First we load the libraries we will need for reading and formatting the input data (pandas for the dataframe structure, train_test_split from sklearn to split the data into test and training sets). Then we load SVC (a kind of SVM, Support Vector Machine).

```
[2] import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.svm import SVC
```

Then we load the text file into google drive so Google colab can access the data.

The loaded data is parsed like a csv file into a pandas dataframe structure

```
[3] df = pd.read_csv('Mushrooms.01.txt', sep="  ", encoding='latin-1')  #invisible separator is some kind of utf arrow in the source file
    df
```

	Id	Odorant	Anneaux	Chapeau bombé	Pied large	Tâches	Comestible
0	champignon 1	1	0	1	1	0	1
1	champignon 2	1	0	1	1	1	1
2	champignon 3	0	0	1	1	1	1
3	champignon 4	1	0	0	1	1	0
4	champignon 5	0	0	1	1	0	1
5	champignon 6	1	0	1	1	0	1
6	champignon 7	0	0	1	1	0	1
7	champignon 8	1	1	0	1	0	1
8	champignon 9	0	0	1	1	0	0
9	champignon 10	1	1	1	1	0	1
10	champignon 11	0	0	1	0	1	1
11	champignon 12	0	1	1	1	0	0
12	champignon 13	1	1	1	1	0	0
13	champignon 14	1	0	1	1	0	1
14	champignon 15	0	0	0	1	1	1
15	champignon 16	0	0	1	1	0	1
16	champignon 17	0	1	0	0	1	0
17	champignon 18	1	0	1	1	1	1
18	champignon 19	1	0	1	1	0	1
19	champignon 20	0	0	1	1	0	1
20	champignon 21	1	1	1	0	1	1
21	champignon 22	1	0	1	1	1	1
22	champignon 23	0	0	1	1	1	1
23	champignon 24	0	0	1	0	0	0

We then isolate the label "Comestible" as the desired predictions "y" :

```
[4] y = df.Comestible #these are the labels we want to predict
    y
```

```
0     1
1     1
2     1
3     0
4     1
5     1
6     1
7     1
8     0
9     1
10    1
11    0
12    0
13    1
14    1
15    1
16    0
17    1
18    1
19    1
20    1
21    1
22    1
23    0
Name: Comestible, dtype: int64
```

Here, we drop useless input columns from our dataset x.

```
[5] x = df.drop("Comestible", axis=1) #this is the input data, so we remove the "truth" column (now stored in "y" variable)
    x = x.drop("Id", axis=1) #the id is useless as we already have ids and it is not a mushroom caracteristic
    x
```

	Odorant	Anneaux	Chapeau bombé	Pied large	Tâches
0	1	0	1	1	0
1	1	0	1	1	1

2	0	0	1	1	1
3	1	0	0	1	1
4	0	0	1	1	0
5	1	0	1	1	0
6	0	0	1	1	0
7	1	1	0	1	0
8	0	0	1	1	0
9	1	1	1	1	0
10	0	0	1	0	1
11	0	1	1	1	0
12	1	1	1	1	0
13	1	0	1	1	0
14	0	0	0	1	1
15	0	0	1	1	0
16	0	1	0	0	1
17	1	0	1	1	1
18	1	0	1	1	0
19	0	0	1	1	0
20	1	1	1	0	1
21	1	0	1	1	1
22	0	0	1	1	1
23	0	0	1	0	0

Our data has been cleaned up and separated into x (inputs) and y (desired outputs). We must split it for cross-validation.

```
[6] X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3) #separate the data into train (70%) and test (30%) variables
```

As stated in the libraries import section, we need a SVC model, I chose "linear" as it is the recommended option.

```
[7] model = SVC(kernel='linear') #load a model
```

We train the model on the training set with the x inputs and desired y predictions

```
[8] model.fit(X_train, y_train) #train the model
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

First validation : we display a vector containing the predicted classes of the test data. For each "x" input, we ouput a "y" value.

```
[9] predictions = model.predict(X_test) #show predictions (0 = poisonous, 1 = comestible) for test set
    print(predictions)
```

```
[1 1 1 0 1 1 1 1]
```

Here we use the score function to compare predictions to the truth and output a percentage of successful predictions.

```
[10] percentage = model.score(X_test, y_test) #show a percentage of accuracy
     percentage
```

```
0.875
```

Then we display a confusion matrix (notice how most predictions fall on the diagonal) and display the size of the test set, as well as the previously observed accuracy to have a nice summary.

```
[11] from sklearn.metrics import confusion_matrix
     res = confusion_matrix(y_test, predictions) #create a confusion matrix to display TN/TP/FP/FN
     print("Confusion Matrix : ")
     print(res)
     print(f"Test Set : {len(X_test)}")
     print(f"Accuracy = {percentage*100} %")
```

```
Confusion Matrix :
[[0 0]
 [1 7]]
Test Set : 8
Accuracy = 87.5 %
```

There is an imbalance between the classes we are trying to predict (only 6/24 poisonous mushrooms) and the dataset is quite small, so we can expect a high variance and maybe some bad surprises if we were to add poisonous mushrooms with features that match the previously edible ones.