University of Pisa


# Data Mining and Machine Learning Project

Matteo Comini
Giacomo Moro

A.Y. 2023-2024

# Contents

# 1 Introduction

Mushrooms are a diverse and fascinating group of organisms that play crucial roles in various ecosystems. While many mushrooms are edible and highly nutritious, others are toxic and can pose serious health risks if ingested. Foraging for mushrooms, therefore, requires a keen eye and extensive knowledge to accurately distinguish between safe and poisonous species. In this context, the development of a reliable and automated mushroom classifier becomes highly valuable, particularly for enthusiasts, researchers, and those involved in food safety.

The primary objective of this project is to create a robust classifier capable of accurately identifying different species of mushrooms from images. This classifier aims to discern between poisonous and edible mushrooms, providing users with information about the detected mushroom species.

To achieve this goal, the project follows a systematic approach beginning with the collection and preprocessing of a diverse dataset of mushroom images encompassing seven distinct species. After cleaning, we employ a state-of-the-art deep learning model to extract features from the images. These features, drawn from the penultimate layer of the convolutional neural network, serve as the foundation for training various classifiers.

Given the high dimensionality of the extracted features, dimensionality reduction techniques are utilized to try enhancing computational efficiency and improving model performance. The reduced set of features is then used to train multiple classifiers.. The performance of these classifiers is evaluated to determine the most effective model for mushroom classification.

The following sections detail the methodology, results, and insights gained from this comprehensive endeavor to create an accurate and reliable mushroom classifier.

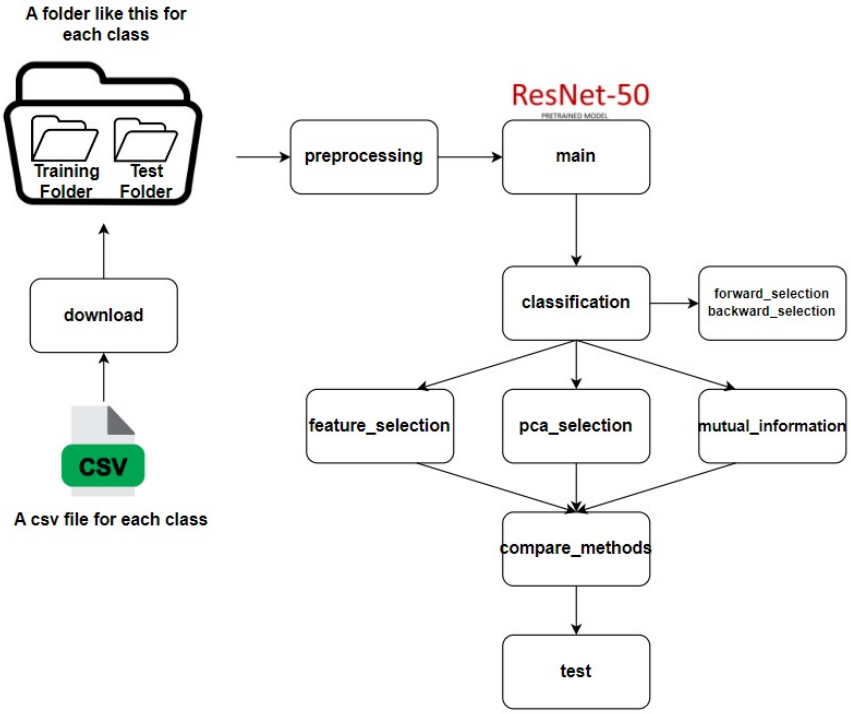



Figure 1: Project structure

# 2 Dataset

The dataset used for this project was sourced from iNaturalist, a platform known for its extensive collection of biodiversity data. The dataset was initially provided in a CSV file containing links to the actual images, which were subsequently downloaded using a custom script.

This comprehensive dataset comprises over 87,000 RGB images representing seven different species of mushrooms. The images vary significantly in terms of size, and the mushrooms are depicted in a diverse range of appearances. These variations include different angles, lighting conditions, and backgrounds, contributing to the dataset's richness and complexity.

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | time_zone | user_name | image_url | latitude | longitude | scientific_name | common_name | iconic_taxon_name |
| 2 | 111472 | Eastern Time (US & Canada) | | https://static.inaturalist.org/photos/158969/medium.jpg | 41.1138333333 | -81523 | Agaricus campestris | | Fungi |
| 3 | 112925 | Eastern Time (US & Canada) | Lindsey | https://static.inaturalist.org/photos/161348/medium.jpg | 41.2358333333 | -81.5546666667 | Agaricus campestris | | Fungi |
| 4 | 121624 | Central Time (US & Canada) | Tony Gerard | https://static.inaturalist.org/photos/173520/medium.JPG | 37.3118429743 | -89.0179806444 | Agaricus campestris | | Fungi |
| 5 | 136251 | Tijuana | | https://static.inaturalist.org/photos/189538/medium.jpg | 47.2153144 | -123.1119275 | Agaricus campestris | | Fungi |
| 6 | 158486 | Hong Kong | Sterling Sheehy | https://inaturalist-open-data.s3.amazonaws.com/photos/215759/medium.JPG | 37.7996595078 | -122.4518564343 | Agaricus campestris | | Fungi |
| 7 | 218959 | Pacific Time (US & Canada) | Jem | https://static.inaturalist.org/photos/275761/medium.PNG | 32.5875713925 | -117.0613485575 | Agaricus campestris | | Fungi |
| 8 | 720971 | Mexico City | | https://inaturalist-open-data.s3.amazonaws.com/photos/906244/medium.jpg | 20.010129 | -99.222507 | Agaricus campestris | | Fungi |
| 9 | 803918 | Mexico City | Sandino Guerrero | https://static.inaturalist.org/photos/1012898/medium.JPG | 20.684324 | -99.803503 | Agaricus campestris | | Fungi |
| 10 | 876795 | Mountain Time (US & Canada) | | https://inaturalist-open-data.s3.amazonaws.com/photos/1112873/medium.jpg | 40.0337060554 | -111.5626294521 | Agaricus campestris | | Fungi |
| 11 | 885180 | Alaska | Matt Bowser | https://inaturalist-open-data.s3.amazonaws.com/photos/1124293/medium.JPG | 60.46414 | -151.07364 | Agaricus campestris | | Fungi |
| 12 | 1024555 | Pacific Time (US & Canada) | Tiffany Rachel Elam | https://static.inaturalist.org/photos/1284457/medium.jpg | 47.170592 | -122.564238 | Agaricus campestris | | Fungi |
| 13 | 1062767 | Pacific Time (US & Canada) | Leslie Flint | https://inaturalist-open-data.s3.amazonaws.com/photos/1332082/medium.jpg | 37.4723866667 | -122.4444966667 | Agaricus campestris | | Fungi |
| 14 | 1088584 | Pacific Time (US & Canada) | Dominic | https://inaturalist-open-data.s3.amazonaws.com/photos/1367387/medium.jpg | 37.957702 | -121.29078 | Agaricus campestris | | Fungi |
| 15 | 1111988 | Pacific Time (US & Canada) | | https://inaturalist-open-data.s3.amazonaws.com/photos/1397728/medium.JPG | 38.057049 | -121.364115 | Agaricus campestris | | Fungi |

Figure 2: Original Dataset Snippet

Figure 3: Images of Agaricus Campestris from our dataset

## 2.1 Preprocessing

The first step in the preprocessing pipeline involved scanning the dataset for corrupted images that could not be opened or processed correctly. These images were removed from the dataset to prevent any negative impact on the training and evaluation of the classifier.

Convolutional neural networks (CNNs) typically require input images of a fixed size. For this project, we adopted a resizing method to transform all images to a standard size of 224x224 pixels, which is the input size required by ResNet50, the convolutional nural network of our choice. The resizing process was designed to minimize the loss of important features from the mushroom images. For each image, the length and width were measured. The smaller of the two dimensions was selected to create a square. This square was centered on the original image, ensuring that the mushroom remains as central as possible. The central square was then scaled down to the required size of 224x224 pixels. This method helps to maintain the integrity of the mushroom's features by focusing on the central part of the image, where the mushroom is most likely located,

and minimizing the probability of cutting away any part of the mushroom during the resizing process.

Figure 4: Amanita Muscaria original image

Figure 5: Amanita Muscaria preprocessed image

## 2.2 Feature extraction

After preprocessing, the next crucial step is feature extraction. For this purpose, we utilized the ResNet50 model, a powerful convolutional neural network pre-trained on ImageNet, known for its excellent feature extraction capabilities. The ResNet50 model was employed to extract features from the mushroom images. Specifically, features were extracted from the penultimate layer of the network. This layer captures high-level representations of the image, encapsulating complex patterns and attributes essential for distinguishing between different mushroom species:

- Each preprocessed image was loaded and passed through the ResNet50 model.
- The activations from the penultimate layer were extracted for each image, resulting in a feature vector of 2048 dimensions.
- These feature vectors were stored for subsequent use in training various classifiers.

The extracted features form the basis for training and evaluating the classifiers, as detailed in the subsequent sections. This structured approach ensures that the classifier is built on a solid foundation of high-quality, representative data, facilitating accurate and reliable classification.

# 3 Feature reduction

In this section, we delve into the methods employed for reducing the number of features in our mushroom classification project. Given the initial 2048 features extracted from the ResNet50 model, it was essential to reduce the dimensionality to enhance computational efficiency and potentially improve model performance. This was achieved through various techniques.

## 3.1 PCA

Principal Component Analysis is a widely used technique for dimensionality reduction that transforms the original features into a new set of uncorrelated features, known as principal components, while preserving as much of the original variance as possible.

For our project, we aimed to retain 95% of the explained variance, which guided us to use the elbow method for selecting the optimal number of principal components. Using PCA, we reduced the number of features from 2048 to 1186. For image data and features extracted from neural networks, this can help in identifying the most important features that capture the essential information. However there isn't a direct correlation between the explained variance of each feature with it's importance in the classification task, which could lead to poor results, like in our experiment

## 3.2 Feature Importance

Another way we tested to refine our feature set was to use feature importance scores from the various classifiers we trained. Each model provides a measure of feature importance that helps in identifying the most significant features for classification. Using the feature importance analysis, we reduced the number of features to the top 1000 based on their importance scores. This method complements PCA by providing another approach to dimensionality reduction.

## 3.3 Mutual Information

Mutual Information (MI) is a measure of the dependency between two variables. In the context of feature selection, MI helps identify features that have the highest dependency on the target variable, thereby providing a basis for selecting the most informative features. Although mutual information can provide insights into the relationship between features and the target variable, its applicability to features extracted from a deep learning model like ResNet50 can be limited. This is because these features are high-level features that the neural network has learned to be relevant for the task of image classification. These features are likely to be somewhat independent, as deep learning models are designed to learn useful representations that capture various aspects of the input data. However, we still explored mutual information scores to select the top 500 features that had the highest dependency on the target variable for comparison

# 4 Model Training

To achieve an optimal classification task in our study we trained different models and we then compared their performances to choose the most fit to the task. The classifiers used are: Support Vector Machine (SVM), Random Forest, AdaBoost, and XGBoost. Each classifier has unique characteristics and strengths.

## 4.1 Support Vector Machine

SVM is a supervised learning algorithm used for classification and regression tasks. It works by finding the hyperplane that best separates the classes in the feature space performing efficiently non-linear classification using kernel functions. It aims to maximize the margin between the classes, making it robust to overfitting. Since only a subset of training data points (support vectors) are used to define the hyperplane, the SVM classifier is memory efficient. For these reasons it's really effective in high-dimensional spaces, which is why we initially we thought it should be the most adequate for the task.

## 4.2 Random Forest

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) of the individual trees. Each tree is trained on a random subset of the data, which helps reduce variance and overfitting. At each split, a random subset of features is considered, making the model robust and reducing correlation between trees. These characteristics make this method good for handling large datasets and high-dimensional spaces, however it can be slow and memory-intensive.

## 4.3 AdaBoost

AdaBoost (Adaptive Boosting) is an ensemble learning technique that combines multiple weak classifiers to form a strong classifier. It adjusts the weights of incorrectly classified instances so that subsequent classifiers focus more on difficult cases. In particular misclassified instances are given higher weights, forcing the model to improve on these cases in subsequent rounds. Another key feature of this classifier is that weak classifiers are trained sequentially, each correcting the errors of the previous ones. For these reasons it's very sensitive to noisy data (or, like in our case, a huge number of features with different importance) and can overfit if the weak classifiers are too complex (like in our case with 2048 features). We expect that for our classification task this classifier will perform poorly.

## 4.4 XGBoost

XGBoost (Extreme Gradient Boosting) is an optimized gradient boosting that builds trees sequentially, each trying to correct the errors of the previous ones, similar to AdaBoost but with gradient descent optimization to minimize the loss function, improving the model iteratively. It incorporates regularization (L1 and L2) to prevent overfitting

and is optimized for parallel processing, making it fast and scalable to large datasets and high-dimensional spaces. For these reasons we expect this classifier to perform pretty well.

## 4.5 Results and Comparison

In this section, we detail the results obtained by the various classifiers we trained. The classifiers were evaluated on their ability to distinguish between different classes of mushrooms, with a particular focus on accuracy, precision and recall. We conducted four sets of experiments: using all features, using the top 1000 features selected through feature importance, using features reduced via Principal Component Analysis (PCA), and using the top 500 features selected through Mutual information technique.

| **Classifier** | **All Features** | **Cross Validation** | **Top 1000 Features** | **PCA** | **MI** |
|---|---|---|---|---|---|
| **Accuracy** | | | | | |
| SVM | 0.9125 | 0.9118 | 0.9000 | 0.9000 | 0.9116 |
| Random Forest | 0.8659 | 0.8594 | 0.8700 | 0.7700 | 0.8655 |
| AdaBoost | 0.6348 | 0.5792 | 0.6300 | 0.6600 | 0.5015 |
| XGBoost | 0.9183 | 0.9147 | 0.9200 | 0.8900 | 0.9137 |
| **Precision** | | | | | |
| SVM | 0.91 | 0.91 | 0.90 | 0.90 | 0.91 |
| Random Forest | 0.87 | 0.88 | 0.87 | 0.79 | 0.88 |
| AdaBoost | 0.65 | 0.56 | 0.65 | 0.67 | 0.55 |
| XGBoost | 0.92 | 0.92 | 0.92 | 0.90 | 0.91 |
| **Recall (Weighted)** | | | | | |
| SVM | 0.91 | 0.91 | 0.90 | 0.90 | 0.91 |
| Random Forest | 0.87 | 0.86 | 0.87 | 0.77 | 0.87 |
| AdaBoost | 0.63 | 0.57 | 0.63 | 0.66 | 0.50 |
| XGBoost | 0.92 | 0.92 | 0.92 | 0.89 | 0.91 |

Table 1: Performance metrics across different classifiers and feature selection methods

**OBSERVATIONS**:

1. **SVM and XGBoost** consistently perform well across all scenarios, with XGBoost generally achieving the highest accuracy. This suggests both are robust classifiers for this dataset, with XGBoost slightly outperforming SVM.

2. **Random Forest** shows variable performance, performing better with all features and feature selection but significantly worse with PCA and mutual information. This suggests that Random Forest did not benefit from PCA, possibly due to PCA transforming features into a space less interpretable by decision trees.

3. **AdaBoost** consistently underperforms, regardless of the feature set used, suggesting it may not be suitable for this problem as we expected.

4. **Feature Selection and PCA:** Both techniques show that dimensionality reduction can be effective, but the choice of method and the classifier's adaptability

to reduced features are critical. While PCA works well for SVM and XGBoost, it significantly affects Random Forest. On the other hand, since feature selection utilize importance scores from various models, it generally performs very well. The main drawback of this method is that it requires the model to be trained with the original feature space to be used.

5. The use of features extracted from multiple epochs of ResNet50 training introduces the potential for **overfitting**. High accuracies in training may not fully translate to unseen data, emphasizing the need for careful validation and potential use of regularization techniques. For example even if the results seems to indicate that mutual information performs pretty well across the various classifiers, it is probably subject to overfitting. This is why we tried the models we trained with another test set of 70 variegate images (10 for each class)

| **Classifier** | **All Features** | **Cross Validation** | **Top 1000 Features** | **PCA** | **MI** |
|---|---|---|---|---|---|
| **Accuracy** | | | | | |
| AdaBoost | 0.63 | 0.63 | 0.63 | 0.47 | 0.39 |
| Random Forest | 0.90 | 0.87 | 0.93 | 0.49 | 0.40 |
| SVM | 1.00 | 1.00 | 1.00 | 0.83 | 0.41 |
| XGBoost | 0.97 | 0.97 | 0.97 | 0.39 | 0.66 |
| **Precision** | | | | | |
| AdaBoost | 0.62 | 0.62 | 0.62 | 0.54 | 0.70 |
| Random Forest | 0.92 | 0.91 | 0.94 | 0.73 | 0.74 |
| SVM | 1.00 | 1.00 | 1.00 | 0.88 | 0.74 |
| XGBoost | 0.97 | 0.97 | 0.98 | 0.73 | 0.78 |
| **Recall (Weighted)** | | | | | |
| AdaBoost | 0.63 | 0.63 | 0.63 | 0.47 | 0.39 |
| Random Forest | 0.90 | 0.87 | 0.93 | 0.49 | 0.40 |
| SVM | 1.00 | 1.00 | 1.00 | 0.83 | 0.41 |
| XGBoost | 0.97 | 0.97 | 0.97 | 0.39 | 0.66 |

Table 2: Performance test with 70 images across different classifiers and feature selection methods

As we already thought, these results suggest that the features selected based on mutual information and PCA may have included noise or patterns specific to the training data, leading to poor generalization on the test set and resulting in overfitting. Both mutual information and PCA aimed to select the most informative features, but they inadvertently included noisy patterns, leading to significant misclassifications.

In our study, we also considered the application of the forward selection method for feature selection. Forward selection is a technique where features are incrementally added to the model one at a time, starting with an empty set. At each step, the feature that most improves the model's performance is added to the set until no significant improvement is observed. By carefully selecting the most relevant features, forward selection could potentially enhance the performance of the model and reduce overfitting by including only the most significant features, thereby simplifying the model. However, due to the high dimensionality of our feature space, performing forward selection

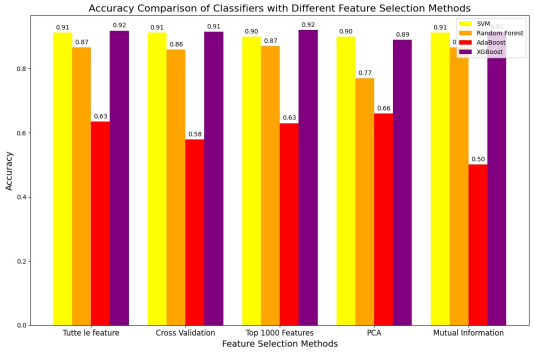
required substantial computational resources, including processing power and memory. Our available resources were insufficient to efficiently handle this process within a reasonable timeframe.

# 5 Conclusions

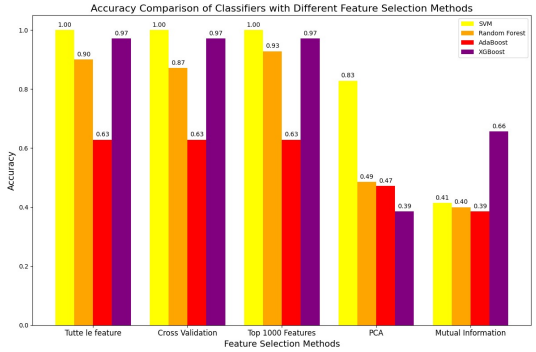
We found that SVM and XGBoost consistently delivered high accuracy across various scenarios, with XGBoost slightly outperforming SVM. These results indicate that both classifiers are well-suited for this task, handling the complexity of the data effectively. Random Forest showed a mixed performance, excelling when all features were used but struggling with dimensionality reduction methods like PCA, likely due to its reliance on interpretable features for decision making. Conversely, AdaBoost consistently underperformed, affirming our expectation that it is less suitable for datasets with high feature complexity.

When it came to feature reduction, techniques like PCA and feature importance generally improved computational efficiency, but their impact varied by classifier. SVM and XGBoost adapted well to PCA, while Random Forest did not, suggesting that PCA may not always be the best fit for models reliant on feature interpretability. Using feature importance scores also provided good results but required initial training with the full feature set, posing potential computational challenges.

The evaluation on the set of 70 new images highlighted the risk of overfitting, especially with features selected through mutual information and PCA. These methods inadvertently included patterns specific to the training data, leading to poorer generalization and performance on unseen data.




(a) Validation results




(b) Test results using 70 unseen images

# 6 Future Work

In order to further enhance the performance and robustness of our mushroom classification model, several avenues for future work can be explored. These enhancements aim to address potential limitations in preprocessing, feature extraction, and model optimization.

- **Advanced Preprocessing Techniques**:
  - **Grayscale Conversion**: converting images to grayscale can reduce the complexity of the data by eliminating color information and can help in focusing on texture and shape, which are more relevant features.
  - **Background Elimination**: implementing background elimination techniques can improve the quality of the features extracted. By removing irrelevant background information, the model can better focus on the mushroom itself, potentially improving classification accuracy.
- **Augmentation Techniques**: applying various data augmentation techniques such as rotation, scaling, flipping, and cropping can help in creating a more robust dataset. This can enhance the model's ability to generalize to unseen data and reduce overfitting.
- **Advanced Feature Selection Methods**: implementing more sophisticated feature selection methods such as Recursive Feature Elimination (RFE) or regularized models (e.g., Lasso) could help in identifying the most informative features.
- **Automated Hyperparameter Optimization**: utilizing automated hyperparameter optimization techniques such as Grid Search, Random Search, or Bayesian Optimization can help in finding the optimal set of hyperparameters for the models (although we tried these methods, in particular for AdaBoost, but the computational complexity rendered the experiment not doable)

There are multiple directions for future work to enhance the performance and applicability of our mushroom classification model. By implementing advanced preprocessing techniques, exploring different feature extraction methods, and utilizing robust feature selection and model optimization strategies, we can further improve the accuracy and reliability of our classifier. Expanding the dataset and testing the model in real-world scenarios would also contribute to its practical utility and effectiveness.