G1

RAPPORT

FRUIT RECOGNITION

Florent MARQUES
Thomas WILHEM

Table des matières

Introduction

Le but de ce projet est de créer une intelligence artificielle permettant de classer des fruits selon une image en entrée. Nous avons un jeu de données de plus de 16000 images de 100 pixels par 100 pixels en couleur.

Identifier le type du problème

Comme dit plus haut, nous avons un jeu de données en entrée de plus de 16000 images de fruits différents et dans différentes positions. Nous souhaitons obtenir la classe de chacune de ses images. La sortie représente une valeur entière que l'on pourra par la suite assimiler à un fruit.

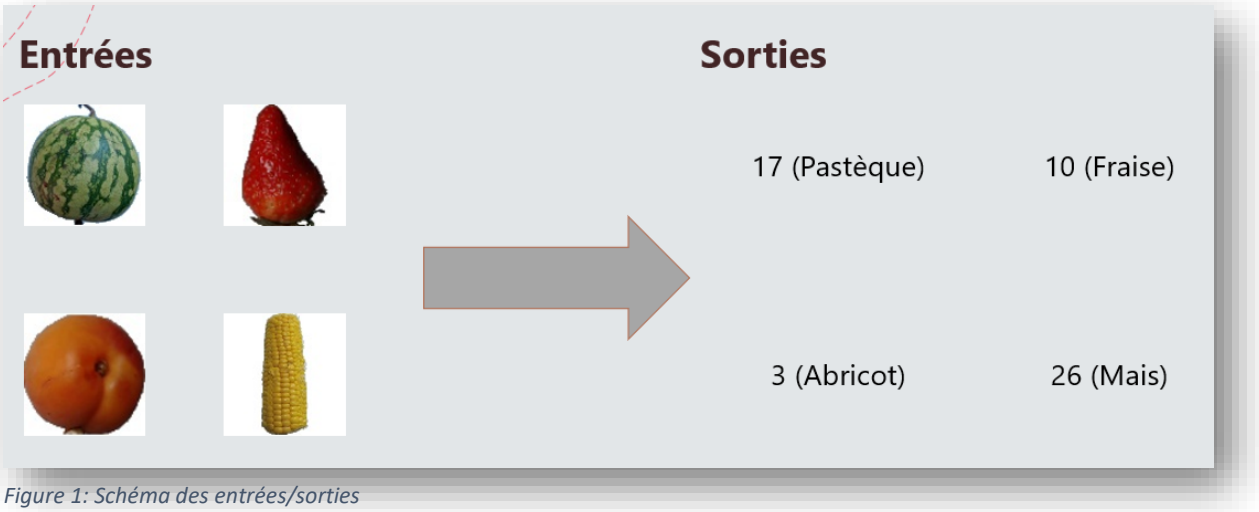


Figure 1: Schéma des entrées/sorties

Ce problème s'apparente donc à une classification car nous voulons mettre dans des boites chacune des données d'entrée. Ce problème nous mène donc à classer des fruits selon leur espèce comme le montre l'image qui suit. Dans l'exemple ci-dessous, nous avons plusieurs animaux différents en entrée et le but est de les classer selon un critère spécifique, ici l'espèce. Dans notre cas, le problème est le même et le critère de classification est la catégorie du fruit.

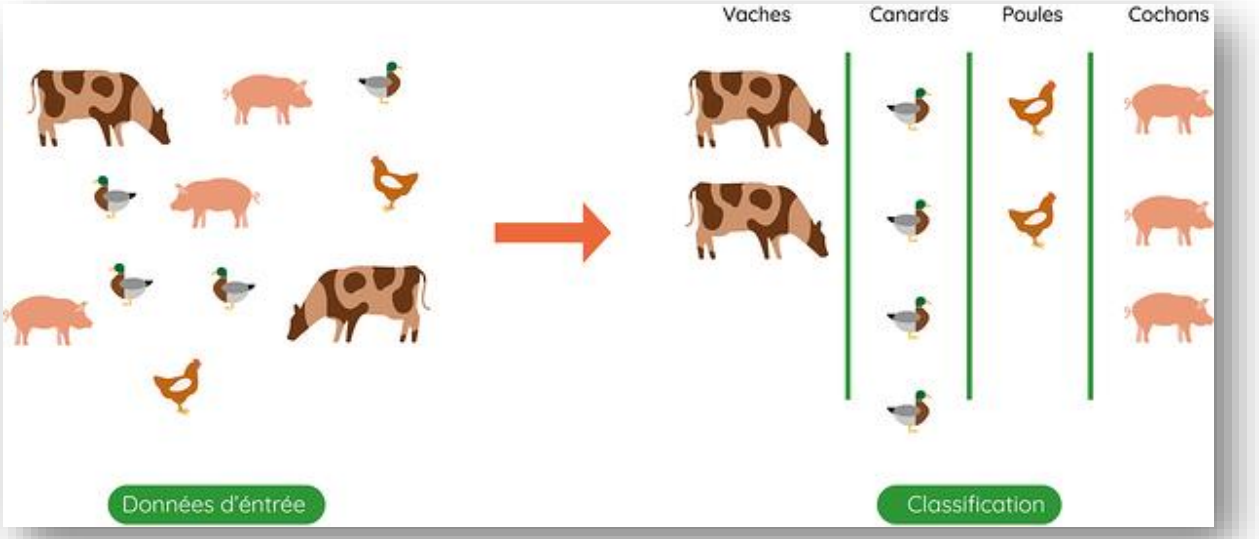


Figure 2: Exemple de la classification

Comprendre les données

Quantité des données

Nous avons procédé au chargement des images dans un DataFrame grâce au module python « Pandas ». Comme nous ne pouvions pas stocker une image brute dans le DataFrame, il a fallu redimensionner l'image en une seule ligne de données.

```
tabImg=pd.DataFrame()
categories = ['Apple Braeburn', 'Apple Granny Smith', 'Apricot', 'Avocado', 'Banana', 'Blueberry', 'Cactus fruit', 'Cantaloupe', 'Cherry', 'Clementine'
label = []
root_dir=os.getcwd()+"\\data\\train\\"

for filename in glob.iglob(root_dir+'**\\*.jpg',recursive=True):
    img=im.imread(filename)
    img=img.reshape(img.shape[0]*img.shape[1]*img.shape[2])

    fruit_name=filename.split('\\')[-2]

    label.append(categories.index(fruit_name))
    tabImg=tabImg.append(pd.Series(img),ignore_index=True)
tabImg['label']=label
```

Conversion des images en lignes

Figure 4: Extrait de code permettant le chargement des images

```
tabImg.head(5)
```

	0	1	2	3	4	5	6	7	8	9	...	29991	29992	29993	29994	29995	29996	29997	29998	29999
0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	...	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0
1	255.0	255.0	246.0	253.0	255.0	247.0	253.0	255.0	249.0	251.0	...	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0
2	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	...	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0
3	255.0	255.0	251.0	255.0	255.0	253.0	255.0	254.0	252.0	255.0	...	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0
4	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	...	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0	255.0

Figure 3: Représentation du DataFrame après le chargement des images

Qualité des données

Figure 5: Représentation d'un fruit sous différents angles

Les différentes catégories de fruits sont représentées dans différentes positions ce qui permet de complexifier la reconnaissance pour l'intelligence artificielle.

Toutes les données du jeu de données sont exploitables et de qualité suffisance du fait qu'elles soient facilement reconnaissable et facilement identifiable par un humain.

Comment évaluer le modèle ?

Pour pouvoir tirer des conclusions sur nos différents tests et l'efficacité de notre intelligence artificielle, il faut que l'on utilise différents indicateurs. Nous en avons retenu 2.

Accuracy Score

```
accuracy_score(ytest,ypredit)
```

```
0.992406264831514
```

Le premier indicateur est l'accuracy score (en français « score de précision »). Cet indicateur va nous permettre d'observer le taux de réussite du test sans savoir où l'intelligence artificielle s'est trompé ni de savoir pourquoi.

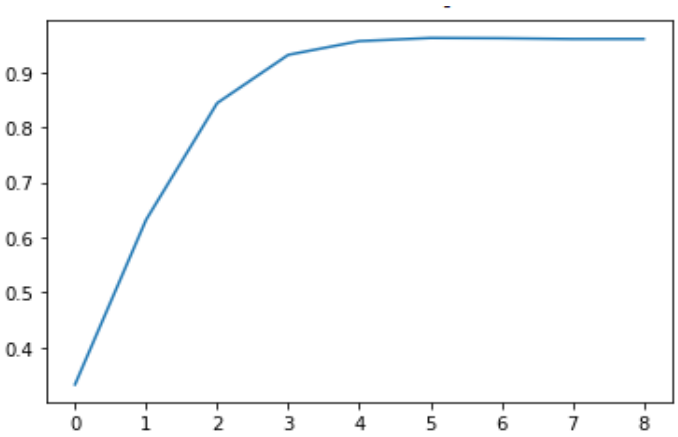


Figure 6: Courbe d'accuracy score

Matrice de confusion et HeatMap

En complément du premier indicateur, nous allons utiliser une HeatMap (en français « carte de chaleur ») qui est une extension de la matrice de confusion. Le principe d'une matrice de confusion permet de déterminer de montrer les résultats pour chaque classe et de ce fait nous pouvons voir sur quelle classe l'intelligence artificielle s'est trompée.

Figure 7: HeatMap d'exemple

Par exemple, sur la heatmap ci-dessus, nous pouvons constater que l'intelligence artificielle s'est beaucoup trompée en confondant quelques classes avec la classe 0. Par exemple, comme le montre le cercle vert, du fait de la couleur de la case, on peut constater que l'intelligence artificielle s'est fortement trompée en prédisant la classe 16 alors que c'était une classe 0, elle a prédit environ 120 fois la classe 16 alors que la classe réelle était la classe 0.

Séparation du jeu de données

Pour réaliser nos tests, nous avons diviser notre jeu de données en 2 : 75% pour l'entrainement de l'intelligence artificielle(train) et 25% pour le test de l'efficacité de l'intelligence artificielle.

Quel modèle choisir ?

Les modèles

Nous avons testé différents modèles : le réseau de neurones simple, le support vector machine, le plus proche voisin, l'arbre de décision et le gaussien.

Nous avons retenu trois modèles où nous avons observé de bons résultats : le réseau de neurones, le support vector machine et le plus proche voisin.

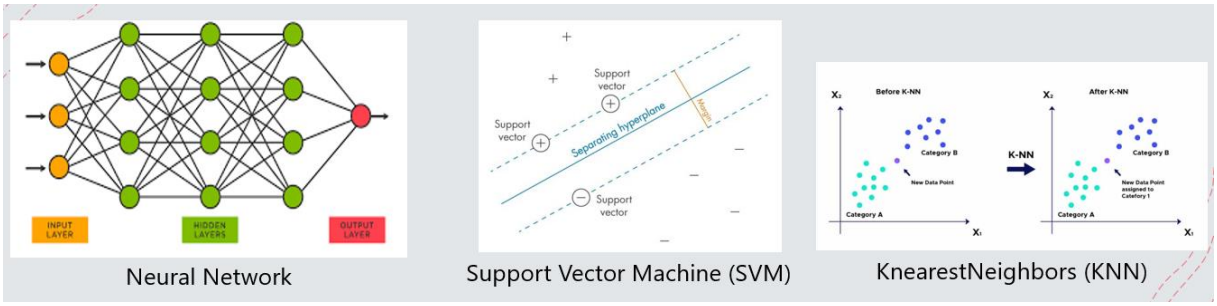


Figure 8: Les différents modèles retenus

Modèle	KNN	SVM	Neural Network
Paramètre optimal	n_neighbors=1	Kernel='poly'	Solver= 'lbfgs', hidden_layer_sizes= (150,150,50)

Trouver les bons paramètres

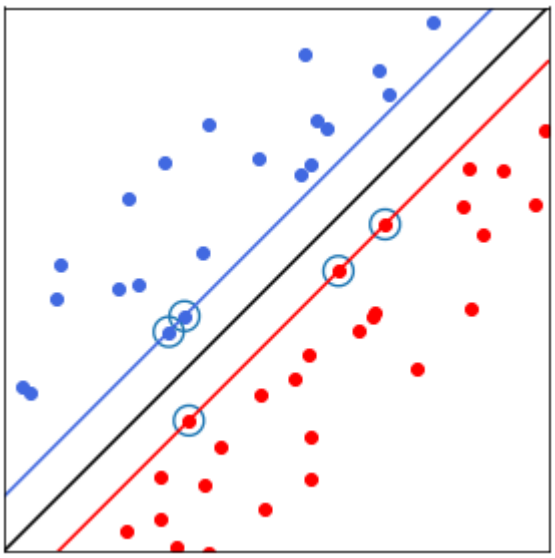
Afin d'obtenir de bons résultats, nous avons dû modifier certains paramètres sur les différents modèles car les paramètres par défaut ne correspondaient pas à notre problème.

Nous avons dû effectuer ces changements du fait de nos données d'entrées. Par exemple, comme nous avons un grand nombre de colonnes, nous avons modifier le noyau du modèle SVM pour mettre en place un noyau polynomiale car il permet d'accepter un plus grand nombre de colonnes en entrée. Pour le réseau de neurones, comme nous avons un jeu de données relativement petit, nous avons utilisé un solver 'lbfgs' car il permet de produire de meilleurs résultats sur un petit jeu de données.

Interprétation et visualisation des résultats

Explication du modèle retenu

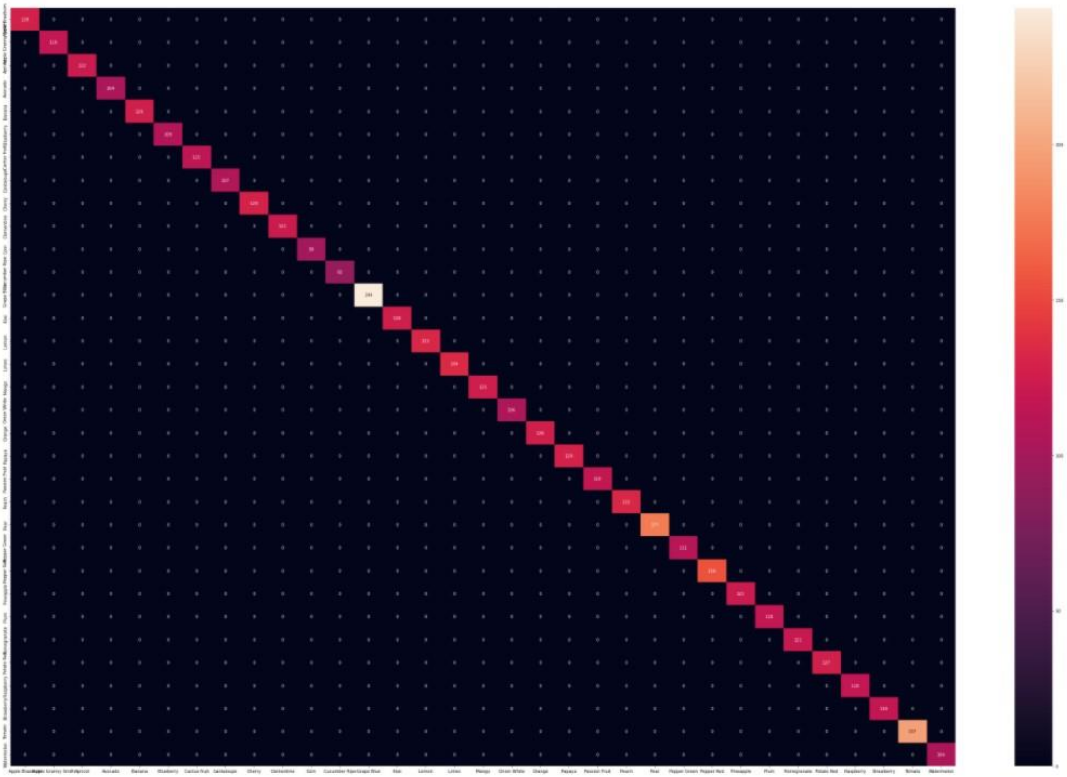
Nous avons choisi d'utiliser le modèle Support Vector Machine (SVM) car c'est un modèle où nous avons observé les meilleurs résultats.



Comme le montre la figure, le principe est simple : il a pour but de séparer les données en classes à l'aide d'une frontière aussi « simple » que possible, de telle façon que la distance entre les différents groupes de données et la frontière qui les sépare soit maximale. Cette distance est aussi appelée « marge » et le SVM est ainsi qualifiés de « séparateurs à vaste marge », les « vecteurs de support » (les points entourés en bleu), étant les données les plus proches de la frontière. Donc le but de ce modèle est de maximiser la marge entre les groupes de données.

Résultats du modèle retenu

En utilisant les paramètres optimaux pour le modèle SVM, on obtient un accuracy score de 1 soit 100% de réussite. A première vue, nous constatons que le modèle fonctionne très bien pour ce jeu de données d'essais.



La matrice est plus visible sur le notebook.

Figure 10: Matrice de confusion du modèle SVM avec les paramètres optimaux

En observant la matrice de confusion, on voit que toutes les prédictions se situent sur la diagonale donc que l'intelligence artificielle n'a commis aucune erreur. On peut donc penser que les données sont très faciles pour l'intelligence artificielle car elles sont toutes formatées de la même façon, c'est-à-dire que toutes les images sont cadrées sur le fruit et ne possèdent pas de détails supplémentaires. Que se passerait-il si les images ne venaient pas du jeu de données ?

Limite du modèle avec des images plus complexes

Comme nous avons obtenu des résultats qui était au-delà de nos espérances, nous avons essayé d'injecter des images plus complexes qui ne viennent pas du jeu de données pour voir si notre intelligence artificielle pouvait s'en sortir. Nous avons essayé d'injecter une image d'abricot avec une feuille pour voir si elle arrivait à détecter l'abricot.

Figure 11 : Abricot avec feuille

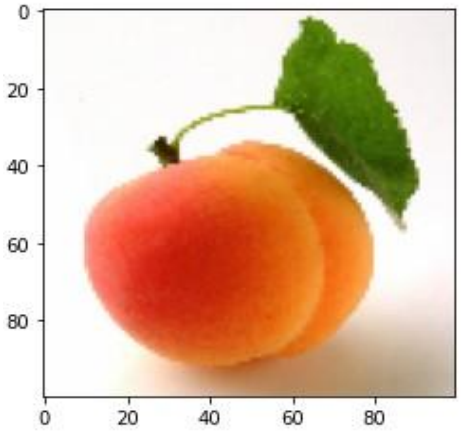


Figure 12 : Abricot avec feuille après redimensionnement

Résultat : Poire

Cet essai fut un échec car l'intelligence artificielle a prédit une poire. Du fait du résultat qui donne un fruit de couleur verte jaune, nous avons pensé que ce résultat était dû à la feuille de l'abricot. Nous avons donc sélectionné une nouvelle image d'abricot sans feuille pour ne pas perturber notre intelligence artificielle.

Figure 13 : Abricot sans feuille

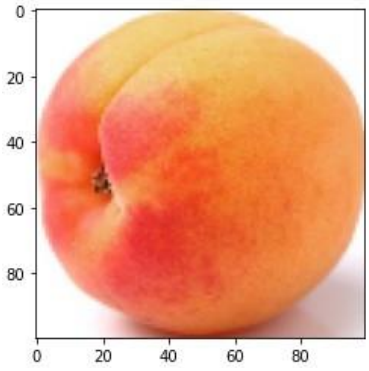


Figure 14 : Abricot sans feuille après redimensionnement

Résultat : Clémentine

Ce nouvel essai fut encore un échec car notre intelligence artificielle n'a pas prédit le bon fruit, elle a prédit une clémentine au lieu d'un abricot. Le résultat s'améliore mais n'est pas encore celui escompté.

On peut donc en déduire que notre modèle fonctionne très bien sur un jeu de données formatées mais lorsqu'il est confronté à de nouvelles images plus complexes qui disposent de détails ou qui ne sont pas formaté de la même manière, l'intelligence artificielle n'arrive plus à reconnaître le fruit.

Améliorations possibles

Pour améliorer notre intelligence artificielle, nous pourrions lui fournir un jeu de données plus importants avec des images qui serait plus complexes car elles possèderaient des détails supplémentaires pour qu'elles puissent s'entrainer sur des images avec des détails et qu'à terme elle puisse reconnaitre des fruits avec leur détail comme l'exemple de l'abricot et sa feuille.

Nous pourrions aussi agréger des images dans un autre format, c'est-à-dire qui ne seraient pas directement cadré sur le fruit à reconnaître. Cela lui permettrait de savoir détecter les zones importantes sur une image.