**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**SCHOOL OF INFORMATION TECHNOLOGY AND COMMUNICATION**

---------------o0o---------------

Logo

Description automatically generated

**Project Report: Mushroom classification**

**Course: Machine learning and data mining**

Lecturer: Ph.D. Nguyen Nhat Quang

| Group |  |
| --- | --- |
| Đinh Thế Kiệt | 20194783 |
| Vũ Hoàng Nam | 20194809 |

[**1. Overview**](#_b531trn69gz6) **3**

[**2. Dataset**](#_7b8ikahy5vis) **3**

[1. Context](#_o6bmo9bs6nks) 3

[2. Content](#_y9qhpxq9j55) 3

[**3. Exploratory Data Analysis and Preprocessing**](#_hb0761sep1wh) **3**

[Check for missing values and dataset details](#_9zih21k87vh0) 3

[Data visualisation](#_wea4e9eia6om) 4

[Covariance matrix](#_8wof2tlo7wlv) 5

[Preprocessing:](#_nhf7o6aaaaa5) 5

[**4. Algorithm**](#_f6wvlp4muqzs) **6**

[Logistic regression](#_gt5oe7qe7wny) 6

[Support vector machine](#_5dlve1capiyf) 8

[Random forest](#_ehaknscfq3d8) 9

[**5. Evaluation**](#_rgl816biiq44) **11**

[Metrics](#_4dfd5q3sz9ix) 11

[Performance](#_a076hstmhh4m) 12

[**6. Issues**](#_nd6u4mxmbgq3) **13**

[**7. Discussions**](#_6xcvafk5e8uz) **14**

# 

# 

# 1. Overview

In this project, we will examine the data and build different **machine learning models** that will detect if the mushroom is **edible or poisonous** by its specifications like cap shape, cap-colour, gill-colour, etc. using different classifiers.

In this project we have used the following classifiers to make the prediction:

1. Logistic Regression
2. Support Vector Machine
3. Random Forest

# 2. Dataset

## 1. Context

Although this dataset was originally contributed to the UCI Machine Learning repository nearly 30 years ago, mushroom hunting (otherwise known as "shrooming") is enjoying new peaks in popularity. Learn which features spell certain death and which are most palatable in this dataset of mushroom characteristics.

## 2. Content

This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family Mushroom drawn from The Audubon Society Field Guide to North American Mushrooms (1981). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like "leaflets three, let it be'' for Poisonous Oak and Ivy.

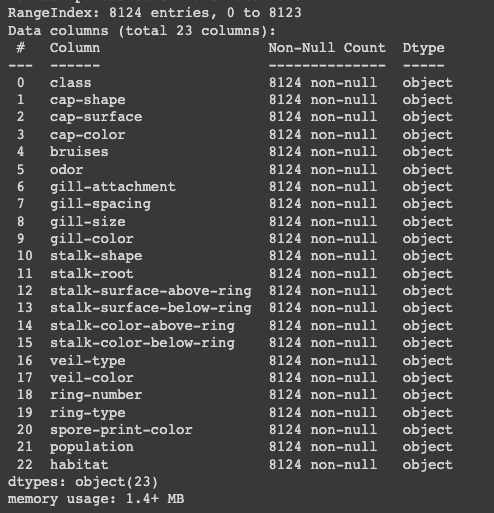
The dataset is released in Kaggle, https://www.kaggle.com/datasets/uciml/mushroom-classification.

# 3. Exploratory Data Analysis and Preprocessing

## Check for missing values and dataset details

1.1. Drop features with too many missing values:

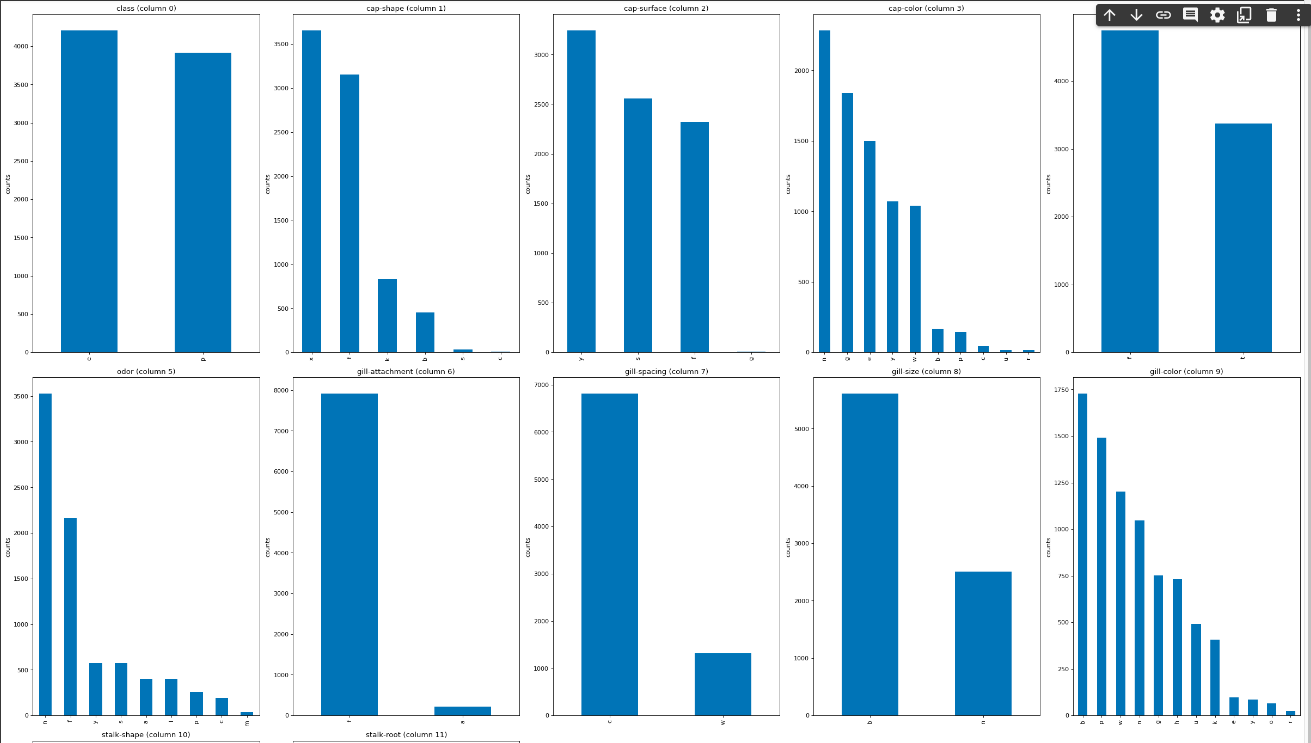
One of the first tasks in any dataset is to find if there are missing values. As this dataset has more features than we’d like, we’re going to drop features that have too many missing values. Let’s see how many missing values each feature has.



It seems to have no null value in the dataset. There is no need to drop any features with null values.

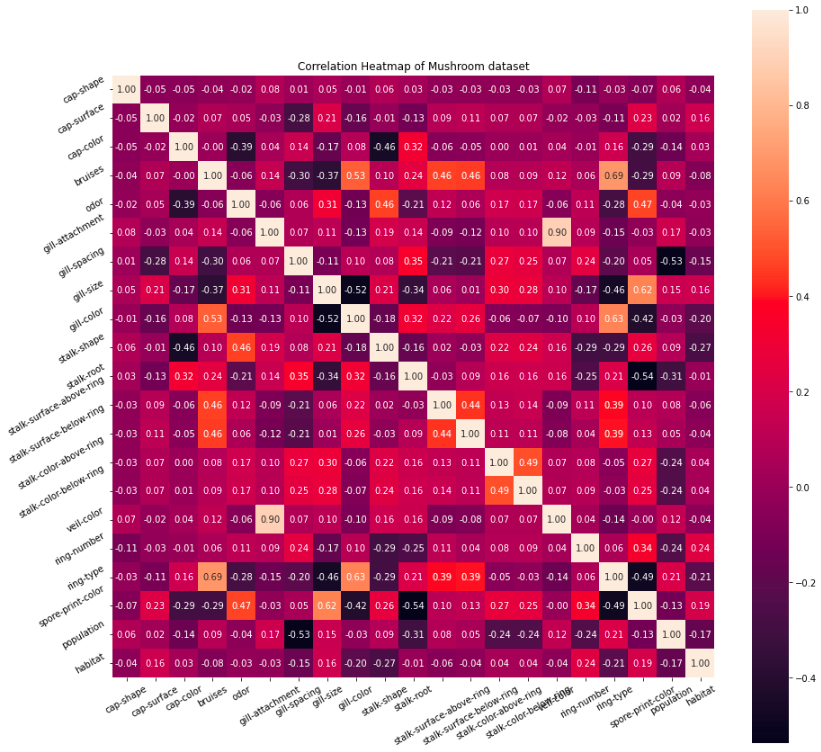
## Data visualisation

Data visualisation is the practice of translating information into a visual context, such as a map or graph, to make data easier for the human brain to understand and pull insights from. The main goal of data visualisation is to find a pattern from data.



## Covariance matrix

A covariance matrix with all non-zero elements tells us that all the individual random variables are interrelated. This means that the variables are not only directly correlated, but also correlated via other variables indirectly.

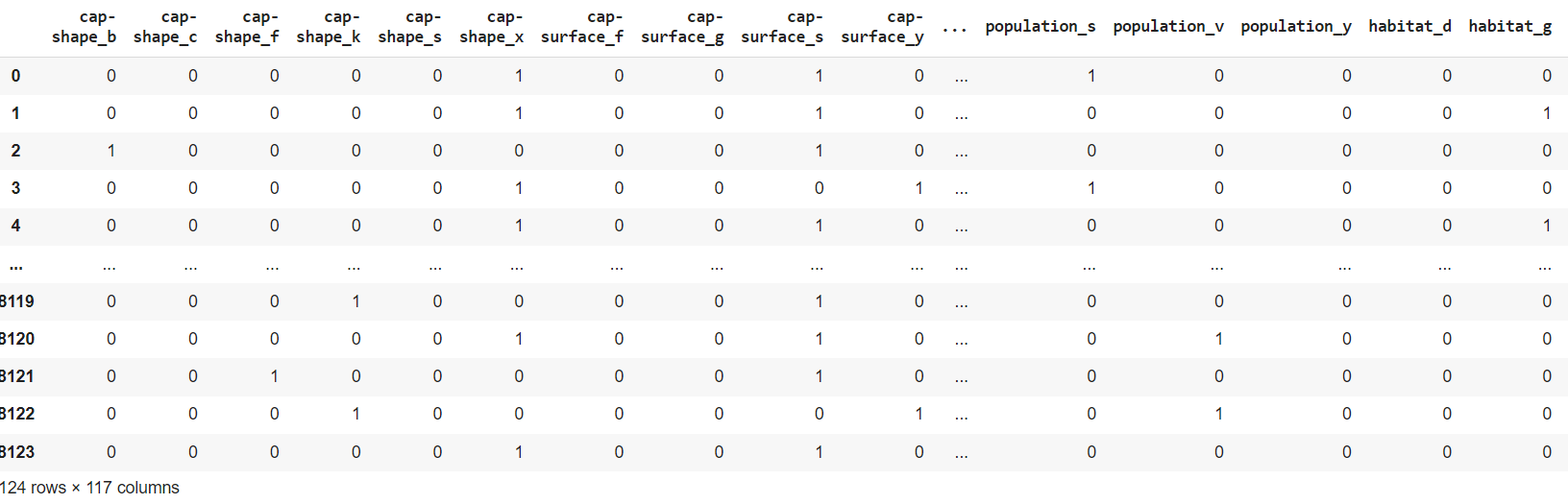


## Preprocessing:

3.1 Convert categorical data to numerical data:

All the features of this dataset is represented as categorical datatype. This results in the difficulty in calculating and building machine learning model. Each feature is composed of a limited number of values (labels), therefore, each values can be converted into a number as an id for that label.

* For independent features, we will convert categorical data into numerical data by using DataFrame.get\_dummies() method. This method will use one-hot encoder which encode categorical features as a one-hot numerical array. The input to this transformer should be an array-like of integers or strings, denoting the values taken on by categorical (discrete) features. The features are encoded using a one-hot (aka ‘one-of-K’ or ‘dummy’) encoding scheme. This creates a binary column for each category and returns a sparse matrix or dense array.



* For dependent features, we will convert categorical features into numerical feature by using LabelEncoder()tool which encodes target labels with values between 0 and n\_classes-1. In this case, a dependent feature is the target column.



3.2 Splitting training set and test set:

* We will use train\_test\_split of Sklearn-Model Selection to split train and test set
* By using the hyperparameter test\_size, we can split the test via the overall dataset, which has the same size as specified. The random\_state will use the random seed on Python to shuffle the dataset
* In this dataset, test\_size = 0.30, random\_state = 0 will be used to split the dataset into training set and test set.

# 4. Algorithm

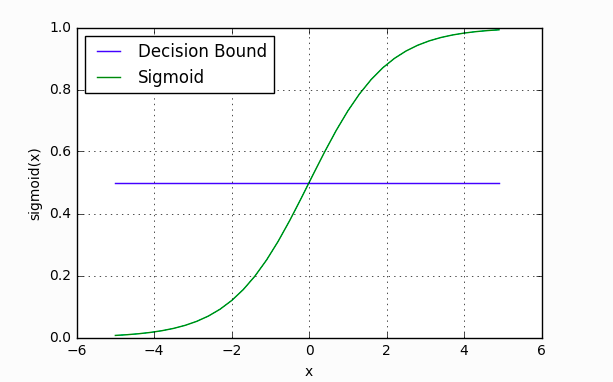
## Logistic regression

1.1. Decision boundary

Our current prediction function(sigmoid function) returns a probability score between 0 and 1. In order to map this to a discrete class (true/false, cat/dog), we select a threshold value or tipping point above which we will classify values into class 1 and below which we classify values into class 2.

𝑝≥0.5,𝑐𝑙𝑎𝑠𝑠=1

𝑝<0.5,𝑐𝑙𝑎𝑠𝑠=0



1.2. Making predictions

* Using our knowledge of sigmoid functions and decision boundaries, we can now write a prediction function. A prediction function in logistic regression returns the probability of our observation being positive, True, or “Yes”.
* Let’s use the same multiple linear regression equation from our linear regression tutorial.

𝑧 = 𝑊0 + 𝑊1\*𝑆𝑡𝑢𝑑𝑖𝑒𝑑 + 𝑊2\*𝑆𝑙𝑒𝑝𝑡

* This time however we will transform the output using the sigmoid function to return a probability value between 0 and 1.

𝑃(𝑐𝑙𝑎𝑠𝑠=1) =

* If the model returns .4 it believes there is only a 40% chance of passing. If our decision boundary was .5, we would categorise this observation as “Fail.””

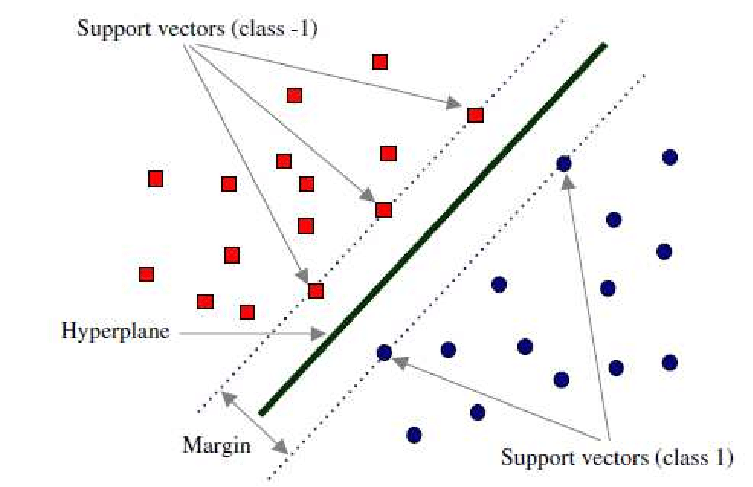
1.3. Gradient descent, update weights

* To minimise our cost, we use Gradient Descent.
* C’ is the derivative of cost with respect to weights
* 𝑦 is the actual class label (0 or 1)
* 𝑠(𝑧) is your model’s prediction
* 𝑥 is your feature or feature vector.
* Update weights: weights is equal to weights minus gradient multiplied by learning rate
* The iteration ends until a certain number of epochs or there is no better performance among epochs

## Support vector machine

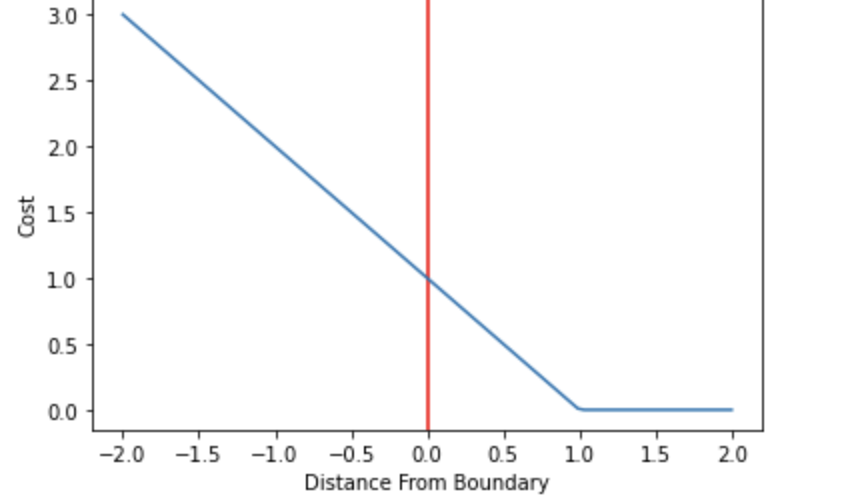
2.1. Hyperplanes, support vectors and margin

* Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes.
* Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximise the margin of the classifier.



2.2. Cost Function and Gradient Updates

1. In the SVM algorithm, we are looking to maximise the margin between the data points and the hyperplane. The loss function that helps maximise the margin is hinge loss.



2. Now that we have the loss function, we take partial derivatives with respect to the weights to find the gradients. Using the gradients, we can update our weights. When there is no misclassification, i.e our model correctly predicts the class of our data point, we only have to update the gradient from the regularisation parameter.

## Random forest

A random forest is a supervised machine learning algorithm that is constructed from decision tree algorithms. It is used to solve regression and classification problems. It utilises ensemble learning which is a technique that combines many classifiers to provide solutions to complex problems.

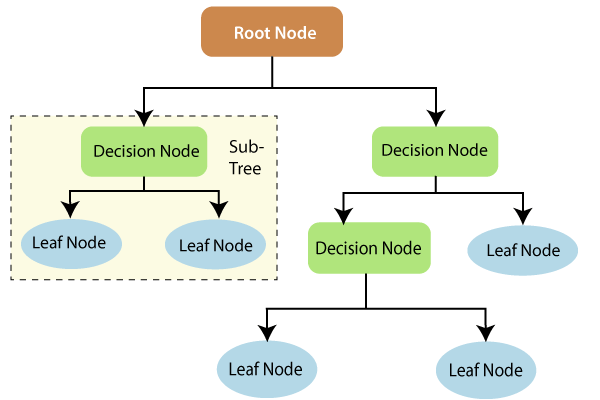
The random forest algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees.

3.1 Decision tree:

Decision trees are the building blocks of a random forest algorithm. A decision tree is a decision support technique that forms a tree-like structure.

A decision tree consists of three components: decision tree, leaf nodes and a root node. A decision tree algorithm divides a training dataset into branches, which further segregate into other subbranches. This sequence continues until a leaf node is attained. The leaf node cannot be segregated further.

The nodes in the decision tree represent attributes that are used for predicting the outcome. Decision nodes provide a link to the leaves. The following diagram shows the three types of node in a decision tree.



Entropy is a metric for calculating uncertainty. Information gain is a measure of how uncertainty in the target variable is reduced, given a set of independent variables.

The information gain concept involves using independent variables (features) to gain information about a target variable (class). The entropy of the target variable (Y) and the conditional entropy of Y (given X) are used to estimate the information gain. In this case, the conditional entropy is subtracted from the entropy of Y.

Information gain is used in the training of decision trees. It helps in reducing uncertainty in decision trees. A high information gain means that a high degree of uncertainty (information entropy) has been removed.

Entropy and information gain are important in splitting branches, which is an important activity in the construction of decision trees.

3.2 Applying decision trees in random forest:

The random forest employs the bagging method to generate the required the required prediction.

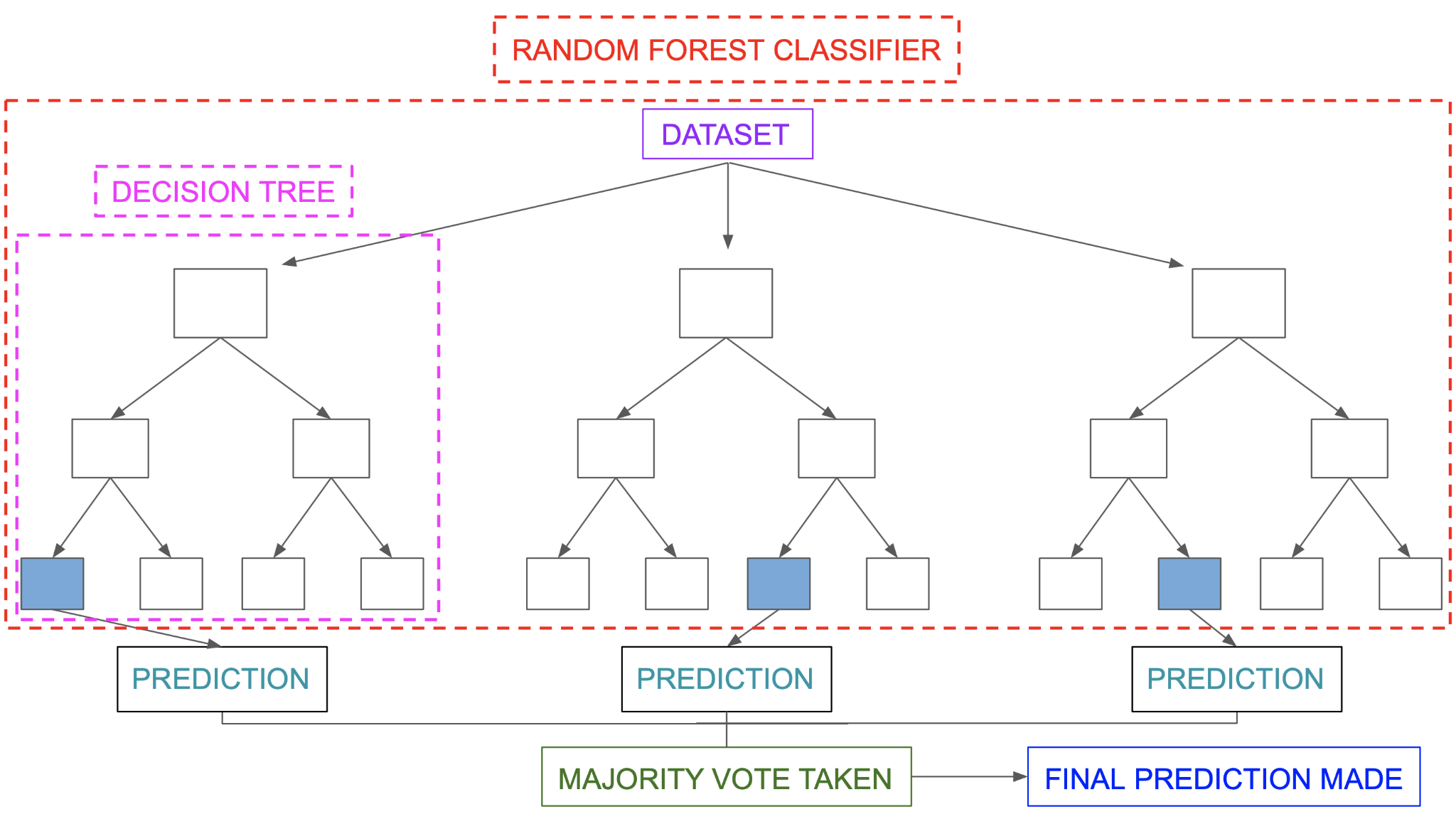
Bagging involves using different samples of data (training data) rather than just one sample. A training dataset comprises observations and features that are used for making predictions. The decision trees produce different outputs, depending on the training data fed to the random forest algorithm. These outputs will be ranked and the highest will be selected as the final output.

Instead of having a single decision tree, the random forest will have many decision trees. The outcome chosen by most decision trees will be the final choice.

3.3 Classification in random forest

Classification in random forest employs an ensemble methodology to attain the output. The training data is fed to train various decision trees. This dataset consists of observations and features that will be selected randomly during splitting of nodes.

A random forest relies on various decision trees. Every decision tree consists of decision nodes, leaf nodes and a root node. The leaf node of each tree is the final output produced by that specific decision tree. The selection of the final output follows the majority-voting system. In this case, the output chosen by the majority of the decision trees becomes the final output of the random forest system



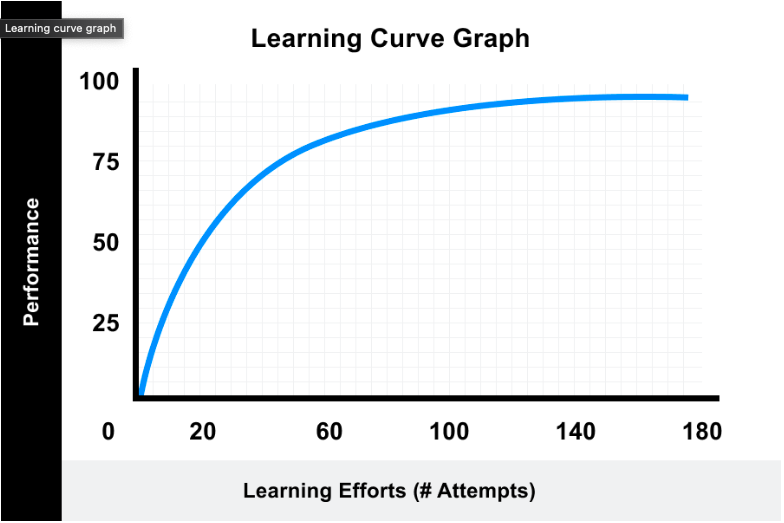
# 5. Evaluation

## Metrics

1.1 Accuracy

Model accuracy is the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data. The better a model can generalise to ‘unseen’ data, the better predictions and insights it can produce, which in turn deliver more business value.

Learning curves: a learning curve is a correlation between a learner's performance on a task and the number of attempts or time required to complete the task; this can be represented as a direct proportion on a graph.



1.2 Confusion matrix

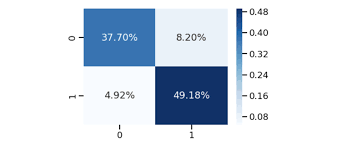
When performing classification predictions, there's four types of outcomes that could occur.

* True positives are when you predict an observation belongs to a class and it actually does belong to that class.
* True negatives are when you predict an observation does not belong to a class and it actually does not belong to that class.
* False positives occur when you predict an observation belongs to a class when in reality it does not.
* False negatives occur when you predict an observation does not belong to a class when in fact it does.

These four outcomes are often plotted on a confusion matrix.

Confusion matrix heatmap

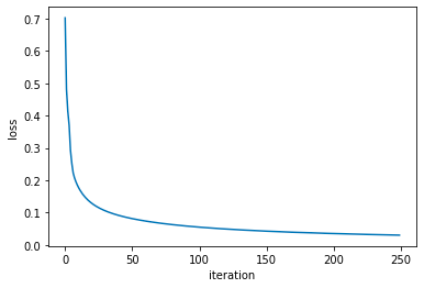
A confusion matrix heatmap is a graphical representation where 4 above values of a matrix are represented as colours. A heatmap is very useful in visualising the concentration of values between two dimensions of a matrix. This helps in finding patterns and gives a perspective of depth.



## Performance

2.1 Logistic regression:

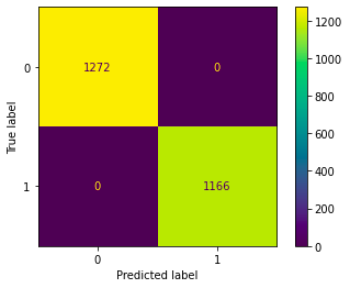
In training phase, the loss witnesses a fall from 0.7 to 0.08 in the 20 first iteration which is followed by a slight decrease until the loss is 99,73%



In test phase, the accuracy is approximately the accuracy in training phase which is 99,79%.

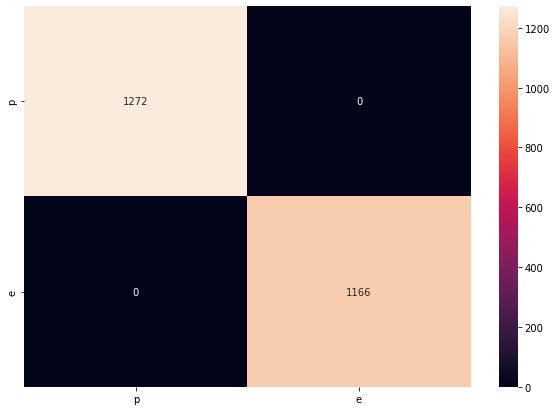
2.2 Support vector machine:

Heatmap confusion matrix shows that the true positives and false negatives are high while true negatives and false positives is approximately 0. It means our model seems to predict correctly.



2.3 Random forest:

For random forest, the confusion matrix shows that the true positives and false negatives are high which implies that the model is perfectly predicted with the test set.



# 6. Issue

*Problem: the accuracy in both training phase and test phase is approximately 100%. It seems to me that the problems might raise in some steps.*

Scenario 1: building models and evaluating algorithms might have some bugs.

* Solutions: we utilise sklearn library to build models and evaluate the system. There are no disparities between with and without library
* Conclusion: Our model and evaluating algorithm at first are correct.

Scenario 2: The dataset contains duplicated rows which leads the model studying from one case to easily predict the others.

* Solutions: we carefully observe the dataset and clean duplicated values using drop\_duplicates and there is no dropped rows.



* Conclusion: duplicated rows didn’t raise the issue.

Scenario 3: The dataset is too easy which any machine learning model studies perfectly.

* Verifying the scenario: we changed the random state to have different ordered dataset and the accuracies in three machine learning models (logistic regression, svm, random forest) changed slightly



**Final conclusions: The dataset is too easy so both easy and complicated models perform well.**

# 7. Discussions

1. Dimensions reduction/feature selection

* Problem: The dataset can contain a huge number of features, some of which are not even required. Such redundant information makes modelling complicated.
* Solutions: Dimension reduction should be used for reducing the cost and complexity. Moreover, it can gain more efficiency.

1. Evaluate the importance ranking of all features: For each model, the contribution of each feature to classify an object as a label is not the same. Therefore, the higher the contribution is, the more relevant that feature is.
2. After ranking the importance of each feature, we will gradually eliminate each feature, which has low relevance score, out of the dataset until the accuracy drops significantly.
3. Small and easy dataset.

* Problem: The dataset shows no null value and contains only 8214 rows and 23 columns which is small. So, while building the model in three machine learning models perform well which is approximately 100% accuracy.
* Solutions: Choosing a more complicated dataset and observing carefully the difference among algorithms or different preprocessing methods to compare the advantages and disadvantages among them.