Mushrooms dataset.

This dataset describes mushrooms in terms of their physical characteristics. They are classified into: poisonous or edible.this dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. 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.

Features

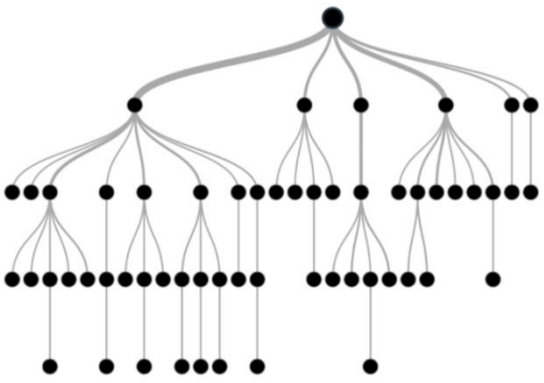
class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habit

Not all mushrooms are edible. While some of them have medicinal properties to treat cancer, other types may contain viruses that transmit infectious diseases.

This report was prepared to study the behavioral characteristics of mushrooms and whether they are edible or not.

The importance of this data comes in helping and determining whether mushrooms are edible or not. Some types of mushrooms are considered very poisonous and may lead to human death quickly. This work helps well in a preliminary identification of the type of mushroom.

ML models



The **decision tree** is a powerful tool for solving classification and prediction problems and is also considered one of the most user-readable models.
Root Nodes – It is the node present at the beginning of a decision tree from this node the population starts dividing according to various features.

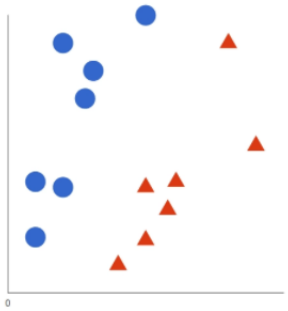
Decision Nodes – the nodes we get after splitting the root nodes are called Decision Node

Leaf Nodes – the nodes where further splitting is not possible are called leaf nodes or terminal nodes

Decision trees are upside down which means the root is at the top and then this root is split into various several nodes. Decision trees are nothing but a bunch of if-else statements in layman terms. It checks if the condition is true and if it is then it goes to the next node attached to that decision

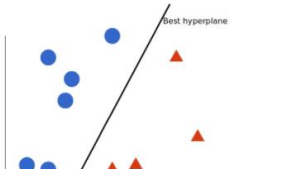
A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize new text.

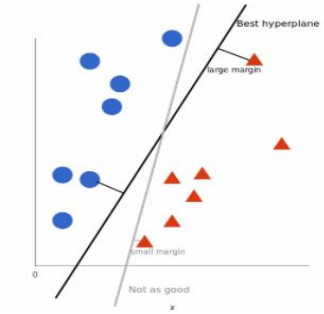
Compared to newer algorithms like neural networks, they have two main advantages: higher speed and better performance with a limited number of samples (in the thousands). This makes the algorithm very suitable for text classification problems, where it's common to have access to a dataset of at most a couple of thousands of tagged samples.



Let's imagine we have two tags: *red* and *blue*, and our data has two features: *x* and *y*. We want a classifier that, given a pair of *(x,y)* coordinates, outputs if it's either *red* or *blue*. We plot our already labeled training data on a plane.

A support vector machine takes these data points and outputs the hyperplane (which in two dimensions it's simply a line) that best separates the tags. This line is the **decision boundary**: anything that falls to one side of it we will classify as *blue*, and anything that falls to the other as *red*. But, what exactly is *the best* hyperplane? For SVM, it's the one that maximizes the margins from both tags. In other words: the hyperplane (remember it's a line in this case) whose distance to the nearest element of each tag is the largest.





Cleaning steps.

1. Encoding dataset
2. Drob any non value in dataset
3. Scalar
4. Pca

1.

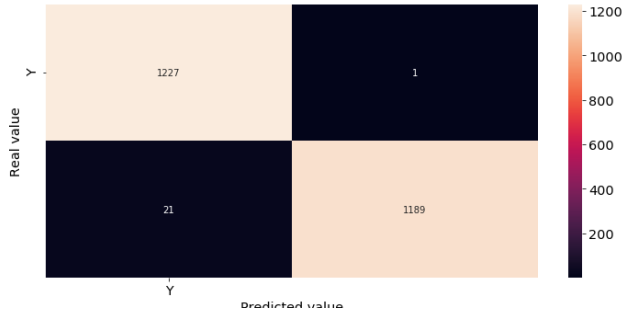
Building the predictive model

1. Importing sklearn library
2. Import the model from sklearn
3. train the model on train data
4. predict the result of test data
5. plot accuracy

Result

Decision tree :

```
DecisionTree = ExtraTreesClassifier(n_estimators = 4)
train_and_test(DecisionTree, X_train_pca, X_test_pca, y_train, y_test)
```



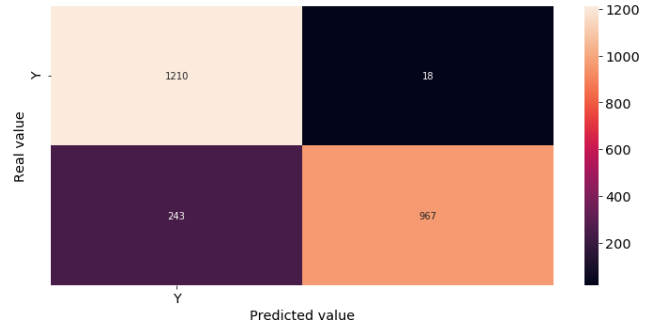

```
accuracy  score: 0.9909762100082035 %
auc  score: 0.9976579535359518
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1228
           1       1.00      0.98      0.99      1210

    accuracy                           0.99      2438
```

SVM:

```
svm = SVC(C=0.01,probability=True)
train_and_test(svm, X_train_pca, X_test_pca, y_train, y_test)
```




```
accuracy  score: 0.8929450369155045 %
auc  score: 0.9721585861576978
              precision    recall  f1-score   support

           0       0.83      0.99      0.90      1228
           1       0.98      0.80      0.88      1210

    accuracy                           0.89      2438
```

From the previous pictures, we can notice that the **decision tree** gives better results than the **svm** algorithm. The reason for this is that I believe that the **decision tree** algorithm has chosen a certain future that helps more in the accuracy of the decision, while the classification mechanism in **svm** has taken all the future with equal importance.