MACHINE LEARNING PROJECT

TREE PREDICTIONS for BINARY CLASSIFICATION

Mushroom Dataset Poisonuos Prediction Analysis with Tree Predictors for Binary Classification

SELEN GEZGINCI – 32204A

Data Science for Economics

Table of Contents

https://github.com/selengez/MLProject

I declare that this material, which I now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

ABSTRACT

The intention of this project is to distinguish mushrooms that are toxic or edible through the use of a Mushroom dataset which is made up of 21 characteristics that describe different types of physical appearances in mushrooms alongside one target column denoting whether they are poisonous or edible. The data being categorical, tree predictors (decision trees) were used for the binary classification process. Initial data preprocessing involved dropping columns with over 40% missing values and replacing them by mean for numerical features, while mode was employed for categorical ones. For feature encoding purpose, categorical variables were subjected to one-hot encoding while label encoding was applied on the target variable. The dataset was split into an 80-20 training-testing proportion. Decision trees based on entropy and Gini index were trained with their performance evaluated using accuracy, precision, recall, F1-score and confusion matrices. Hyperparameter tuning was done using RandomizedSearchCV to optimize the decision tree classifier. Model performance was also assessed by drawing learning curves as well as checking if there is any sort of overfitting going on. Not only did these results show that entropy-based decision tree performed better than Gini-based but also outdid it in terms of accuracy while maintaining high balance between precision and recall.

1. INTRODUCTION

The aim of the project is to predict whether a mushroom in the Mushroom dataset is poisonous (p) or edible (e). It was used for binary classification for this purpose. The dataset contains 21 features describing various physical properties of mushrooms and a target column (class) indicating edibility. The dataset is rich in categorical data with only a few numerical features; this makes it an interesting challenge for machine learning models that process mixed data types.

Feature	Description	Value type
class	indicates whether the mushroom is poisonous (p) or edible (e).	object
cap-diameter	It represents the diameter of the mushroom cap in centimeters	float
cap-shape	It describes the shape of the mushroom cap (e.g., bell, conical, convex).	object
cap-surface	It describes the surface texture of the cap (e.g., fibrous, grooves, scaly).	object
cap-color	It represents the color of the mushroom cap (e.g., brown, yellow, white).	object
does-bruise-or-bleed	It indicates whether the mushroom bruises or bleeds when damaged (bruises, no).	object
gill-attachment	how the gills are attached to the mushroom cap (e.g., attached, free).	object
gill-spacIng	It represents the spacing of the gills (e.g., close, crowded, distant).	object
gill-color	It represents the color of the gills (e.g., black, brown, white).	object
stem-height	It represents the height of the mushroom stem in centimeters.	float
stem-width	It represents the width of the mushroom stem in centimeters.	float
stem-root	It describes the root of the mushroom stem (e.g., bulbous, club, cup).	object
stem-surface	It represents the surface texture of the stem (e.g., fibrous, scaly, smooth).	object
stem-color	It represents the color of the stem (e.g., brown, yellow, white).	object
veil-type	type of veil covering the mushroom (e.g., partial, universal).	object
veil-color	It represents the color of the veil (e.g., brown, orange, white).	object
has-ring	It indicates whether the mushroom has a ring on the stem (yes, no).	object
ring-type	type of ring present on the stem (e.g., evanescent, flaring, large).	object
spore-print-color	It represents the color of the spore print (e.g., black, brown, white).	object
habitat	It describes the habitat where the mushroom is commonly found	object
season	It represents the season during which the mushroom is typically found	object

Table 1 Description of Features

2. DATA PRE-PROCESSING

2.1.Handling Missing Values

To maintain integrity in the dataset and make accurate model predictions, it becomes essential to handle missing values accordingly. These steps consist of:

2.1.1. Identifying Missing Values: Missing values in each column were checked. When this dataset was first encountered, 307463 values were empty in various columns. Null values rate in each column were calculated.

Feature	# of missing values	Rate
class	0	0%
cap-diameter	0	0%
cap-shape	0	0%
cap-surface	14120	23%
cap-color	0	0%
does-bruise-or-bleed	0	0%
gill-attachment	9884	16%
gill-spacing	25063	41%
gill-color	0	0%
stem-height	0	0%
stem-width	0	0%
stem-root	51538	84%
stem-surface	38124	62%
stem-color	0	0%
veil-type	57892	94%
veil-color	53656	87%
has-ring	0	0%
ring-type	2471	4%
spore-print-color	54715	90%
habitat	0	0%
season	0	0%

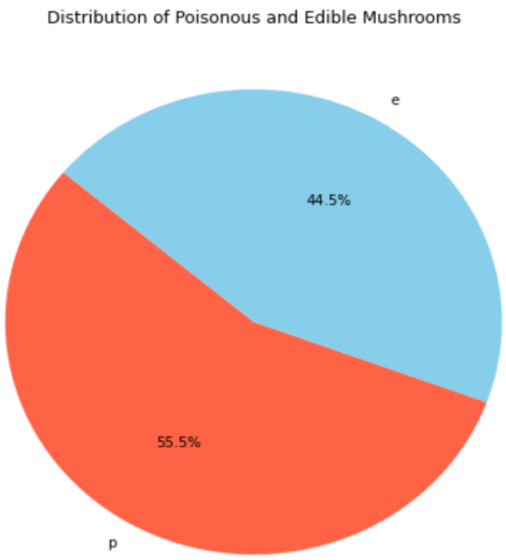
Table 2 Number of Missing Values

- Drop the high missing values rate columns: Columns having high percentage missing values (more than 40%) were dropped from the dataset.
- Drop the missing rows: Missing rows in the "gill-attachment" column have been removed.
- Implementing Missing Values: Appropriate imputation techniques were employed to fill in columns with missing values that had not been removed:
- Numeric Columns: For the numerical columns, the mean of the column was used to fill in for any missing values available.
- Categorical Columns: The mode (most frequent value) of a column was used to populate null values.

3. EXPLANATORY DATA ANALYSIS

3.1.Distribution of the Target Variable:

A pie chart was created to visualize how the target variable distribution looks like. It shows whether a mushroom is poisonous or edible and this can be seen by looking at the variables' descriptions as mentioned earlier in this report.

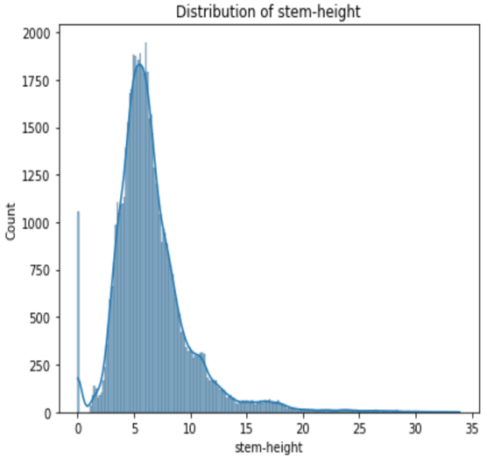


The pie chart indicates that these two classes are well-balanced within the dataset where there are almost equal amounts of edible and poisonous mushrooms. This equality is useful when training machine learning models since it prevents overfitting towards certain categories thereby enabling them to learn both groups effectively.

Figure 1

3.2. Distribution of Numerical Features:

Numerical features distribution such as cap diameter, stem height, stem width was shown by histograms.



- The histogram shows that small stem heights are very frequent while large ones are extremely rare.
- It has a right skewness with a long tail to the right which suggests some variation in stem heights but strong preference for smaller ones.
- There is an obvious nonuniform distribution with significant clustering around lower stem heights.

Figure 2

- This histogram exhibits a right-skewed distribution indicating that majority mushrooms have smaller cap diameter while fewer mushrooms possess larger diameter.
- High frequency of small head diameters rapidly drops off as size increases; fewest occur at largest sizes.
- Therefore, it seems likely that cap may not be distributed evenly over all possible

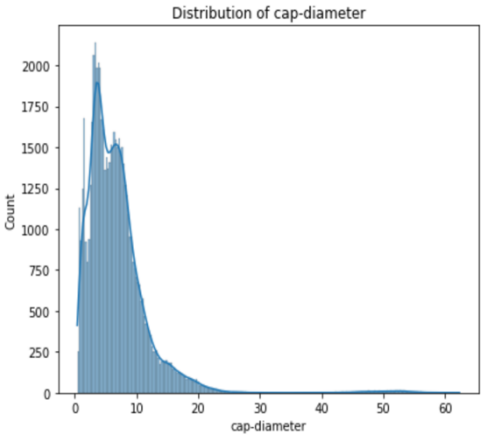


Figure 3

values but instead restricted within certain ranges which tend to occur more frequently among mushrooms.

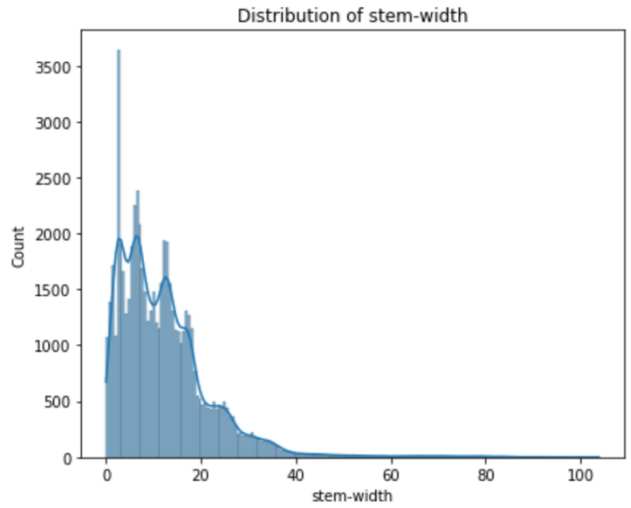


Figure 4

- The histogram shows that smaller stem widths are very common, while larger stem widths are quite rare. The distribution is skewed to the right with a long tail on the right; this implies there exists some variability in stem widths but strong preference for lesser widths.
- There is an evident nonuniform distribution with substantial clustering around narrower stem width.

3.3. Distribution of Categorical Features:

Bar charts visualize the categorical variables distributions such as cap surface, ring type and gill attachment.

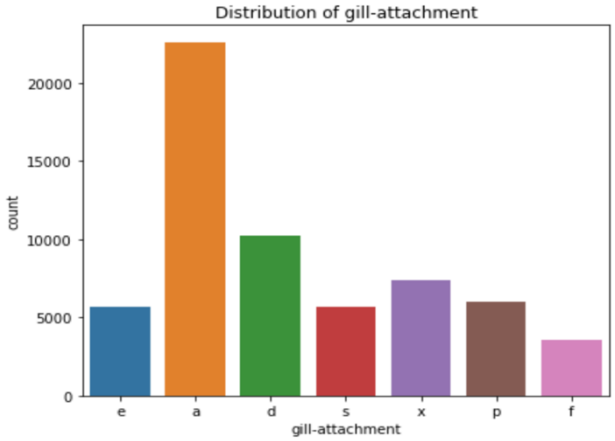


Figure 5

- The type 'adnate' comes first in terms of the number of values as the most common gill attachment type.
- The second and third types are ‘decurrent’ and ‘adnexed’ respectively; they have a considerable number but smaller than ‘adnate’.
- ‘Free’, ‘sinuate’, and ‘pores’ are less common with the highest number being that of ‘e’.

- The most common type of cap surface is ‘sticky’, with over 20,000 occurrences.
- The next most common types are ‘smooth’ and ‘scaly’, with substantial counts but significantly less than ‘sticky’.
- The types ‘grooves’ and ‘shiny’ have moderate counts.

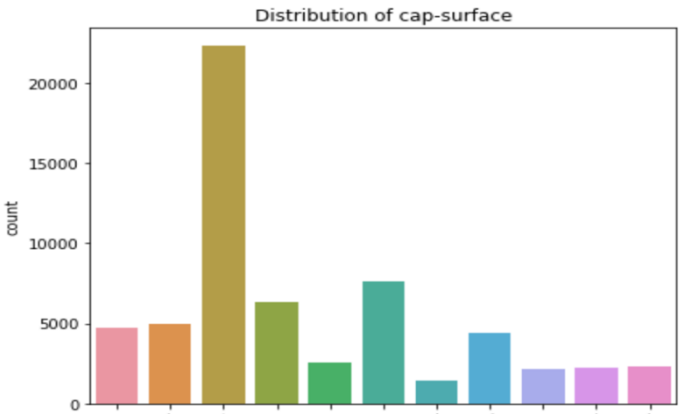
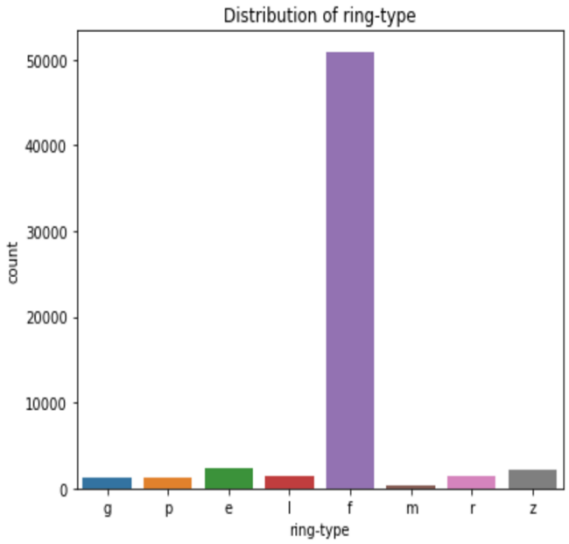


Figure 6

- The types ‘fleshy’, ‘leathery’, ‘d’, ‘wrinkled’, ‘fibrous’, and ‘silky’ are less common.



•The most common type of ring is ‘none, with over 50,000 values, making it the predominant ring type by a large margin.

•The other types (‘grooved’, ‘pendant, ‘evanescent, ‘large’, ‘movable’, ‘flaring’, ‘zone’) have relatively low counts, indicating they are much less common.

Figure 7

4. Encoding Process

In our dataset, the classes representing whether mushrooms are poisonous (‘p’) or edible (‘e’) are imbalanced, with one class having significantly more instances than the other. To ensure our machine learning model does not become biased towards the majority class, the dataset needs to balanced.

Figure 8

<u>One-hot Encoding:</u>

To address these types of features, one-hot encoding was used. Each categorical value was transformed into a new categorical column through one-hot encoding and assign 1 or 0 binary values to the columns. pd.get_dummies function was used from Pandas library for this purpose which creates new columns automatically for each unique category and assigns appropriate binary values.

<u>Target Variable Encoding:</u>

The target variable in this dataset, which indicates whether a mushroom is poisonous (‘p’) or edible (‘e’), is also categorical. To convert this target variable into a numerical format, LabelEncoder were used the from the sklearn.preprocessing module. Label encoding assigns a unique integer to each class label, converting ‘p’ to 1 and ‘e’ to 0.

Scaling Numerical Features:

Additionally, specific numerical features were scaled, namely cap-diameter, stem-height, and stem-width, using the MinMaxScaler. Scaling was done to normalize the range of these features between 0 and 1, which helps in improving the performance and convergence speed of machine learning algorithms.

Splitting the Datasets

While the training set is used to train the model, the testing set is reserved for evaluating its performance. The dataset was split using an 80-20 split; here 80% of the data was used for training and 20% for testing.

5. Method Application

5.1. Tree Predictors for Binary Classification

Tree predictors, commonly known as decision trees, are a powerful and interpretable method for binary classification tasks. They are particularly useful because they mimic human decision-making processes by splitting data into subsets based on feature values, ultimately leading to a decision or classification. This section details the implementation and functionality of tree estimators, focusing specifically on the use of single feature pairwise tests as decision criteria at internal nodes. In the context of tree predictors, each internal node in the tree represents a decision point that splits the data based on a specific feature. The decision criteria can be broadly categorized into two types based on the nature of the feature: numerical/ordinal features and categorical features.

In this study, tree predictors were used to predict whether a mushroom is poisonous or not using the mushroom data set. The details of this process:

Numerical/Ordinal Features:

- For numerical or ordinal features, the decision criteria involve selecting a threshold value.
- The decision test at an internal node check whether the value of the selected feature for a data point is less than or equal to the threshold.
- Data points satisfying the condition (i.e., having feature values less than or equal to the threshold) are directed to the left child node, while the remaining data points are directed to the right child node.

Categorical Features:

- For categorical features, the decision criteria involve membership tests, which determine whether the value of the feature belongs to a specified subset of categories.
- The decision test at an internal node check whether the value of the selected feature for a data point matches one of the specified categories.

- Data points satisfying the condition (i.e., having feature values in the specified subset) are directed to the left child node, while the others are directed to the right child node.

For this research, a tree of decisions was made out to predict if a mushroom is poisonous or not from the mushroom dataset. Moreover, this tree's foundation TreeNode class has been elaborated on. Decision trees are made up of nodes denoted by the TreeNode class where every single node possesses some details concerning itself as well as whether it is a leaf.

All parts required to build a decision tree, train it and make predictions are contained in the DecisionTree class. The criteria used for initializing hyperparameters include maximum depth; minimum number of samples in leaf nodes; and minimum information gain among others. Entropy metrics such as Gini index and variance reduction are utilized for calculating impurity of data. Best split criteria are determined by dividing data with respect to particular features and threshold values while feature importance is updated through computing information gains between nodes. The training dataset trains the model which optimizes structure of decision tree whose each data point during prediction processes traces back to leaf node then calculates prediction probabilities before finally selecting most likely class. Parameters of models together with their corresponding trees' structures are presented visually in such manner that users can comprehend the way these models operate. These procedures show how effective decision trees can be when it comes solving classification problems within intricate datasets like biological ones.

The determination regarding how the information is split at each internal node is made by specific splitting criteria that have been designed to maximize classification accuracy. These are a few commonly used splitting criteria.

6. Test Results

The performance of models that use a decision tree were estimated on the basis of two standards: entropy and Gini index. They were run through the mushroom data set in order to predict if a mushroom is poisonous or not. The precision, accuracy, recall F1-score and confusion matrices were used to assess the models.

```
Entropy Criterion:
Accuracy: 85.08%
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.98      0.86      4731
           1       0.98      0.74      0.84      5506

    accuracy                           0.85     10237
   macro avg       0.87      0.86      0.85     10237
weighted avg       0.88      0.85      0.85     10237

Confusion Matrix:
[[4660   71]
 [1456 4050]]
```

Figure 9

Accordingly, the entropy-based decision tree attained an accuracy level of 85.08% which implies that it categorized most cases correctly. For non-poisonous mushrooms (class 0), precision was at 0.76; meaning that 76% of all predicted non-poisonous mushrooms were actually non-poisonous while recall stood out at 0.98 – this shows how well the model identified actual negative samples. Therefore its F1-score amounted to 0.86 thereby indicating fair balance between these two measures when used together as one entity known as F-measure or FBetaScore (where Beta weighs importance given towards precision against recall). With respect to poisonous mushrooms (class 1), precision reached up to 0 .98 while recall fell short at 74%, hence resulting into F1 score being equal with 84%. In terms of numbers from confusion matrix there are true positives equaling four thousand six hundred sixty for class zero ; false positives seventy one; true negatives four thousand fifty one for class one with false negatives standing at fourteen hundred fifty six.

```
Gini Criterion:
Accuracy: 77.47%
Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.95      0.80      4731
           1       0.94      0.62      0.75      5506

    accuracy                           0.77     10237
   macro avg       0.81      0.79      0.77     10237
weighted avg       0.82      0.77      0.77     10237

Confusion Matrix:
[[4494  237]
 [2069 3437]]
```

Figure 10

On the other hand, Gini-based decision trees had lower than expected results; achieving an accuracy rate as low as77%. It can be observed from this statement alone that such models tend not perform very well but they still have some advantages over other types because their simplicity makes them easier implement and understand by different users even those who do not possess deep background knowledge in statistics or machine learning fields.Precision value for identifying non- poisonous mushroom using gini index was recorded at68%; meaning only a few examples were classified correctly while majority got misclassified. However when it comes recall then things change dramatically as its total value equals to 95%, thus indicating that most non-poisonous mushrooms were detected by the model thereby leading us into f1-score calculation which is at0.80 only . For poisonous mushrooms precision went up significantly with a value of 0 .94 compared to recall which dropped down drastically at 62%; hence resulting into F1 score being equal with 75%. Looking at numbers

from confusion matrix there are true positives equaling four thousand four hundred ninety four for class zero ; false positives two hundred thirty seven; true negatives three thousand four hundred thirty seven for class one with false negatives standing at twenty six hundred nine.

In comparing both these models It can be clearly seen that decision tree based on entropy does better than those created using gini index when looking at overall accuracy and balance between precision & recall in all classes. The better classification may be due to fact that more poisonous samples were identified as such (higher recall) which is important for safety reasons; also reflected through higher f1 scores for both poisonous and non-poisonous classes showing improved precision-recall balance.

7. Learning Curves

This study used hyperparameter tuning using the RandomizedSearchCV technique to optimize a decision tree classifier to predict whether mushrooms are poisonous or edible based on the mushroom dataset.

Additionally, learning curve analysis was performed to evaluate the performance stability of the model and detect potential overfitting or underfitting issues. Learning curves were plotted using varying sizes of the training data set; The results showed that the performance of the model stabilized as the number of training examples increased, indicating a well-generalized model. Cross-validation mean scores and standard deviations were also reported, providing information about the variability and robustness of the model across different data subsets.

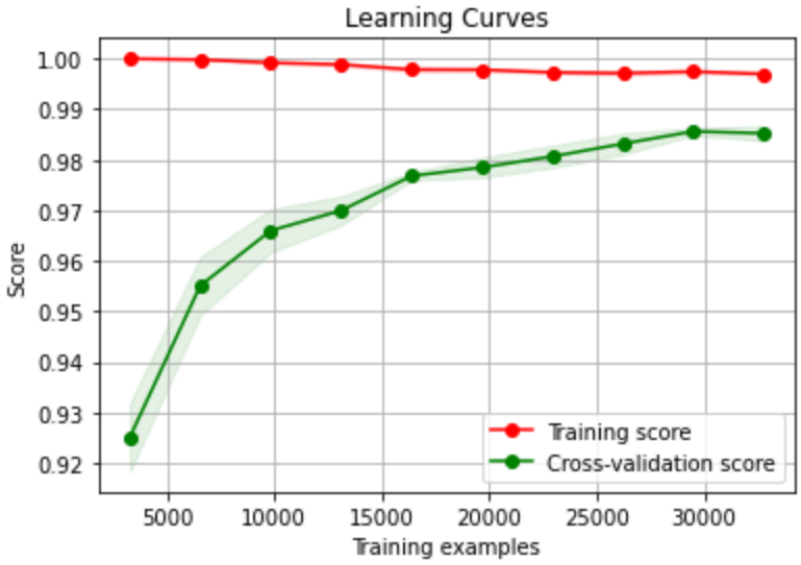


Figure 11

```
Best Parameters: {'min_samples_leaf': 1, 'min_information_gain': 0, 'max_depth': 15, 'criterion': 'gini'}
Best Cross-validation Accuracy: 98.43%
Training Accuracy: 99.46%
Test Accuracy: 98.83%
```

- The performance of a decision tree classifier is shown in the learning curve plot as against the number of training examples. There are two curves on this plot: one for training score (red) and another one for cross-validation score (green).
- The training score is very high and almost perfect (1.0).
- The cross-validation score is high but not as high as the training score, indicating a performance gap.
- This means overfitting: the model is too complex and captures noise in the training data, performing less effectively on unseen data.

Handling Overfitting

- Pruning the Decision Tree:
- Reduced maximum depth: Reduced the maximum depth of the tree to prevent it from becoming too complex.
- Increasing min_samples_leaf: Added more samples per leaf to make the tree less sensitive to noise.
- Increased min_information_gain: Adjusted a higher threshold for minimum information gain to decide splits.

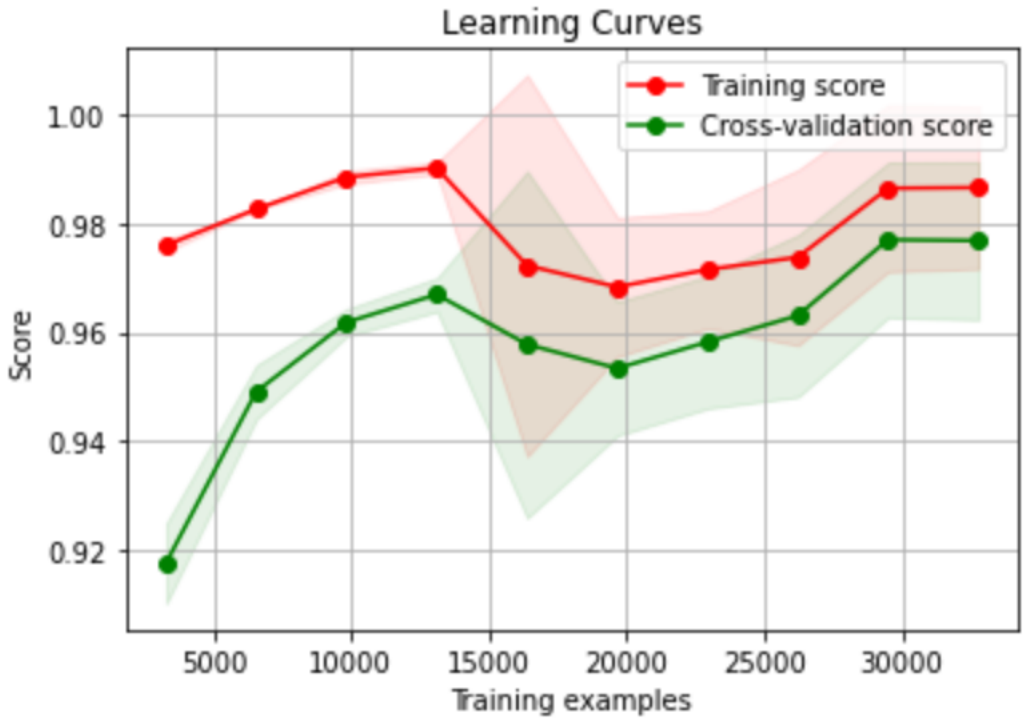


Figure 12

```
Best Parameters: {'min_samples_leaf': 1, 'min_information_gain': 0, 'max_depth': 15, 'criterion': 'entropy'}
Best Cross-validation Accuracy: 98.52%
Training Accuracy: 99.68%
Test Accuracy: 98.83%
```

- The training score is high, but not perfect, indicating that the model is fitting the training data well without overfitting completely.
- The cross- validation score is closer to the training score, indicating that the model generalizes better than previous graph.
- This suggests reduced overfitting but there might still be a small amount of overfitting as indicated by the gap between the training and cross-validation scores.

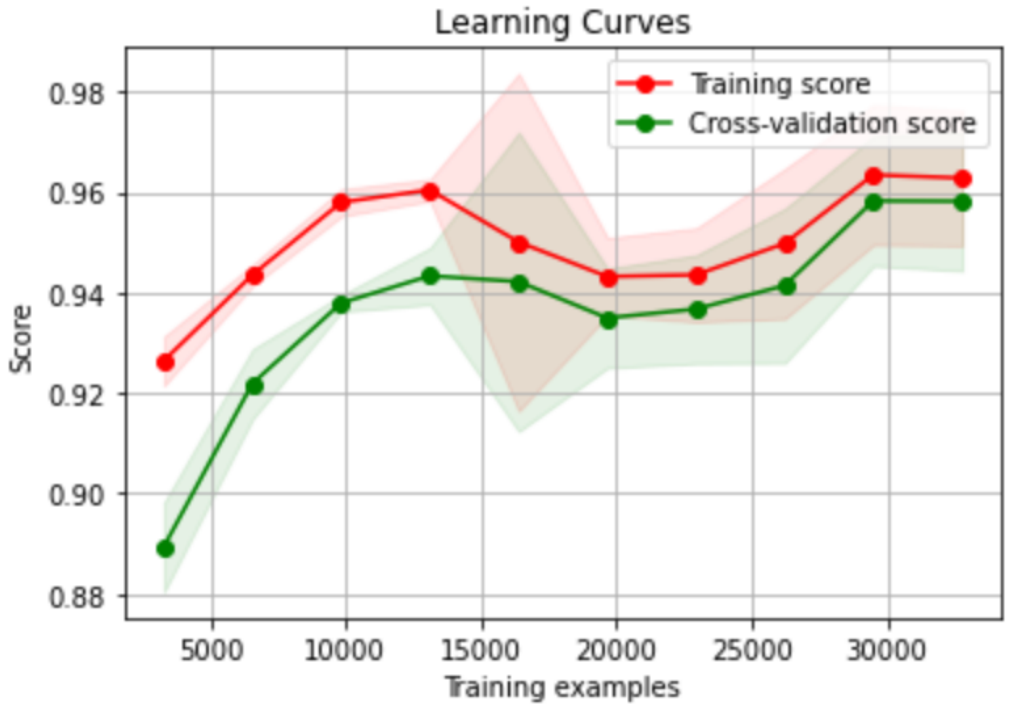


Figure 13

```
Best Parameters: {'min_samples_leaf': 2, 'min_information_gain': 0.005, 'max_depth': 15, 'criterion': 'entropy'}
Best Cross-validation Accuracy: 98.17%
Training Accuracy: 99.64%
Test Accuracy: 98.98%
```

- The training score is high, but not perfect, indicating that the model fits the training data well.
- The cross-validation score is close to the training score, indicating that the model generalizes well.
- The overfitting issue has been significantly reducing and the model is performing well with good generalization.

8. Feature Selection

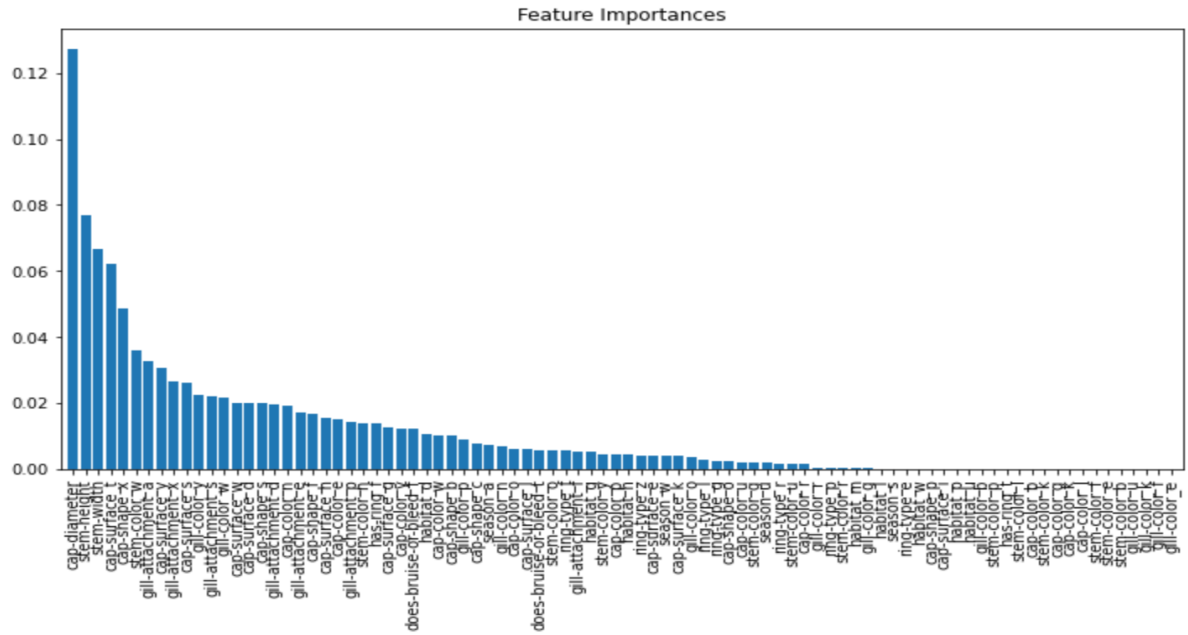


Figure 14

Cap Diameter: This feature has the highest importance, indicating it plays a significant role in predicting the target variable.

Stem Width, Cap Color, Stem Color: These features also have high importance, contributing substantially to the model's predictions.

9. Conclusion

This project showed the importance of data preprocessing techniques, model tuning for improving performance in classification tasks. When we used hyperparameter tuning, we found that an entropy-based model with specific parameters (min_samples_leaf=1, min_information_gain=0, max_depth=15, criterion='entropy') showed a cross-validation accuracy of 98.52%, a training accuracy of 99.68%, and a test accuracy of 98.83%. By comparison, a Gini-based model with similar parameters achieved a cross-validation accuracy of 98.43%, a training accuracy of 99.46%, and a test accuracy of 98.83%. Another entropy-based model with slightly different parameters (min_samples_leaf=2, min_information_gain=0.005, max_depth=15, criterion='entropy') showed a cross-validation accuracy of 98.17%, a training accuracy of 99.64%, and a test accuracy of 98.98%. When analysing the learning curves, the first graph showed signs of overfitting. The training score was almost perfect at 1.00, suggesting the model fit the training data too closely. However, the model with parameters (min_samples_leaf=2, min_information_gain=0.005, max_depth=15, criterion='entropy') showed more stable performance. Both the training and cross-validation scores out 0.96 with a minimal gap, indicating a well-generalized model with less risk of overfitting.