Final Year Project Report

Full Unit - Final Report

Machine learning for classifying mushroom edibility

Rickus de Goede

A report submitted in part fulfilment of the degree of

BSc (Hons) in Computer Science

Supervisor: Nicolo Colombo

Department of Computer Science
Royal Holloway, University of London

February 18, 2021

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count:

Student Name: Rickus de Goede

Date of Submission:

Signature:

Table of Contents

Abstract 4

Project Specification 5

1 Introduction 7

1.1 Aims and Objectives 7

1.2 Image Data-set used 7

1.3 Algorithms used 9

1.4 Literature survey 10

2 Theory 12

2.1 What is classification? 12

2.2 Nearest neighbors 12

2.3 k-nearest neighbors definition 13

2.4 Decision tree 15

2.5 Support Vector Machines 18

2.6 Cross validation 19

2.7 Confusion Matrices 19

2.8 ROC curve 19

3 Implementation 21

3.1 Which programming language was used? 21

3.2 Data pre-processing 21

3.3 Modules and Notebooks 22

3.4 Data normalisation 24

3.5 Testing nearest neighbor module functionality 24

3.6 Testing classification tree module functionality 24

4 Experiments 25

5 Results . . . 26
6 Professional issues . . . 27
6.1 Safety . . . 27
6.2 Licensing and use of dataset . . . 27
6.3 Plagiarism . . . 28
6.4 Management . . . 28
6.5 Privacy and data-protection . . . 28
7 Diary . . . 29
8 Bibliography . . . 31

Abstract

What this project is going to achieve is to program and implement supervised classification algorithms to test on an image mushroom data-set. These different algorithmic approaches are k-nearest neighbors, a decision tree based approach, a support vector machine and a neural network. The models created by these algorithms will predict, based on only features extracted from the image data-set, what the species of the mushroom in the image is (multi-class classification problem) then this will give whether or not a picture of a mushroom is edible or poisonous by giving the edibility of the species predicted (a binary classification problem) as there are no set general rules for mushroom edibility across species this is the best approach. Key results metrics will be generated that are required for the comparison of the algorithms.

Project Specification

Aims: To implement and compare on benchmark data sets various machine learning algorithms

Background: Machine learning allows us to write computer programs to solve many complex problems: instead of solving a problem directly, the program can learn to solve a class of problems given a training set produced by a teacher. This project will involve implementing a range of machine learning algorithms, from the simplest to sophisticated, and studying their empirical performance using methods such as cross-validation and ROC analysis. In this project you will learn valuable skills prized by employers using data mining.

Early Deliverables

- You will implement simple machine learning algorithms such as nearest neighbours and decision trees.
- They will be tested using simple artificial data sets.
- Report: a description of 1-nearest neighbour and k-nearest neighbours algorithms, with different strategies for breaking the ties;
- Report: a description of decision trees using different measures of uniformity.

Final Deliverables

- May include nearest neighbours using kernels and multi-class support vector machines.
- The algorithms will be studied empirically on benchmark data sets such as those available from the UCI data repository and the Delve repository.
- For many of these data set judicious preprocessing (including normalisation of attributes or examples) will be essential.
- The performance of all algorithms will be explored using a hold-out test set and cross-validation.
- The overall program will have a full object-oriented design, with a full implementation life cycle using modern software engineering principles.
- Ideally, it will have a graphical user interface.
- The report will describe: the theory behind the algorithms.
- The report will describe: the implementation issues necessary to apply the theory.
- The report will describe: the software engineering process involved in generating your software.
- The report will describe: computational experiments with different data sets and parameters.

Suggested Extensions

- Modifications of known algorithms.
- Dealing with machine learning problems with asymmetrical errors (such as spam detection) and ROC analysis.
- Nontrivial adaptations of known algorithms to applications in a specific area, such as medical diagnosis or option pricing.
- Exploring the cross-validation procedure, in particular the leave-one out procedure. What is the optional number of folds?
- Comparative study of different strategies of reducing multi-class classifiers to binary classifiers (such as one-against-one, one-against-the-rest, coding-based).

Chapter 1: **Introduction**

1.1 Aims and Objectives

Aim: To implement and compare algorithms for classifying mushroom edibility with image data, optimize and build a model with the high accuracy generalised performance. Why the project was chosen to be about image classification is because computer vision is a field of computer science that I would like to work in therefore comparing the algorithms in relation to computer vision makes sense. The reason mushrooms were chosen is that it is a very difficult problem to determine whether a mushroom is poisonous or not, which makes it a difficult computer vision problem; it is difficult as there are no real set rules for edibility. If given two very similar mushrooms, one can be poisonous, the other edible. I want to take a different approach to the approaches taken in the wider academic publishing on mushroom image classification and focus on using the color information from image for the classification, the reason being is that this technique could have wider applications to image classification problems of objects that are not photographed uniformly and from the same angle.

Objectives:

1. Program k-nearest neighbor algorithm.
2. Program a classification decision tree.
3. Program a random forest
4. Program a support vector machine
5. Pre-process the image data, this is done by reducing the noise by removing the background of the images, extracting colour information about the images and extract shape information.
6. Applying the algorithms to the pre-processed data of the mushroom images.
7. Getting accuracy scores from applications of all algorithms on both data-sets.
8. Generating confusion matrices to display the binary classification results.
9. Generating ROC curves for all algorithms.
10. Comparing the results and discussing how best to improve them.

1.2 Image Data-set used

The mushroom images used are sourced from the Danish Svampe Atlas [21] and split into six categories with one hundred images each: Agaricus Arvensis, Amanita Muscaria, Amanita Rubescens, Boletus Edulis, Cortinarius Rubellus and Galerina Marginata. Each are species of mushrooms. Where Agaricus Arvensis, Amanita Rubescens and Boletus Edulis are edible and Amanita Muscaria, Cortinarius Rubellus and Galerina Marginata are poisonous. This data-set is exactly half edible, half poisonous.

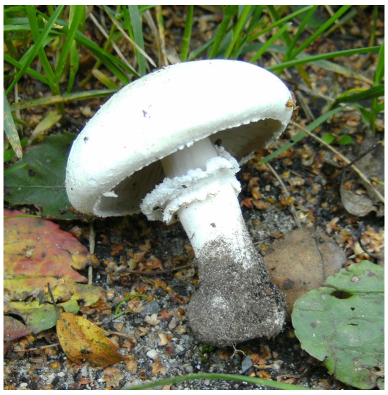


Figure 1.1: Agaricus_Arvensis

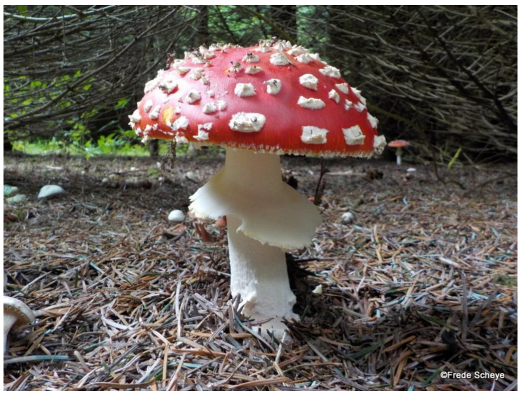


Figure 1.2: Amanita_Muscaria

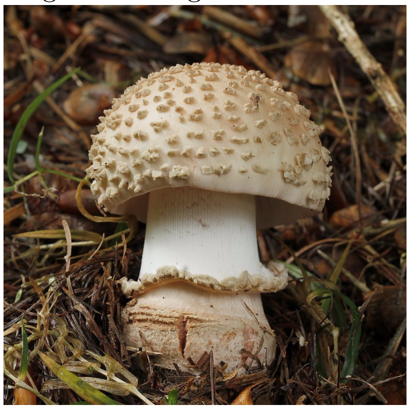


Figure 1.3: Amanita_Rubescens

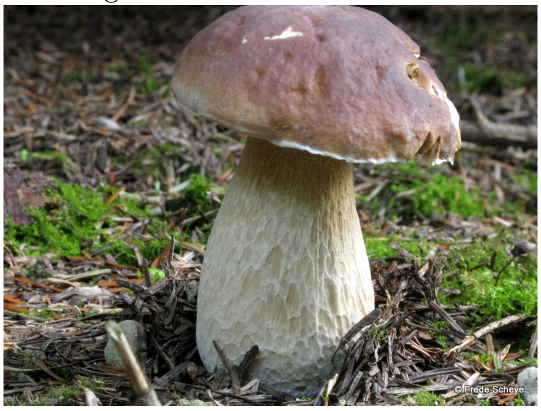


Figure 1.4: Boletus_Edulis

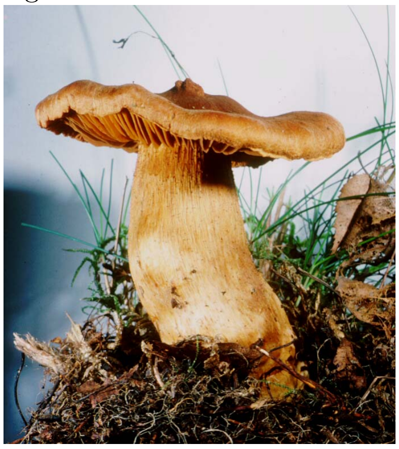


Figure 1.5: Cortinarius_Rubellus

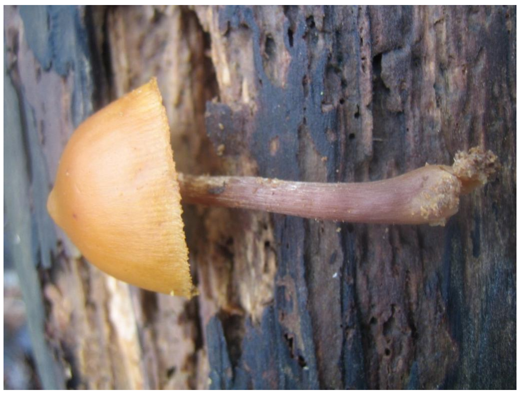


Figure 1.6: Galerina_Marginata

Figure 1.7: An example of each species of mushroom used in this project

1.3 Algorithms used

1.3.1 Algorithm selection

Nearest neighbors algorithm was chosen as it is a relatively simple algorithm to implement and optimise due to only having one parameter k therefore is a good first port of call in most machine learning problems.

In the mushroom kingdom representation in biology, species are organised in a tree like structure therefore I found the question interesting of whether or not the structure human experts have created to classify species results in a high performing machine learning algorithm for classification.

Random forest was chosen due to the fact that decision trees are prone to over-fitting and by implementing random forests could yield better performance.

Support vector machines with a linear kernel can be very powerful in binary classification problems with linearly distributed data, however due to the fact that the project is implementing a multi-class classification problem of classifying species and then reducing this to the binary edibility classification problem, there are complexities involved; overcoming these complexities will make this project more interesting. With the fact that SVMs are said to be just as powerful as one layered neural networks and much easier to train I will be interested to see if performance rivals the other algorithms.

Convolutional neural networks were not chosen to be implemented because even though they perform extremely well with this kind of problem, they require a very large data-set to get good results, my dataset is not large enough, therefore I will be attempting a "smarter" approach that requires less training data.

1.3.2 k-nearest neighbors

The Nearest-Neighbors algorithm is a simple, yet highly effective supervised algorithm in machine learning. It has a tendency to over-fit with small values of k, predictions are often accurate but can be unstable, however can achieve high accuracy for such a simple model. Proposed by Thomas Clover in the paper "Nearest neighbor pattern classification"[9] for use in classification and regression.

1.3.3 CART

"Tree-based methods partition the feature space into a set of rectangles, and then fit a simple model (like a constant) in each one. They are conceptually powerful. A popular method for tree-based classification and regression is called CART." [20]. The term CART was created by Leo Breiman to name decision tree algorithms that can be used for regression or classification problems.

1.3.4 Random Forest

Random forest (also called random decision forest) is an ensemble method for learning that is used for classification by constructing multiple different decision trees whilst training and outputting the class that is the mode of the classes predicted by the individual trees in the set of trees. Also proposed by Leo Brieman[5]

1.3.5 Support Vector Machine

The idea of an SVM is create a hyperplane that best splits the data into two classes. Support vectors are the points which are closest to the hyperplane which if changed, will move the hyperplane. Due to the fact that the hyperplane best splits the data-set into two classes developed by Vladimir Vapnik. Various approaches can be taken to apply this algorithm to a multi-class classification problem for example, one vs rest and one against one vs one.

1.4 Literature survey

The paper ”Classification of Mushroom Fungi Using Machine Learning Techniques” [13] found that their experimental results showed that the best technique for classifying mushroom images is kNN with accuracy of 94% based on features extracted from images with dimensions of mushroom types, and 87% extracted from images only. This study had access to the real dimensions of the mushroom (meaning actual measurements of size) therefore they could give size and shape data to their algorithms, my project does not have access to real dimensions therefore results are expected to be more in-line with the 87% achieved with features only extracted from the images.

The paper ”Image Analysis of Mushroom Types Classification by Convolution Neural Networks”[14] found that with 45 species of mushrooms in their data-set 35 being edible and 10 being poisonous. Their proposed model gives the results of 0.78, 0.73, 0.74 of precision, recall and F1 score, respectively. The image dataset of 35 types has 6,000 images and the image dataset of poisonous mushrooms has 2,556 images, this is a lot higher than the 600 total images my project has. Therefore I will not be expecting to see the impressive results of this study.

The paper ”A Zoning based Feature Extraction method for Recognition of Handwritten Assamese Characters”[11] Even though not about mushroom image classification but about character recognition helped in me understanding why zoning was not applicable to the mushroom image classification problem. This strategy works well for characters because characters are always ”upright” and exist in a two-dimensional space, therefore checking zones of the image for pixel density, turns out to be extremely beneficial. The mushroom image space is one of three-dimensional space where the 2-d zoning pixel density values do not provide much use.

1.4.1 Examples of applications of k-nearest neighbors

- Breast cancer diagnosis [17]
- Tea category identification on optimal wavelet entropy [24]
- Noninvasive blood pressure classification [19]
- Diabetes classification [8]
- Financial modeling [25]

This is a very versatile algorithm with application in a huge variety of fields.

1.4.2 Examples of applications of CART

- A statistical tool to investigate risk factor interactions with an example for colon cancer [7]
- Predicting stroke inpatient rehabilitation outcome [10]
- Tomato plant disease classification in digital images. [16]
- Credit risk modelling [26]
- classification of Takayasu arteritis (inflammation of the arteries) [2]

As displayed above, both have similar popular applications in biology and finance. This project lands in the domain of biology, therefore this project seems to be in the domain of the general application of these algorithms. Tomato plant disease classification in digital images [16] is a standout example as it involves image classification.

Chapter 2: **Theory**

2.1 What is classification?

Classification is the act of predicting the label of sample just given it's features. Labels are also called classes, targets and categories. Classification predictive modeling is the creating a mapping function from input variables (which is commonly called X) to discrete output variables (which is commonly called y).

Classification is a type of supervised learning where the input data is given with the labels. There are two types of learners in classification, there are lazy learners and eager learners.

Lazy learners just store the training data and wait for testing data to be given and when it is classification is conducted based on the most closely related data in the training data. Compared to eager learners, lazy learners take less time to train but take longer to predict. k-nearest neighbors is an example of a lazy learner.

Eager learners create a classification model trained on data before receiving data for classification. It must be able to create a singular hypothesis that covers the entire instance space. Because of the fact that the model must be constructed when training, training takes a long time and the model takes less time to predict. Decision trees, random forests and support vector machines are all examples of eager learners.

[3]

2.2 Nearest neighbors

2.2.1 Different measures of closeness

Let a and b be two n-dimensional vectors,

Dot product:

$$a \cdot b = \sum_{i=1}^{n} a_i b_i$$

L1 norm (Manhattan/Taxi-cab distance):

$$||x||_1 = \sum_{i=1}^{n} |x_i|$$

L2 norm (Euclidean distance):

$$||x||_2 = \sqrt{x \cdot x}$$

p-norm:

$$||x||_p = (\sum_{i=1}^{n} |x_i|^p)^{1/p}$$

where a_i is the i^{th} component of a , b_i is the i^{th} component of b, n is dimension of vector space , x is a vector, x_i is the i^{th} component of x. L1 and L2 norms are p-norms where $p = 1$ and $p = 2$ respectively. All the norms can be expressed as dot products. L2 norm is the most popular used distance metric. Here is L2 norm expressed as a combination of dot products where x is a sample in the training set and x' is a test sample and $dist$ is the distance function:

$$dist(x, x') = \sqrt{(x \cdot x) + (x' \cdot x') - 2(x \cdot x')}$$

Due to the fact that distance can be expressed in dot products we can apply the kernel trick to the method and implement the kernel of choosing by swapping out the dot product with kernel function of choice K, so $dist$ becomes:

$$dist(x, x') = \sqrt{K(x, x) + K(x', x') - 2K(x, x')}$$

[22]

Neighbourhood function

$$D_i = dist(x', X_i)$$
$$N_0 = \{\}$$
$$N_{i+1} = N_i \cup \{min(D/N_i)\}$$

$N_k(x')$ is the neighbourhood of x', where k is number of nearest neighbors. D is the set of distances, x' is the test vector, X is the training set, $dist$ is the distance metric of choosing:

2.3 k-nearest neighbors definition

Nearest-Neighbor methods use elements in the training set closest to x' to form $\widehat{Y}$. The k-nearest neighbor fit for $\widehat{Y}$ (for classification) is defined as follows:

$$\widehat{Y}(x') = mode(y_i | x_i \epsilon N_k(x')),$$ [20]

where $mode()$ is the most frequently occurring number, $N_k(x)$ is the neighbourhood of x';

the neighborhood is defined by the k closest elements in the training set. We must define a way of measuring closeness and therefore define a distance metric, often euclidean distance is used. We have now seen the two parameters of the k-nearest neighbor model, these are the value of k and the metric chosen to be used for closeness. For regression problems we do not use the mode of the labels given by $x_i \epsilon N_k(x')$, we instead use the mean.

2.3.1 1-nearest neighbor definition

The nearest neighbor method is the k-nearest neighbor model when $k = 1$ therefore the nearest neighbor fit for $\widehat{Y}$ (for regression and classification) is defined as follows: $\widehat{Y}(x') = y_i | x_i \epsilon N_1(x')$

2.3.2 k-Nearest Neighbors in pseudo-code

1. Calculate distance of test sample to all samples in training-set.
2. Arrange these distances in ascending order.
3. Select the samples that produced the first k distances.
4. Then for regression calculate the mean of the labels of this sample and this is the predicted label for the test sample or for classification calculate the mode of the labels and this would be the predicted label for the test sample.

2.3.3 Methods for tie-breaking

Especially when using k-nearest neighbors for classification problems it is possible to encounter a situation when there is a draw between labels of the nearest samples. There are a few approaches to dealing with this problem:

- **Random tie-breaking:** This method breaks ties as randomly as possible, this solves the tie-break however is inconsistent as every-time the same tie-break happens a different outcome is given; This could make the algorithm's accuracy scores less consistent.
- **Using weights:** This gives each sample an importance metric, this can often be the 1/distance to the test sample, then if there is a tie between labels the weightings of the samples are added up and the samples of the same label that have the highest sum of weights wins the tie-break. However, this creates the opportunity for another form of tie-break which can happen if the sum of weightings are equal, then you still haven't solved the tie-break.

2.3.4 Selecting k

A small value of k is more sensitive to noise of the data and is a more complex model. A larger value of k is less sensitive to noise and is a simpler model. The optimal model complexity achieves higher the larger the training set becomes.

Using Cross Validation: Using the training samples and labels calculate a cross validation score for a range of values of k this gives an idea of generalisation. Then choose the value of k that achieves the highest cross validation score, this doesn't involve data snooping as only the training set is used.

2.3.5 Strengths and weaknesses of k-nearest neighbors

A strength of the k-nearest-neighbors model is that it is simple and easy to understand, and often gives good performance without too many adjustments of parameters. It is good to try before thinking of using more advanced techniques.

Another strength is that the algorithm could be implemented in an online format very easily as training is O(1) time complexity, so re-training with new data iteratively is possible.

A strength is that building the model is fast. A weakness however is prediction can be slow for data sets with large number of features and or a large number of samples. For k-nearest-neighbors time complexity for training is O(1) but time complexity for testing is O(N).

The model often does not perform well when a data-set has many features and when many of those features are 0 which is called a sparse data-set.

Another weakness is that it is important to pre-process your data so that distance measurements between features are scaled to the range of the features.

A further weakness is that due to storing the training-set and working out distances, the algorithm is very memory intensive compared to other algorithms.

2.4 Decision tree

Classification trees

The algorithm tries to split the training-set on all the classes contained in features in turn, then applies an impurity measure on all the splits and selects the class that caused the purest split (not equal), then recursively calls itself on the split data-sets until a pre-defined stopping criterion for leaf size is reached. The tree is comprised of the resulting category questions that will predict a test sample by sending it to a leaf node in the tree and the most common label in the leaf node is returned.

2.4.1 Impurity metrics

Algorithms for constructing decision trees usually work top-down, by choosing a variable at each step that best splits the set of items. [15]

2.4.2 Gini impurity

for j classes, $i\epsilon(1,2,3,...,j)$, p_i is the fraction of items labelled with class i in the set:

$$gini(p) = 1 - \sum_{i=1}^{j} p_i^2$$

2.4.3 Information gain

Used by C4.5, C5.0 and ID3, tree-generation algorithms. Information gain is based on the concept of entropy and information content.

$$gain(p_1, p_2, ..., p_i) = -\sum_{i=1}^{j} p_i log_2 p_i$$

[23]

2.4.4 Classification decision-tree in pseudo-code

"

1. Assign all training instances to the root of the tree. Set curent node to root node.
2. For each attribute
   (a) Partition all data instances at the node by the value of the attribute.
   (b) Compute the information gain ratio from the partitioning.
3. Identify feature that results in the greatest information gain ratio. Set this feature to be the splitting criterion at the current node.
   - If the best information gain ratio is 0, tag the current node as a leaf and return.
4. Partition all instances according to attribute value of the best feature.
5. Denote each partition as a child node of the current node.
6. For each child node:
   (a) If the child node is "pure" (has instances from only one class) tag it as a leaf and return.
   (b) If not set the child node as the current node and recurse to step 2.

" [4]

2.4.5 Pre-pruning

In pre-pruning, we manipulate parameters such as minimum leaf size, maximum depth of the tree, threshold for purity of split and use them as stopping criterion to avoid over-fitting. This can be tricky to carry out in practice as picking the optimum time to stop the tree is difficult to find out.

2.4.6 Post-pruning

"In post-pruning first, it goes deeper and deeper in the tree to build a complete tree. If the tree shows the over-fitting problem then pruning is done as a post-pruning step. We use a cross-validation data to check the effect of our pruning. Using cross-validation data, it tests whether expanding a node will make an improvement or not.

If it shows an improvement, then we can continue by expanding that node. But if it shows a reduction in accuracy then it should not be expanded i.e, the node should be converted to a leaf node." [4]

2.4.7 Limitations of CART

- **Instability of trees:** A big issue with decision trees is that they have high variance. A very small alteration in the data-set can lead to different splits. A major reason for this instability is that the nature of the model is hierarchical, the effect of an error on a split at the top of the tree has an effect on all the splits underneath it, a way of getting round this issue could be to have a more stable split criterion however the instability does not go away. It is the cost of having a simplistic tree model.[20]
- **Lack of smoothness:** Trees do not produce a smooth prediction surface due to the partitioning of the feature space into a set of rectangles, these sharp decision boundaries, this becomes more of an issue when it comes to a regression problem as then it would be expected that the underlying function would be smooth.[20]
- **Binary splits:** We could consider using multi-way splits at each level instead of using binary splits, while this is possible it is usually not the best approach to go with as it can lead to the data getting split too quickly, which doesn't leave enough data lower down in the hierarchy. Multi-way splits can be represented with combinations of binary split, therefore the most commonly used split is binary.[20]
- **Over-fitting:** Individual tree methods have a tendency to over-fit however this issue can be overcome by the use of ensemble methods such as bagging and random forest.

2.4.8 Ensemble methods

- **Bagging:** Also called bootstrap aggregation, is used when we need to avoid over-fitting and minimise the variance of the tree. We create multiple subsets from the training sample chosen at random. Each section of subset data is now used to train the decision trees. We end up with the resulting ensemble of multiple models each trained on a different section of the training-set. We then use the average prediction from all trees in the ensemble which results in a more robust model than using a singular tree. [12]
- **Random forest:** This method extends bagging. It goes one step further where on top of taking the random subsets of the training-set, it also takes a random selection of features, therefore it drops certain features and only uses the selected features to grow the tree. A strength of random forest is it handles data with high-dimensionality well due to random feature selection; however this model can lack in precision in regression problems due to the fact that the mean prediction is taken from all singular trees in the random forest. [12]
- **Boosting:** This method is an ensemble technique to create multiple predictors. In this technique, models are trained in sequence with early models training simple trees on the subsets of the training-set and then scrutinising the results for errors. Another way

to explain this is we fit consecutive trees (using a random subset of the training set like in the other ensemble methods) and after every tree, the aim is for the next tree to solve for the sum of error from the previous tree. [12]

- **Gradient boosting:** This method is an extension of boosting involving the use of gradient descent. The gradient descent algorithm can optimize any loss function that is differentiable. Due to the fact that the trees are built in turn trying to minimise the net loss, the gradient descent algorithm can minimise this loss. The next tree can use this algorithm to recover the loss (this loss is the difference between predicted and actual values). A strength of this method is that different loss functions can be used however it does tend to over-fit, also it requires tuning of hyper-parameters. [12]

2.5 Support Vector Machines

2.5.1 The formulation of a support vector machine

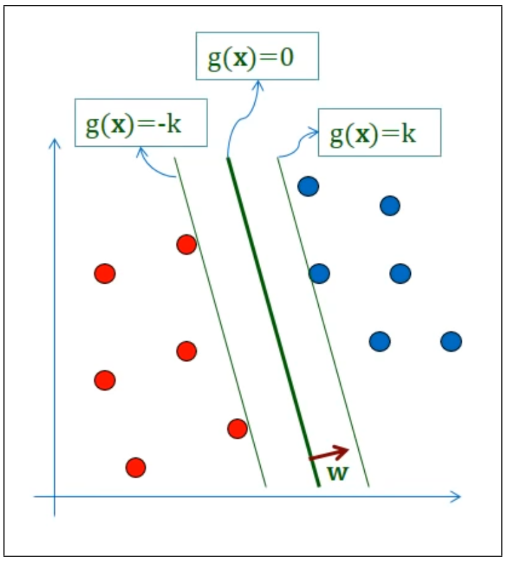


Figure 2.1: An example of a margin

$$g(x) = w^T x + b \tag{2.1}$$

Maximise k such that:

$$-w^T x + b \geq k \quad \text{for} \quad d_i = 1 \tag{2.2}$$
$$-w^T x + b \leq k \quad \text{for} \quad d_i = -1 \tag{2.3}$$

Value of $g(x)$ depends upon $||w||$:

1. Keep $||w|| = 1$ and maximise $g(x)$ or,
2. $g(x) \geq 1$ and minimise $||w||$

[1]

2.6 Cross validation

2.6.1 Definition

Divide up the data into an arbitrary number of segments (called folds) use each segment in turn as the test set and the rest as the training set. The result is then summarised by adding the tallies of the correct and incorrect results. When the the number of folds is K and the sample size is m, if $K = m$ this is called "leave one out cross validation".

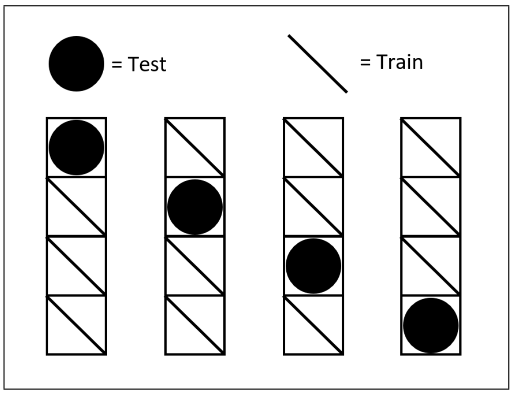


Figure 2.2: Four Fold Cross Validation

2.6.2 Why use cross validation?

The reason cross validation is used is to give a sense of generalisation only using the training-set, this gives an idea of the validity of the model. This can also be used for parameter selection as the best performing parameter in cross validation will often be the best generalised parameter.

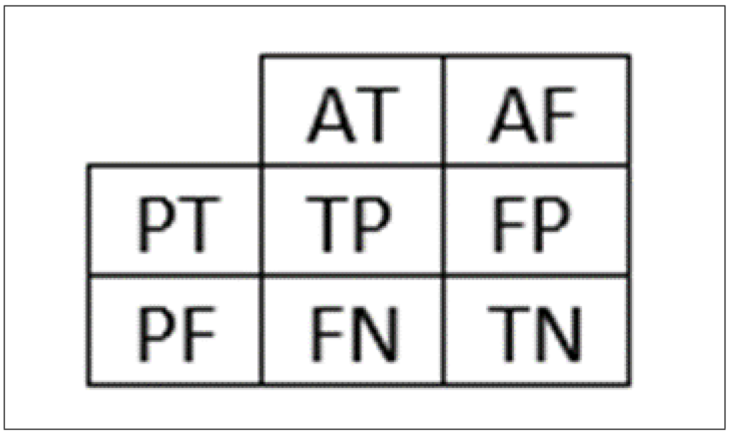
2.7 Confusion Matrices

Matrix to represent the number of true positives(TP), true negatives(TN), false positives(FP) and false negatives(FN). Below is a graphical representation of matrix that would be used for a binary classification problem. Where AT stands for actual value true, AF stands for actual value false, PT stands for predicted value true, PF stands for predicted value false. This format will be used for all confusion matrices in this report.

2.8 ROC curve

2.8.1 Definition

ROC stands for receiver operator characteristic, it is the curve that results in plotting 1 - specificity (false positive rate) on the x-axis and sensitivity on the y-axis.



	AT	AF
PT	TP	FP
PF	FN	TN

Figure 2.3: Binary Confusion Matrix

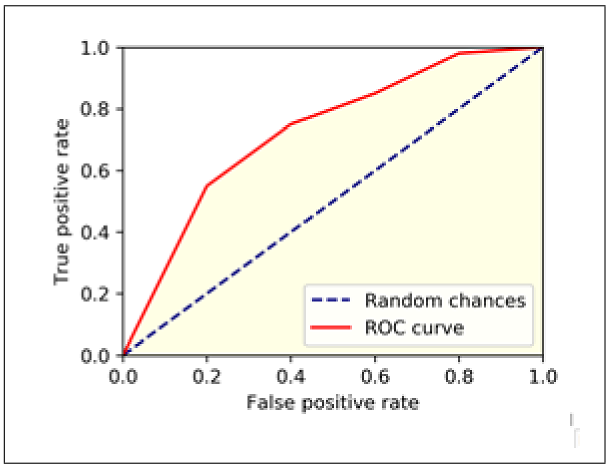


Figure 2.4: Example of a ROC curve [6]

$$Sensitivity = \frac{True\_Positives}{True\_Positives + False\_Negatives}$$

$$Specificity = \frac{True\_Negatives}{True\_Negatives + False\_Positives}$$

2.8.2 AUC (area under the curve)

The AUC is the area under the ROC curve. The larger the area under the ROC curve the better the classifier is. AUC gives an aggregate measure of performance over all classification thresholds.

Chapter 3: **Implementation**

3.1 Which programming language was used?

The programming language used was python however as python is a relatively slow language the program relies on using many Numpy library functions as the functions of this library are written in C which run much faster. The program encountered heavy performance issues when the interpreter would spend too much time in the python for-loops, therefore, where possible, python for-loops were avoided. As these algorithms are relatively simple the slow performance is bearable and the results are accurate.

3.2 Data pre-processing

3.2.1 Why pre-process the data?

The algorithms that I'm using (with the exception of neural networks) tend not to perform well on raw pixel data as the features are convoluted and don't hold meaning themselves, it's when a group of pixels is taken that meaning full information is given. Feeding an algorithm like nearest neighbors raw pixel data is like giving a person the constituent pixels of an image and no positional data and asking them to tell you what species they are looking at.

3.2.2 Resizing the images

All images were resized to the size 200 by 200 pixels, this allows the data of the images to be comparable and have the same number of features. What is lost in this process however are the original proportions of the image but this is not a major issue as most of the images in the data-set are of the mushrooms in different angles and rotations.

3.2.3 Removing the background

The background of the mushroom images are noise because the pixel information they give do not have an effect on what species of mushroom is in the image. therefore using the OpenCV python module, the backgrounds of all the images in the data set were removed.

3.2.4 Creating a colour histogram

This process works by creating bins which each cover a color intensity interval and the bins count up every-time they get a pixel that lands in the bin's intensity interval, there are a set of bins for red intensities, a set for green and a set for blue, the more bins the smoother the graph. What is very useful about the colour information generated by the color histogram is that it does not depend at all on the positioning of the mushroom in the image. Using 8 bins for red, blue and green this process produced 512 features for each sample.

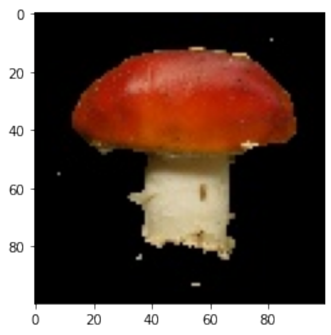


Figure 3.1: Image of Amanita Muscaria

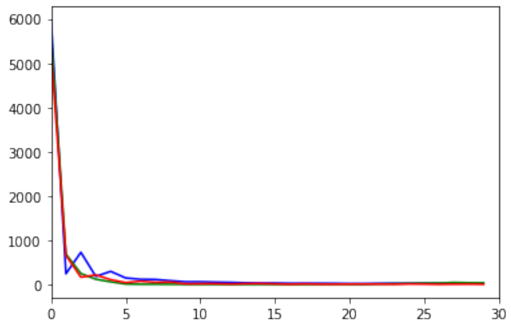


Figure 3.2: Colour histogram of image where red line is red intensity the blue line is blue intensity and the green line is green intensity

Figure 3.3: An example of an image and its colour histogram

3.2.5 Why not use shape information?

It was attempted to use canny line detection which is an algorithm used for producing a new image with just white lines on a black background in-order to get shape data on resized forty by forty pixel, gray-scale mushroom images . The issue that was had with this data was the non-homogeneity of the angles the images were taken from, therefore the wide variety of shapes lead to poor performance of the algorithms. If I could get a data-set were the images of the of the mushrooms were all from the same angle then shape data would have resulted in a useful data-set. I decided to omit this data from my project as it seemed to hinder the algorithms in preliminary exploratory tests.

3.3 Modules and Notebooks

(This section has not been updated to include Random Forest and SVM yet as they are not fully programmed)

3.3.1 UML

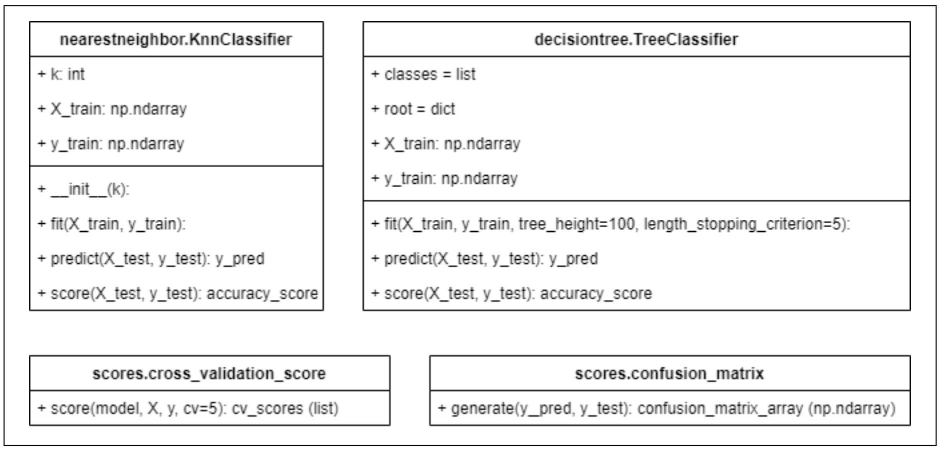

nearestneighbor.KnnClassifier
+ k: int
+ X_train: np.ndarray
+ y_train: np.ndarray
+ __init__(k):
+ fit(X_train, y_train):
+ predict(X_test, y_test): y_pred
+ score(X_test, y_test): accuracy_score
decisiontree.TreeClassifier
+ classes = list
+ root = dict
+ X_train: np.ndarray
+ y_train: np.ndarray
+ fit(X_train, y_train, tree_height=100, length_stopping_criterion=5):
+ predict(X_test, y_test): y_pred
+ score(X_test, y_test): accuracy_score
scores.cross_validation_score
+ score(model, X, y, cv=5): cv_scores (list)
scores.confusion_matrix
+ generate(y_pred, y_test): confusion_matrix_array (np.ndarray)

Figure 3.4: UML of classes in this project

3.3.2 Description of modules and notebooks

"scoring/scores.py" module

This module was programmed and includes the classes cross_validation_score and confusion_matrix. Cross_validation_score is a correct implementation of the cross validation procedure, it takes a data-set then partitions the data-set into folds using numpy slicing index then trains and tests on selected folds in turn and returns an array of scores from the cross validation. Confusion matrix takes as parameters two numpy arrays, one is the predicted values, and the other is actual values; it then returns a numpy array with the shape and contents of a binary classification problem's confusion matrix.

"algorithms/nearestneighbor.py" module

This module includes the class KnnClassifier which contains three essential methods for carrying out classification problems which are: fit, predict and score. Fit takes in the training data for the model to use, predict returns an array of postulated labels produces by the algorithm given a test-set and score returns the accuracy score which is also $1-$ error_rate, where the error_rate is the number of miss-classified samples divided by the number of samples.

"algorithms/decisiontree.py" module

This module also contains the fit, predict, and score methods, however this module does not work for multi-class classification as it was not needed to be implemented.

"notebooks/data-visualization-cleaning" notebook

This notebook is used to load the image data from the image_data files and assigns the correct binary label depending on which file the sample came from (edible=0 or poisonous=1). Also, the images are compressed down from their original size down to (100,100) resolution which then gives 300 features for the data-set, and it then saves the files to mushroom_image_data.csv for the pixel-data samples and the corresponding labels are saved in mushroom_image_target.csv.

"notebooks/nn-knn"

This notebook is used to train and test the k-nearest neighbor algorithm and to generate all graphs and performance results shown in this report.

"notebooks/decision-tree"

This notebook is used to train and test the decision tree classification algorithm and to generate all graphs and performance results shown in this report.

3.4 Data normalisation

It is important to normalise the features so that they are measured on roughly the same scale. It is important to avoid data-snooping when carrying out normalisation therefore the test set should not be used. There are a few classes in the sci-kit learn package for python available for this purpose. I will outline the Scalers and explain how they function.

- **StandardScaler:** For every feature in the resulting data-set the mean is zero and the variance is one.
- **RobustScaler:** Still makes sure that the features are measured on the same scale but uses the median and quartiles to avoid using outliers in the calculations. It shifts the data down by its median and divides by the interquartile distance. Outliers can be an issue for other Scalers.
- **MinMaxScaler:** Moves data so that all features lay inbetween zero and one by subtracting the minimum value for a feature from all values for that feature and dividing by the maximum value for that feature.
- **Normalizer:** This divides each sample by the euclidean norm of the sample. This means that each sample is normalised independently therefore the training and test set can be normalized together with this approach as there won't be any data snooping.

The StandardScaler was chosen as the Scaler of choice on the image-data due to the variance controlling nature. Every step necessary was taken to avoid data-snooping, scalers fit to a training set were only applied to the validation set and test set. The scaler used was never fit using the validation set or the test set.

3.5 Testing nearest neighbor module functionality

There is a python notebook to test the results of the project's version of k-nearest neighbors against sci-kit learns implementation and although the results were similar on sci-kit learns iris data-set (Chosen as flower classification is similar to mushroom classification) there are small differences that are believed to come from different methods for tie breaking as the iris classification problem is a multi-class classification. However tie-breaking will not be a large issue on application as this is a binary classification problem, therefore if an odd number for k is chosen then there will not be ties.

3.6 Testing classification tree module functionality

There is a python notebook to test this project's version of a classification tree to make sure it operates as expected, this time on the breast cancer data-set from the sci-kit learn package as my classification tree was only programmed to work with binary classification problems.

Chapter 4: **Experiments**

(Due to changing the data-set and removing the UCI repository from this project and adding two algorithms this section is getting totally re-done)

Chapter 5: **Results**

(This section is getting totally re-done) However results for decision tree on the binary edibility accuracy scored 0.74% and random forest achieved accuracy score of 0.87% (exactly the same as the paper in the literature review's best result from features extracted from the image alone). Graphs will be produced including roc curves, the parameters for my knn algorithm haven't been optimised using cross validation yet, currently my implementation of svm is about to be fully programmed, when this is done I can create graphical results for all four algorithms. I want to plot accuracy scores with just the species classification problem with all algorithms. Also precision, recall, sensitivity, specificity and F1 scores will be plotted.

Chapter 6: **Professional issues**

6.1 Safety

This project highlights the moral dilemma of how much trust to put into an algorithm, if a poisonous mushroom is classified as edible by this projects model and someone acts on this information and ingests the mushroom there could be fatal consequences. If this algorithm could perform with higher accuracy than a foraging expert, then maybe there could be grounds to using the model instead. However, since there are features that cannot be extracted from an image, such as spore colour, smell and size, there is almost no chance that this algorithm will perform better than a human expert. Meaning it needs to be disclaimed to not act on the output of this statistical model. This model can be used as a tool to aid in classification however should be checked by a mushroom foraging expert. A real world example in a different domain is the introduction of driverless cars, and at what point is a driverless car safer than a human driving, is it when it has less accidents? less fatalities? the driverless car makes moral decisions? The current rule in America is that a driverless car can be in full control of the vehicle if there is a person at the wheel of the car ready to override the system. I think the same approach is necessary with any safety critical machine learning application, to have an override and a "expert" human making the final call. The UK government has started consultation on allowing autonomous driving, therefore it seems likely that the approach of safety critical machine learning model use will become widespread.

6.2 Licensing and use of dataset

This project uses image data from the Danish Svampe Atlas and these rules are stated on the github download page for the dataset:

"

1. You will abide by the Danish Svampe Atlas Terms of Service
2. You will use the data only for non-commercial research and educational purposes.
3. You will NOT distribute the above images.
4. The Danish Svampe Atlas makes no representations or warranties regarding the data, including but not limited to warranties of non-infringement or fitness for a particular purpose.
5. You accept full responsibility for your use of the data and shall defend and indemnify the Danish Svampe Atlas, including its employees, officers and agents, against any and all claims arising from your use of the data, including but not limited to your use of any copies of copyrighted images that you may create from the data.

" [18]
therefore during this project cannot infringe on these rules otherwise it will be breaking the terms of use of the data the Danish Svampe Atlas requires. therefore there will be no commercial applications of these models and full responsibility will be taken for any claims arising from the use of this data, in the project. There were more open-source options for

mushroom images that didn't restrict commercial applications such as the images from mushroomworld.com, however they did not have anywhere close to the number of images available from the Danish Svampe Atlas therefore the decision was made to go with this dataset of images and except the Danish Svampe Atlas Terms of Service. The Danish Svampe Atlas Terms of Service is written entirely in danish and google translate was used to understand the text, therefore it might be a good idea to get someone who can read danish to check I have understood the terms correctly.

6.3 Plagiarism

Plagiarism is the act of taking someone's ideas and passing them off as your own, therefore here I will state some examples where it could have been plagiarism if it was not made clear that the work was not mine. This project uses quite a few python modules which counts as code that has not been produced for this project, therefore every python module that is used must be referenced to give acknowledgement of the use of others code. This project's implementation of the decision tree algorithm was heavily inspired by the work of Jason Brownlee, therefore it is stated at the top of the decisiontree.py, as it is implemented with python dictionaries much like the aforementioned implementation. If this was not stated, then this would be plagiarism. It is very important to carry out correct citation because otherwise you could be carrying out accidental plagiarism is you fail to cite your sources, therefore I created a References.bib file where I've keep all references as I've gone along and can citate anything I've read in relation to the project in my report.

6.4 Management

An issue with time management of the project is at the start of the project it was not foreseen how time-consuming the data-pre-processing stage of the images was going to be, it was thought this step was going to be relatively straight forward however this did not turn out to be the case, this in turn was exaggerated by having lots of outliers in my data set that had to be discarded by eye, such as a picture of just a hand or of the spores of the mushroom under a microscope. There should have been more time allocated in the plan to data pre-processing.

6.5 Privacy and data-protection

Data protection means keeping safe the data about someone that they do not want in the public domain. In some of the images of mushrooms there are also people standing and holding the mushrooms and the person is clearly visible in frame. The person who uploads the image to the Danish Svampe Atlas gives consent for the image to be used but the person in the image may not be the same person as who is uploading and the person in the image may not be aware that an image of them is being used in this dataset without them having given consent. There are data privacy concerns related to this. Due to concerns any images containing other people displayed in them were removed from the dataset that was used for this project.

Chapter 7: **Diary**

October 9, 2020
I have been preparing the project format to push to SVN and trying to set up connection with pycharm, and doing the correct installs to allow subversion compatibility, I have also started work on my data report, where I go in-depth and study my data-set and pull key information from it and attempt to visualize it.

October 10, 2020
In my meeting with my supervisor we went over my draft project plan and I changed my project idea (to create an application to apply any machine learning algorithm you wanted to a data-set and to see the results) as my supervisor and I agreed it was a bit general. Therefore, I changed my project to be specifically about comparing algorithms for mushroom edibility classification and decided to use Jupyter Notebooks to code in and to display my findings.

October 18, 2020
I created a notebook to clean my data, encode the labels, standardize, and save the resulting data-set. I also programmed the nearest neighbour algorithm.

October 25, 2020
I programmed the K-nearest-neighbour algorithm and created a test folder for test driven development. I also converted the jupyter notebook code of nn-knn into a python module called nearestneighbor.py and created a folder, algorithms where all the python versions of the algorithms I program will reside.

October 31, 2020
I created my nearest neighbour report as a .tex file and completed 3 sections of this, the introduction, the description of k-nearest neighbour and the description of nearest-neighbour. I also downloaded a large image data-set of mushroom images , due to the fact that the data-set file was about 12GB I decided not to upload this to the repository. In my meeting with my supervisor we decided that my algorithms should be self optimizing for parameters, and the final product will be a comparison of an optimised k-nearest neighbours algorithm against an optimised decision tree. We also decided that the data-cleaning report was of little relevance and to focus more on the machine learning.

November 8, 2020
Collected resources for finishing k-nearest neighbors report, made progress on processing image data-set that is too large to be kept on the repository.

November 15, 2020
Implemented k-nearest neighbor on uci mushroom repository, achieved 100% used sci kit learn implementation in areas as I haven't programmed my cross validation function yet so put the sci-kit learn classifier temporarily in place in sections, this will be replaced.

November 21, 2020
Included sections in my knn report of who came up with the algorithms and explained different distance metrics. I programmed my own implementation of cross validation, removed sci-kit learn implementation from knn notebook.

November 29, 2020
Added decision tree section in report, programmed a confusion matrix class, programmed the scoring for the decision tree in decision-tree notebook generated all graphs and metrics

for report .

January 17, 2021
This week worked on planning out what I was going to pursue this term, finding resources about neural networks and image feature extraction.

January 24, 2021
Added a python notebook on image-feature extraction in-order to get meaningful features extracted from my image data-set

January 31, 2021
Removed background of mushroom images created file format for multi-class classification.

February 7, 2021
Created an image data set of line only mushroom images using cannyline detection and using zoning created a data set of 16 features and created a data set than is produced from colour analysis of the images.

February 14, 2021
Added both colour data and shape data into one data-set ready for the comparisons of performance of the algorithms achieving approximately 75% accuracy for both nearest neighbors and decision tree.

Chapter 8: **Bibliography**

[1] Atul Agarwal. “Support Vector Machine — Formulation and Derivation”. In: *https://towardsdatascienc vector-machine-formulation-and-derivation-b146ce89f28* (Sept. 2019).

[2] William P Arend et al. “The American College of Rheumatology 1990 criteria for the classification of Takayasu arteritis”. In: *Arthritis & Rheumatism* 33.8 (1990), pp. 1129–1134.

[3] Sidath Asiri. “Support Vector Machine — Formulation and Derivation”. In: *https://towardsdatascience learning-classifiers-a5cc4e1b0623* (June 2018).

[4] bhiksha. “Building decision trees”. In: *https://www.cs.cmu.edu/ bhiksha/courses/10-601/decisiontrees/* (2021).

[5] L Breiman. “Random Forests”. In: *Machine Learning* (2001).

[6] Huy Bui. “ROC Curve Transforms the Way We Look at a Classification Problem”. In: *https://towardsdatascience.com/a-simple-explanation-of-the-roc-curve-and-auc-64db32d75541* (2020).

[7] Nicola J Camp and Martha L Slattery. “Classification tree analysis: a statistical tool to investigate risk factor interactions with an example for colon cancer (United States)”. In: *Cancer Causes & Control* 13.9 (2002), pp. 813–823.

[8] Mohamed Amine Chikh, Meryem Saidi, and Nesma Settouti. “Diagnosis of diabetes diseases using an Artificial Immune Recognition System2 (AIRS2) with fuzzy K-nearest neighbor”. In: *Journal of medical systems* 36.5 (2012), pp. 2721–2729.

[9] T. Cover and P. Hart. “Nearest neighbor pattern classification”. In: *Information Theory, IEEE Transactions on* (1967).

[10] Judith A Falconer et al. “Predicting stroke inpatient rehabilitation outcome using a classification tree approach”. In: *Archives of Physical Medicine and Rehabilitation* 75.6 (1994), pp. 619–625.

[11] Elima Hussain, Abdul Hannan, and Kishore Kashyap. “A Zoning based Feature Extraction method for Recognition of Handwritten Assamese Characters”. In: *www.ijcst.com* 6 (Jan. 2015).

[12] Anuja Nagpal. “Decision Tree Ensembles Bagging and Boosting”. In: *towardsdatascience.com* (2020).

[13] Mohammad Ashraf Ottom. “Classification of Mushroom Fungi Using Machine Learning Techniques”. In: *International Journal of Advanced Trends in Computer Science and Engineering* 8 (Oct. 2019), pp. 2378–2385. DOI: `10.30534/ijatcse/2019/78852019`.

[14] Jitdumrong Preechasuk et al. “Image Analysis of Mushroom Types Classification by Convolution Neural Networks”. In: Dec. 2019, pp. 82–88. DOI: `10.1145/3375959.3375982`.

[15] Lior Rokach and Oded Maimon. “Top–Down Induction of Decision Trees Classifiers–A survey”. In: *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* (2005), p. 487.

[16] H Sabrol and K Satish. “Tomato plant disease classification in digital images using classification tree”. In: *2016 International Conference on Communication and Signal Processing (ICCSP)*. IEEE. 2016, pp. 1242–1246.

[17] Manish Sarkar and Tze-Yun Leong. “Application of K-nearest neighbors algorithm on breast cancer diagnosis problem.” In: *Proceedings of the AMIA Symposium*. American Medical Informatics Association. 2000, p. 759.

[18] Brigit Schroeder. “fgvcx$_f$ungi$_c$omp”. In: *https://github.com/visipedia/fgvcx$_f$ungi$_c$omp* (2021).

[19] Hendrana Tjahjadi and Kalamullah Ramli. "Noninvasive Blood Pressure Classification Based on Photoplethysmography Using K-Nearest Neighbors Algorithm: A Feasibility Study". In: *Information* 11.2 (2020), p. 93.

[20] Robert Tibshirani Trevor Hastie and Jerome Friedman. *The Elements of Statistical Learning.* Second edition. Springer, New York, 2009.

[21] Vispedia. "Danish Svampe Atlas". In: *https://github.com/visipedia/fgvcx_fungi_comp* (2018).

[22] Eric W Weisstein. "Vector Norm". In: *mathworld.wolfram.com* (2020).

[23] Hall Mark Witten Ian Frank Eibe. "Data Mining". In: *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* (2011), pp. 101–103.

[24] Xueyan Wu, Jiquan Yang, and Shuihua Wang. "Tea category identification based on optimal wavelet entropy and weighted k-Nearest Neighbors algorithm". In: *Multimedia Tools and Applications* 77.3 (2018), pp. 3745–3759.

[25] Qi Yu et al. "Optimal pruned K-nearest neighbors: OP-KNN application to financial modeling". In: *2008 Eighth International Conference on Hybrid Intelligent Systems.* IEEE. 2008, pp. 764–769.

[26] Junni L Zhang and Wolfgang K Härdle. "The Bayesian additive classification tree applied to credit risk modelling". In: *Computational Statistics & Data Analysis* 54.5 (2010), pp. 1197–1205.