbors : 7
- weights : uniform

➔ Training of the model with the best parameters determined.

The curve of the Knn Accuracy with different k values from 1 to 20.

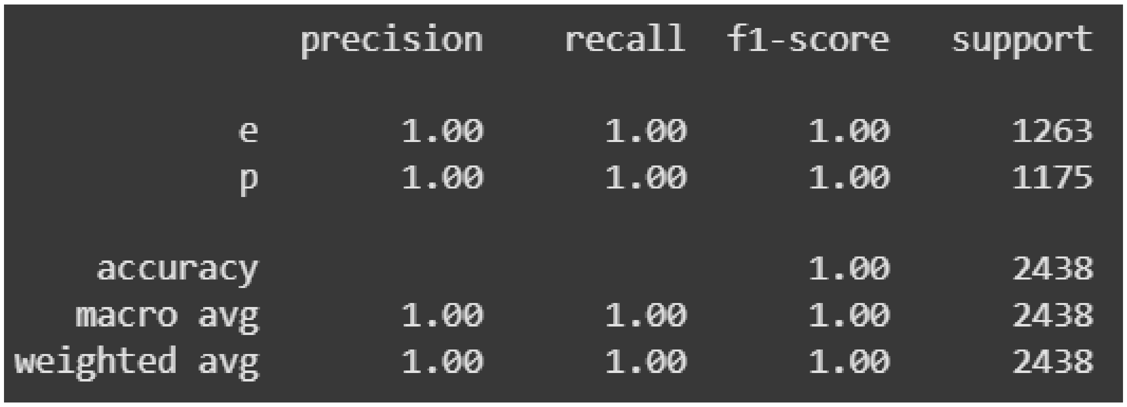


Step 2 : validation model

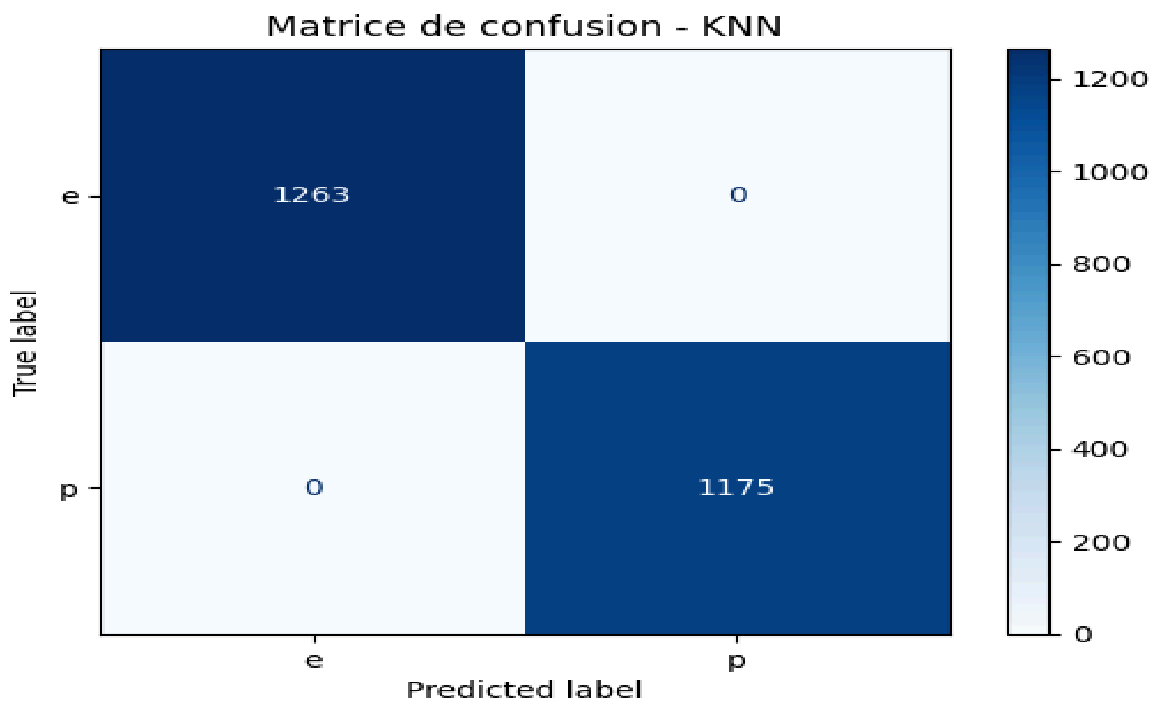
➔ After the construction of the model, we do the prediction with the best model using the test set.

➔ We can **evaluate the model performance** with different metrics :

- ◆ The classification report shown a great performance of this model:
    - **precision:** 100% of predictions for this class that are correct.
    - **Rappel**: 100% of true examples in this class correctly predicted.
    - **F1-score**: 100% harmonic mean of precision/recall.
    - **Support:** number of true examples in the test set.

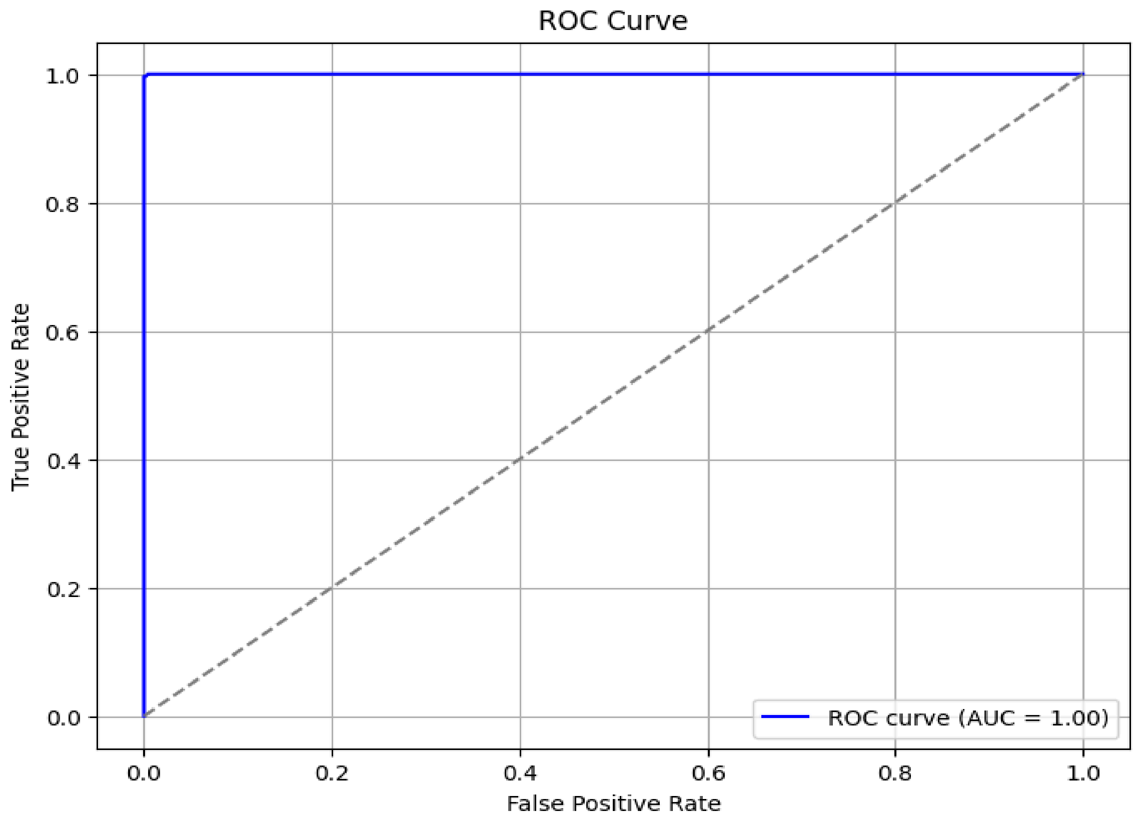


```
              precision    recall  f1-score   support

           e       1.00      1.00      1.00      1263
           p       1.00      1.00      1.00      1175

    accuracy                           1.00      2438
   macro avg       1.00      1.00      1.00      2438
weighted avg       1.00      1.00      1.00      2438
```



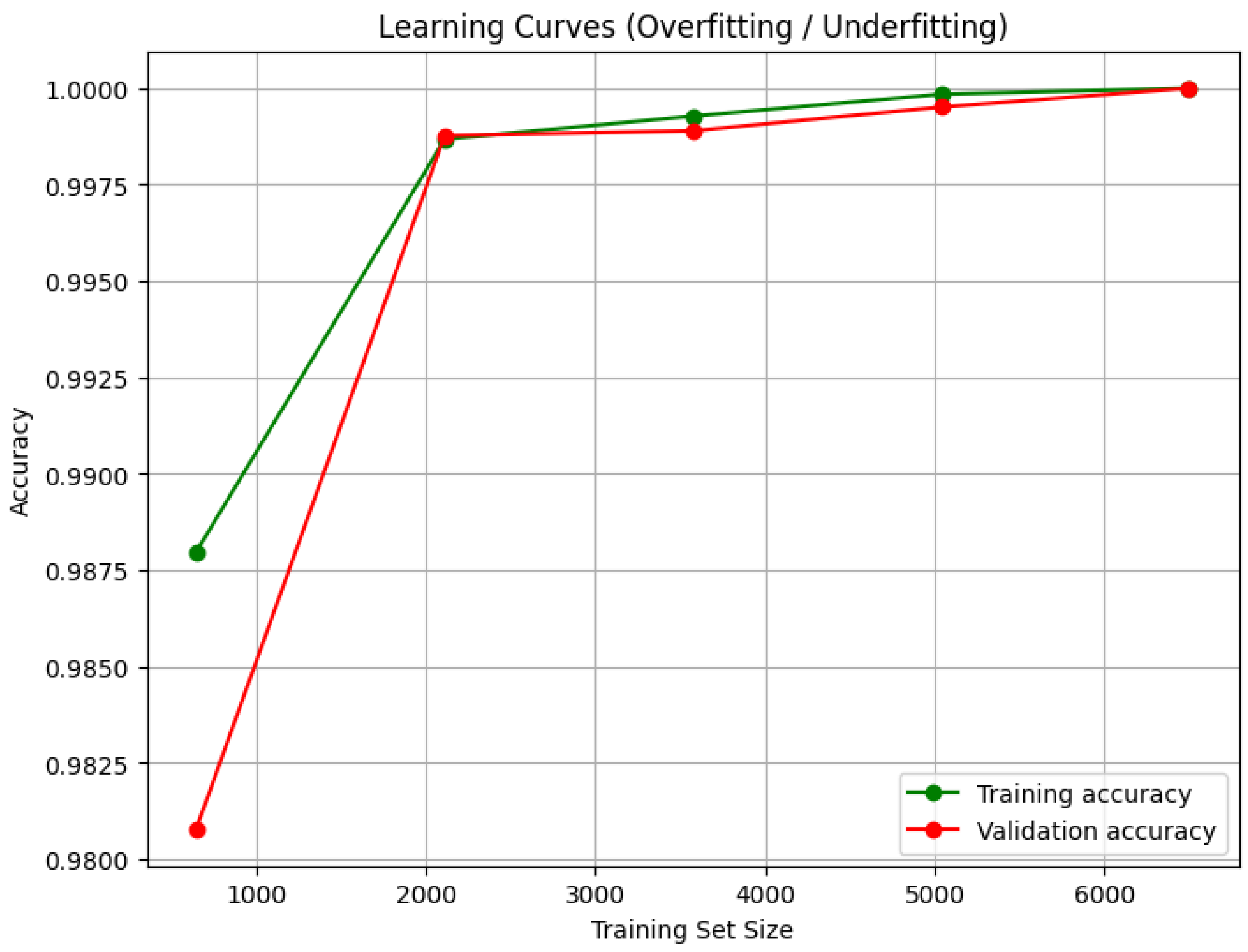
➔ **0 false ‘poisonous’ and 0 false ‘edible’ : 0 error prediction of the classification.**

✅ This is a perfect classifier on the evaluation dataset



➔ The ROC curve reaches the top-left corner of the plot (TPR = 1, FPR = 0) and stays flat at the top, which is **ideal behavior**.

➔ The Area Under the Curve is 1.00, indicating **perfect classification** -> the model distinguishes all positive samples from negative ones with zero error.



➔ Both curves are similar with the same variation as more data is added. Their accuracy is very important (from 98% to 100%).

➔ The gap between the two lines is quietly small, meaning the model generalizes well with minimal overfitting.

The result obtained by the KNN model is correctly predicted 100% of the labels during **5-fold cross-validation**.

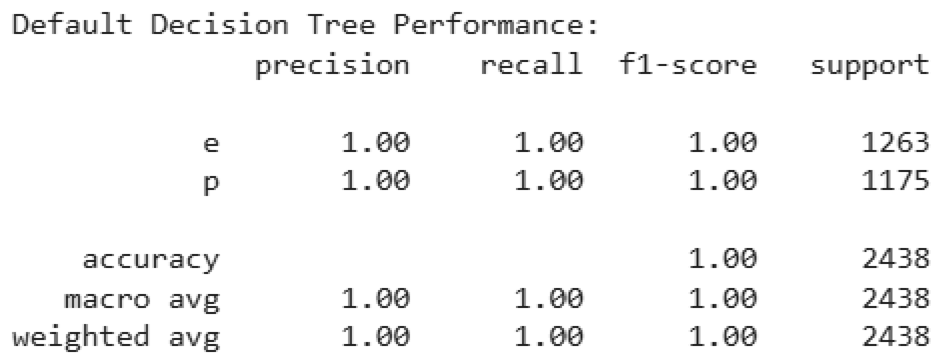
➔ No overfitting, no underfitting but the accuracy is high means that the dataset is too easy to classify (classes are well-separated), it's not complex despite the dropping of some attributes.

The supervised model Decision Tree without parameters :

➔ For the implementation :

- ◆ Used DecisionTreeClassifier(random_state=42) with no pruning parameters.

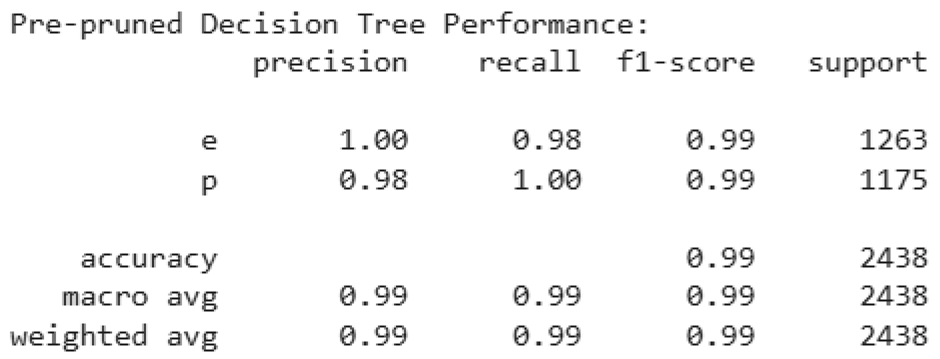
    - ◆ The tree is allowed to grow until all leaves are pure or have fewer than the minimum number of samples required to split.
- ➔ The default tree achieves perfect accuracy, which strongly suggests overfitting.



```
Default Decision Tree Performance:
              precision    recall  f1-score   support

           e       1.00      1.00      1.00      1263
           p       1.00      1.00      1.00      1175

    accuracy                           1.00      2438
   macro avg       1.00      1.00      1.00      2438
weighted avg       1.00      1.00      1.00      2438
```

Pre-Pruning:

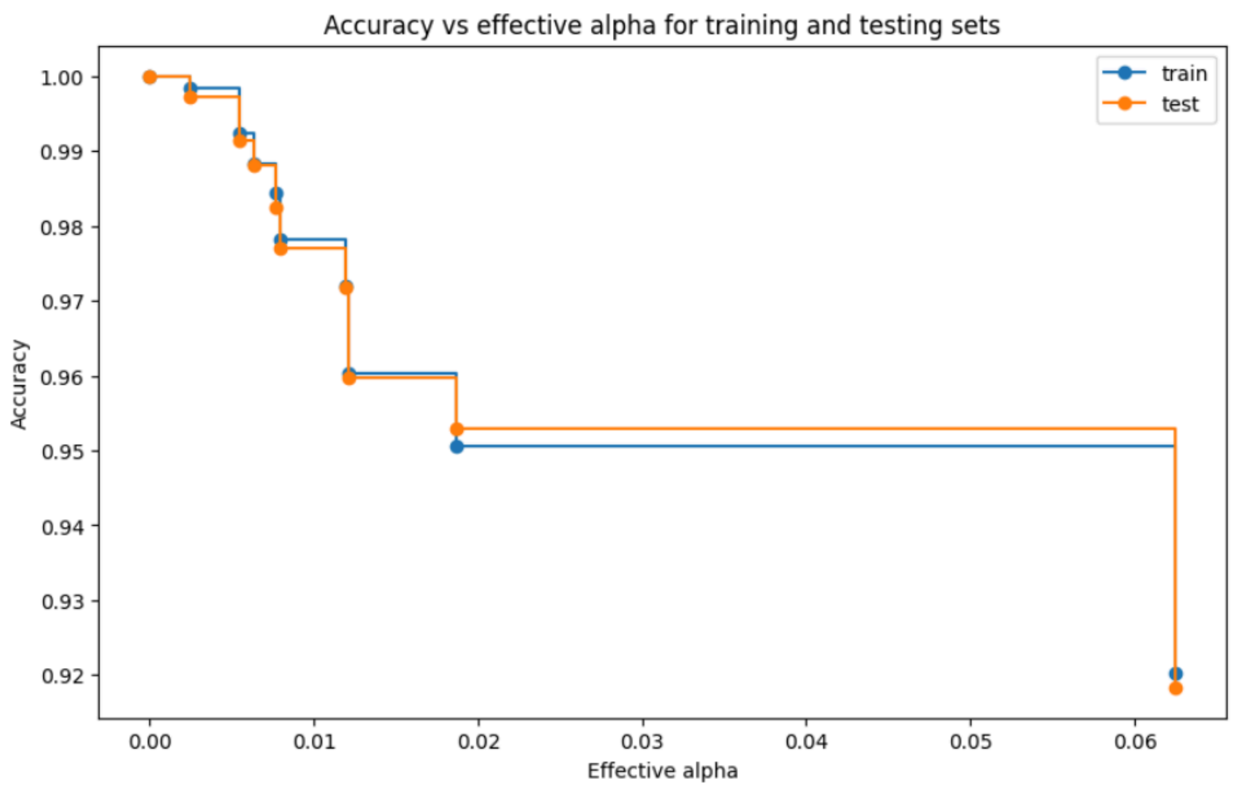
- ➔ The pre-pruned tree introduces regularization, slightly reducing accuracy but improving generalization and interpretability.
- ➔ In real-world applications, pre-pruning is usually preferred unless interpretability is more critical than generalization.



```
Pre-pruned Decision Tree Performance:
              precision    recall  f1-score   support

           e       1.00      0.98      0.99      1263
           p       0.98      1.00      0.99      1175

    accuracy                           0.99      2438
   macro avg       0.99      0.99      0.99      2438
weighted avg       0.99      0.99      0.99      2438
```

Post-Pruned Decision Tree

- ➔ Used cost_complexity_pruning_path() to extract effective ccp_alpha values.
- ➔ Trained multiple trees across a range of ccp_alpha values.
- ➔ Selected the tree with the best test accuracy.



The plot of accuracy vs. effective alpha shows:

- ➔ Highest accuracy at alpha = 0.0, indicating that the fully grown tree performed best on this dataset.
- ➔ As alpha increases, the tree becomes simpler, and accuracy slightly drops.

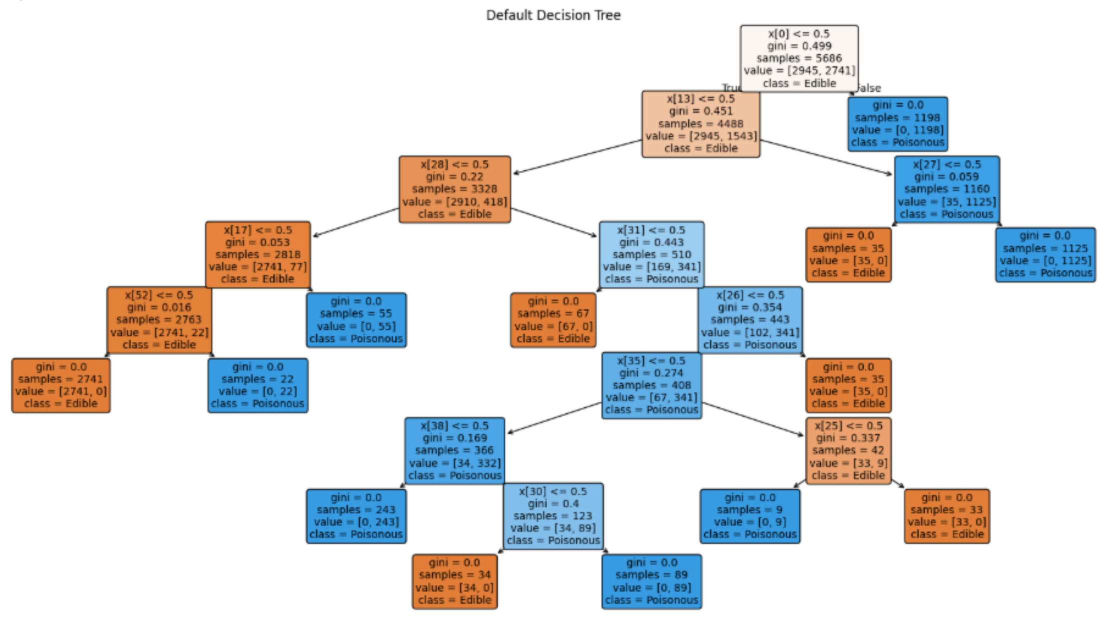
Optimal Alpha:

- ➔ Best CCP Alpha: 0.0
- ➔ Indicates that no post-pruning improved the model — the unpruned tree performed best (though possibly due to the dataset's simplicity or cleanness).

⛔ Post-pruning didn't yield improvements in this case, but the approach is valuable when the default tree overfits noisy or real-world data. The analysis confirms that post-pruning offers transparency into how pruning affects accuracy and model complexity.

Decision Tree Visual Interpretation

```
Best CCP Alpha: 0.0
```


Default Decision Tree
x[0] <= 0.5 gini = 0.499 samples = 5686 value = [2945, 2741] class = Edible
True
False
x[13] <= 0.5 gini = 0.451 samples = 4488 value = [2945, 1543] class = Edible
gini = 0.0 samples = 1198 value = [0, 1198] class = Poisonous
x[28] <= 0.5 gini = 0.22 samples = 3328 value = [2910, 418] class = Edible
x[27] <= 0.5 gini = 0.059 samples = 1160 value = [35, 1125] class = Poisonous
x[17] <= 0.5 gini = 0.053 samples = 2818 value = [2741, 77] class = Edible
x[31] <= 0.5 gini = 0.443 samples = 510 value = [169, 341] class = Poisonous
gini = 0.0 samples = 35 value = [35, 0] class = Edible
gini = 0.0 samples = 1125 value = [0, 1125] class = Poisonous
x[52] <= 0.5 gini = 0.016 samples = 2763 value = [2741, 22] class = Edible
gini = 0.0 samples = 55 value = [0, 55] class = Poisonous
gini = 0.0 samples = 67 value = [67, 0] class = Edible
x[26] <= 0.5 gini = 0.354 samples = 443 value = [102, 341] class = Poisonous
gini = 0.0 samples = 2741 value = [2741, 0] class = Edible
gini = 0.0 samples = 22 value = [0, 22] class = Poisonous
x[35] <= 0.5 gini = 0.274 samples = 408 value = [67, 341] class = Poisonous
gini = 0.0 samples = 35 value = [35, 0] class = Edible
x[38] <= 0.5 gini = 0.169 samples = 366 value = [34, 332] class = Poisonous
x[25] <= 0.5 gini = 0.337 samples = 42 value = [33, 9] class = Edible
gini = 0.0 samples = 243 value = [0, 243] class = Poisonous
x[30] <= 0.5 gini = 0.4 samples = 123 value = [34, 89] class = Poisonous
gini = 0.0 samples = 9 value = [0, 9] class = Poisonous
gini = 0.0 samples = 33 value = [33, 0] class = Edible
gini = 0.0 samples = 34 value = [34, 0] class = Edible
gini = 0.0 samples = 89 value = [0, 89] class = Poisonous

➔ Early splits are highly informative:

x[0] and x[13] already make major separations.

Most of the purity is achieved within 2–3 levels.

➔ Gini values close to 0 = confident prediction:

Gini = 0 means perfect classification at that node.

Many leaves are pure (either fully edible or poisonous).

➔ Unbalanced splits (not 50/50):

Some splits lead to very few samples (e.g., samples = 9), which might suggest overfitting unless pruned effectively.

➔ Feature Indexes (x[i]):

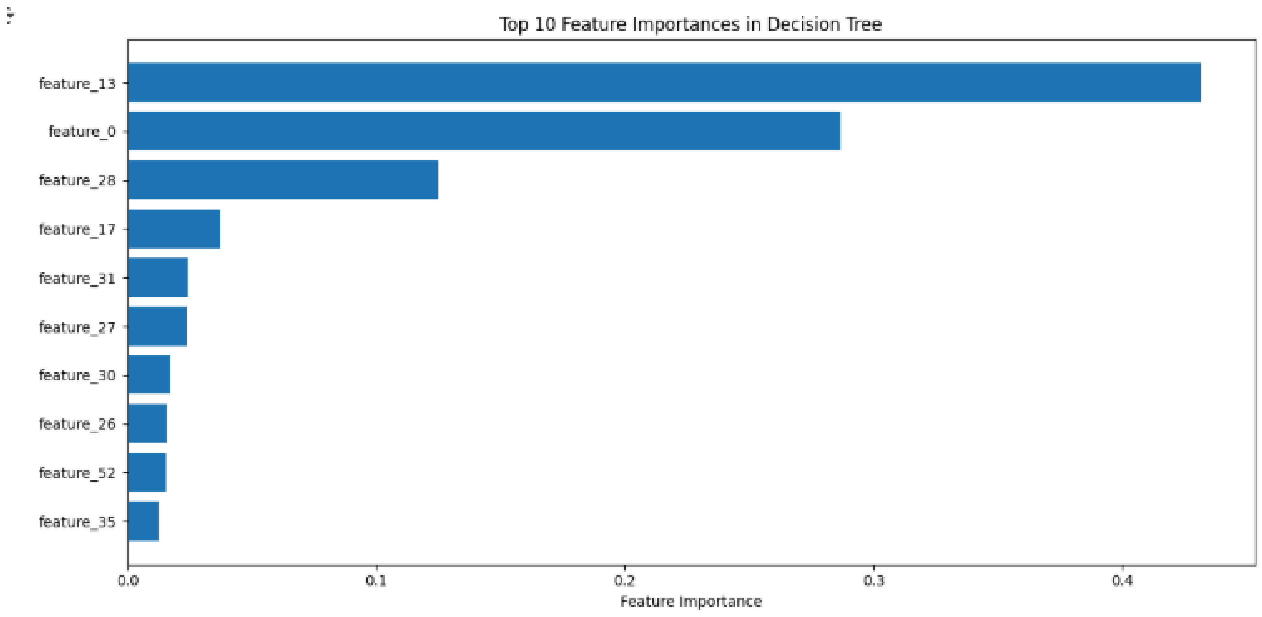
Each x[i] corresponds to a feature (e.g., cap shape, gill color) in your dataset. Overall Performance :

- ◆ Highly accurate: Many nodes are pure.
- ◆ Balanced: Handles both edible and poisonous fairly.
- ◆ Efficient: Splits with low Gini show the model learned strong rules from the data.

Feature importance:

Shallow vs. Deep Splits:

Features like feature_13 and feature_0 dominate because they're used near the root, affecting more samples.



Grid search:

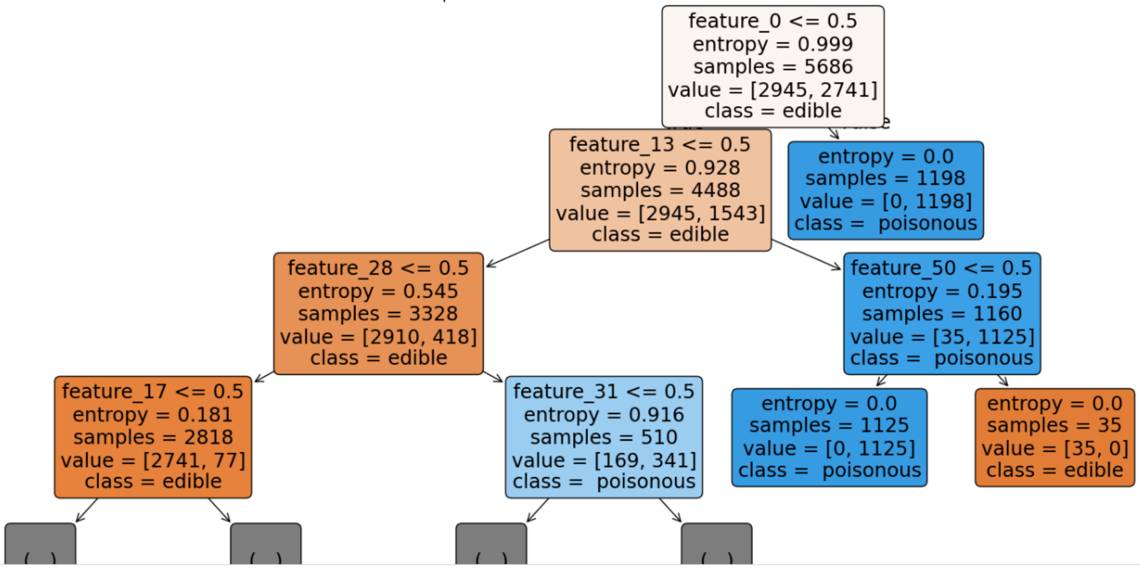
From our result we can tell that:

```
Best Parameters:
{'ccp_alpha': 0.001, 'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2}

Best Cross-validation Score: 1.0000

Best Model Performance:
              precision    recall  f1-score   support

           e       1.00      1.00      1.00      1263
           p       1.00      1.00      1.00      1175

    accuracy                           1.00      2438
   macro avg       1.00      1.00      1.00      2438
weighted avg       1.00      1.00      1.00      2438
```

✅ Dataset is highly separable — possibly due to strong class-distinguishing features (e.g., odor or spore print color).

- ➔ Our early feature importance chart supports that a few features (like feature_13 and feature_0) might already explain most of the label.

Top Levels of the Best Decision Tree


feature_0 <= 0.5
entropy = 0.999
samples = 5686
value = [2945, 2741]
class = edible
feature_13 <= 0.5
entropy = 0.928
samples = 4488
value = [2945, 1543]
class = edible
entropy = 0.0
samples = 1198
value = [0, 1198]
class = poisonous
feature_28 <= 0.5
entropy = 0.545
samples = 3328
value = [2910, 418]
class = edible
feature_50 <= 0.5
entropy = 0.195
samples = 1160
value = [35, 1125]
class = poisonous
feature_17 <= 0.5
entropy = 0.181
samples = 2818
value = [2741, 77]
class = edible
feature_31 <= 0.5
entropy = 0.916
samples = 510
value = [169, 341]
class = poisonous
entropy = 0.0
samples = 1125
value = [0, 1125]
class = poisonous
entropy = 0.0
samples = 35
value = [35, 0]
class = edible
(...)
(...)
(...)
(...)