Compte Rendu du projet N^{o}2 de SY19

Julien Krause et Gabriel Trombini

2 janvier 2024

La répartition des tâches a été équitable, chaque étudiant a réalisé 50 % du travail effectué. Chacun était chargé de travailler avec deux ensembles de données distincts et des réunions était fait chaque semaine pour synthétiser les avancées et s'aider mutuellement.

1 Données Phonemes

Utilisant l'ensemble de données TIMIT (Corpus de parole continue acoustique-phonétique) pour la reconnaissance de la parole, l'accent est initialement mis sur l'identification de cinq phonèmes spécifiques : "sh", "dcl", "iy", "aa" et "ao". Une particularité de ce problème est que l'ensemble de test contient une sixième classe inconnue, représentée par " ?". Nous n'avons accès qu'à l'ensemble d'apprentissage, qui contient les cinq classes mentionnées. Le jeu de données est composé de 126 prédicteurs avec une variable de réponse. La stratégie adoptée dans le projet est la suivante :

Dans un premier temps, nous avons appliqué l'Analyse en Composantes Principales (ACP) pour la réduction de la dimensionnalité. Ensuite, nous avons procédé à la division entre la base d'apprentissage et la base de test. Dans la base d'apprentissage, l'une des cinq classes a été retirée, aboutissant à quatre classes dans la base d'apprentissage et aux cinq classes d'origine dans la base de test. Cette approche vise à simuler une classe inconnue.

Nous avons ensuite entraîné deux classifieurs. Le premier vise à apprendre à identifier les valeurs aberrantes, c'est-à-dire à reconnaître les classes qui pourraient être ajoutées à la base de données. Le deuxième classifieur est formé dans le but de reconnaître les classes d'origine. L'idée est que le premier classifieur, en classifiant une donnée comme une valeur aberrante, l'attribue à la classe " ?". Les autres données sont soumises au deuxième classificateur pour identifier leurs classes spécifiques.

1.1 Analyse du dataset

En examinant les données de l'ensemble, nous constatons que la répartition de nos classes est assez équilibrée, ce qui écarte la nécessité d'utiliser des techniques pour équilibrer les données. Cependant, en ce qui concerne la matrice de confusion, nous remarquons qu'elle n'est pas clairement visible, ce qui suggère la nécessité d'une réduction de la dimensionalité des données.

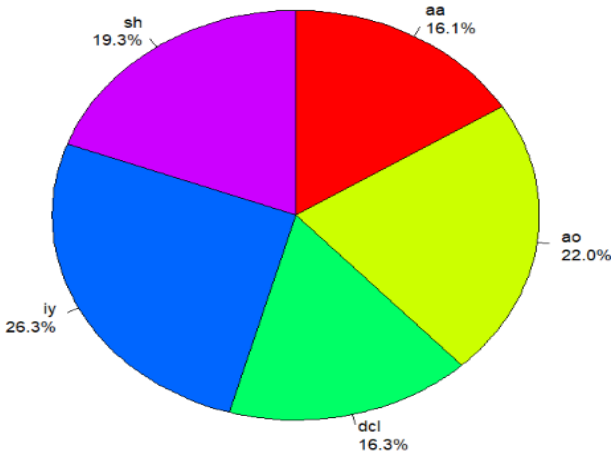

FIGURE 1 – Distribution des classes

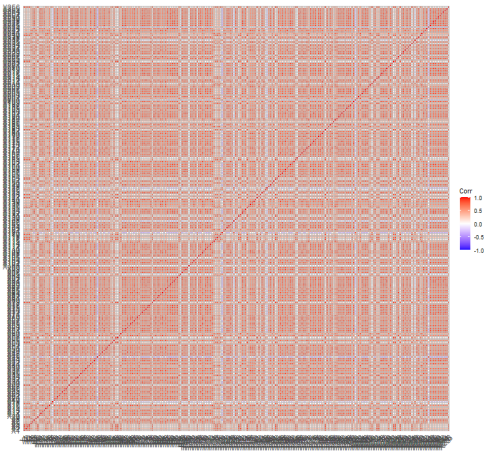

FIGURE 2 – Matrice de corrélation du dataset phonemes

1.2 PCA

Pour la réduction de dimensionnalité, les techniques de régularisation, à la fois LASSO (L1) et Ridge (L2), l'Analyse Discriminante Linéaire (LDA) et l'Analyse en Composantes Principales (PCA) ont été employées. Toutes ces méthodes ont été testées, et celle qui a présenté les meilleures performances a été le PCA, raison pour laquelle nous avons choisi de nous concentrer sur cette dernière. Nous avons utilisé le PCA pour identifier les composantes principales tout en préservant la variabilité des données d'origine. Par la suite, nous avons eu recours à la PCR pour confirmer qu'il n'est pas nécessaire d'utiliser toutes les variables disponibles, tant que l'Erreur Quadratique Moyenne (MSE) ne présente pas une variance significative. Le choix des 25 meilleurs composants était basé sur la recherche d'un équilibre entre une représentation efficace des données et la réduction de la dimensionnalité.

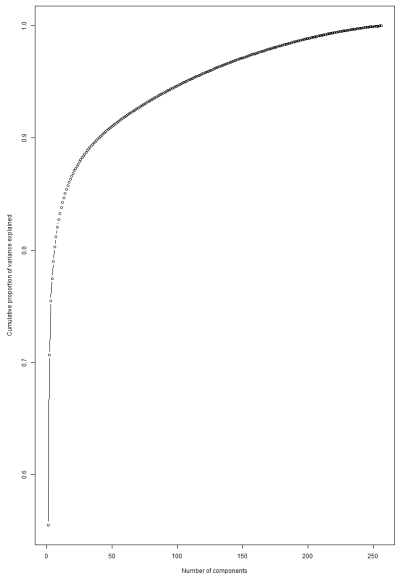

FIGURE 3 – Composantes Principales (PCA)

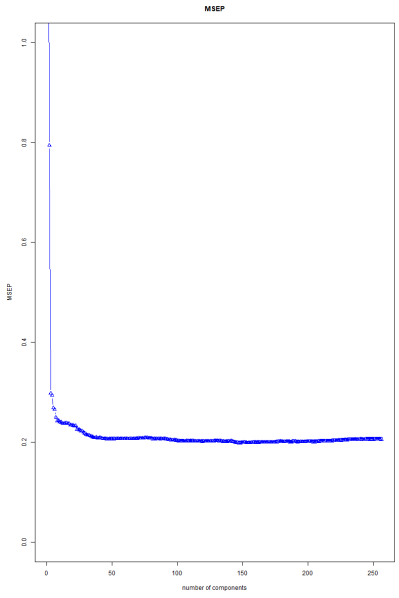

FIGURE 4 – Régression Principale Composante (PCR)

1.3 Classifieurs

Initialement, une validation croisée manuelle a été effectuée en définissant 10 ensembles (folds). Pour chaque ensemble, l'accuracy a été calculée, ce qui s'avère intéressant car à la fin, il est possible d'analyser visuellement la variation entre les ensembles à l'aide d'un boxplot. Ensuite, il a été nécessaire de créer le modèle SVM One Class pour la classification des anomalies (classe " ?"). Les hyperparamètres du SVM One Class ont été choisis par validation croisée, et le modèle final était :

```
fit1 <- ksvm( x = as.matrix(train[, 1:10]),type = "one-svc",kernel = "rbfdot",
kpar = list(sigma = 0.01), nu = 0.01)
```

Pour la classification des classes restantes, les classificateurs suivants ont été utilisés : Régression Logistique, SVM, et Random Forest :

1.3.1 Comparaison des classificateurs

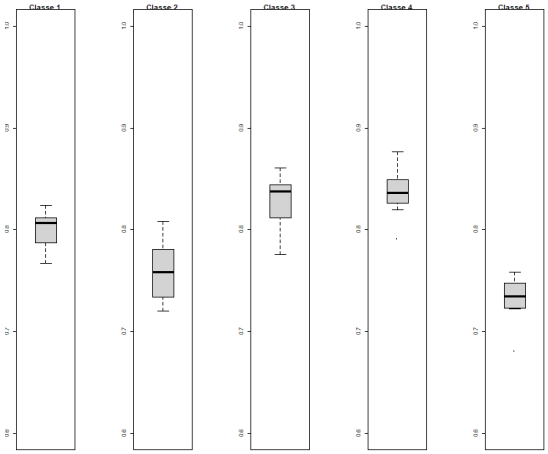

FIGURE 5 – L'accuracy du modèle Random Forest

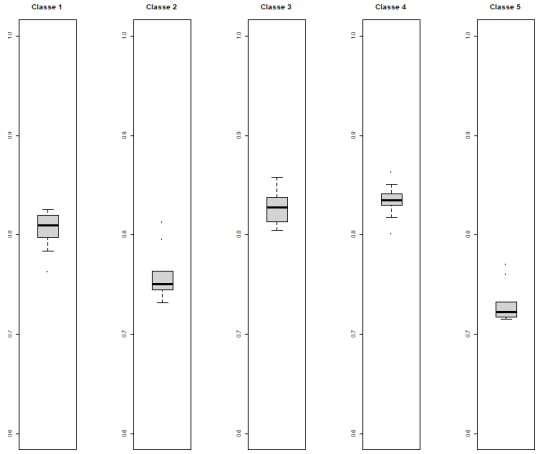

FIGURE 6 – L'accuracy du modèle Regression Logistique

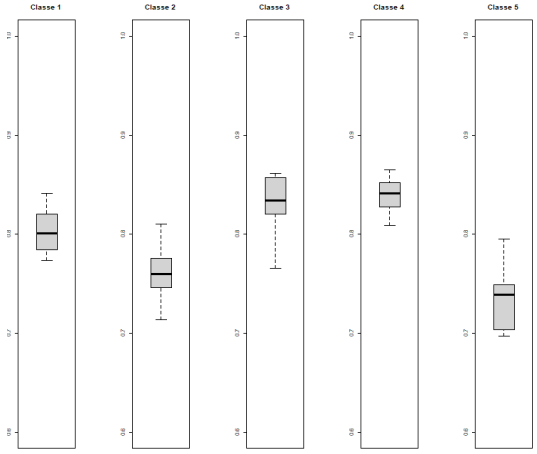

FIGURE 7 – L'accuracy du modèle SVM

1.4 Conclusion

Pendant l'évaluation des divers classifieurs, pour laquelle une validation croisée a été effectuée afin de déterminer les meilleurs hyperparamètres, nous avons adopté des approches variées pour la construction de modèles prédictifs. En ce qui concerne le Random Forest, nous avons utilisé le modèle suivant :

```
modele <- randomForest(y ~ ., data = train, ntree = 200, mtry = 8)
```

La Régression Logistique, quant à elle, a été implémentée comme suit :

```
modele <- multinom(y ~ ., data = train)
```

En ce qui concerne le SVM, nous avons employé le modèle suivant :

```
modele <- svm(y ~ ., data = train, kernel = "radial", cost = 12, gamma = 0.01)
```

Après l'implémentation et l'entraînement de ces modèles, nous avons calculé l'accuracy moyenne pour chaque classe, comme présenté dans le tableau ci-dessous.

TABLE 1 – Accuracy Moyenne des Classificateurs par Classe

Classe Exclue	**Ramdom Forest**	**Logistic Regression**	**SVM**
1	0.80	0.80	0.80
2	0.76	0.75	0.76
3	0.82	0.85	0.83
4	0.83	0.83	0.84
5	0.74	0.73	0.73

En analysant les résultats, nous constatons qu'il n'y a pas de différence significative entre les classificateurs testés. Cependant, après une considération attentive, nous avons décidé d'adopter la Regression Logistique comme notre modèle choisi. Ce choix a été soutenu par une variance légérement plus faible comme vous pouvez le constater sur la *Figure 6*. Ce classifieur a fourni des résultats cohérents et appropriés pour notre ensemble de données.

2 Données Robotics

Ce dataset est constitué de 8 variables et de 5000 observations proposant un problème de regression pour prédire les valeurs.

2.1 Analyse des données

Dans un premier temps, nous avons cherché à comprendre les données. Pour cela nous avons effectué une ACP et visualisé la matrice de corrélation.

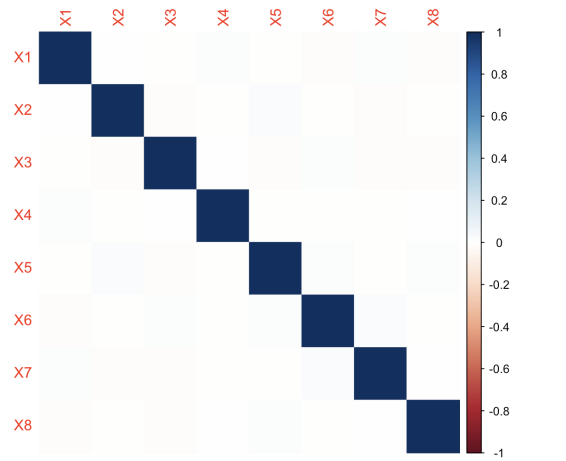

FIGURE 8 – Matrice de corrélation du dataset robotics

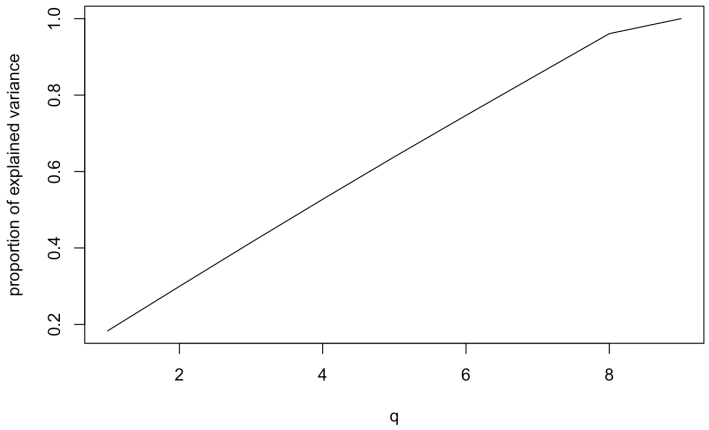

FIGURE 9 – ACP du dataset robotics

A la suite de cette première étape nous pouvons constater que chacune des variables est importante et qu'il ne sera pas possible de baisser la dimension du dataset sans perdre d'information. En effet, on constate via la matrice de corrélation que les variables sont indépendantes puisque la corrélation est nulle pour chacune des variables à l'exception d'elle-même. L'importance de chacune des variables est confirmée par l'ACP. Le cumul de la proportion de chaque composante est presque linéaire démontrant l'égal répartition de l'information parmi les variables. Ainsi, par la suite nous ne chercherons pas à réduire la dimension mais plutôt à l'augmenter pour voir si la transformation de variable peut donner davantage d'information pour entraîner nos modèles.

2.2 Classifieurs

Avant d'appliquer les modèles nous avons chercher à créer de nouvelles variables en les mettant au carré, au cube... Cependant, l'ajout de ces variables n'a pas eu d'influence mejeur sur nos résultats. Nous avons donc fait le choix de ne pas les ajouter. Après avoir effectué cette phase d'analyse nous avons comparé différents modèles de régression étudiés ce semestre. Nous avons pu utiliser la régression linéaire avec la régularisation ridge et lasso, les arbres de régressions, les splines... Pour comparer ces modèles, nous avons utilisé une seed afin d'effectuer une cross validation avec les mêmes train set et test set. Le modèle ayant retenu notre attention est la SVM car elle donnait une MSE plus petite que les autres modèles. Une fois le choix du modèle SVM effectué, nous avons identifié les meilleurs hyperparamètres de ce modèle. Au sein

du langage R, on retrouve une fonction de grid search permettant de déterminer les meilleurs hyper paramètres. Vous pouvez trouver ci-dessous l'utilisation que nous en avons fait.

```
Cs <- seq(1, 10, length.out = 10)
gamma <- seq(-5, 5, length.out = 10)
sigmas <- 10^seq(-8, 0, length.out = 10)

best_param <- tune(svm, y~., data = data.frame(X.train, y.train),
                ranges = list(cost = Cs, gamma = gamma, sigma = sigmas-08),
                tunecontrol = tune.control(sampling = "cross"))
```

Afin d'imager le comportement de cette fonction, nous avons réaliser un graphique permettant de visualiser l'erreur de classifieur ksvm en fonction du paramètre C. La fonction va ainsi selectionner la valeur de C minimisant l'erreur.

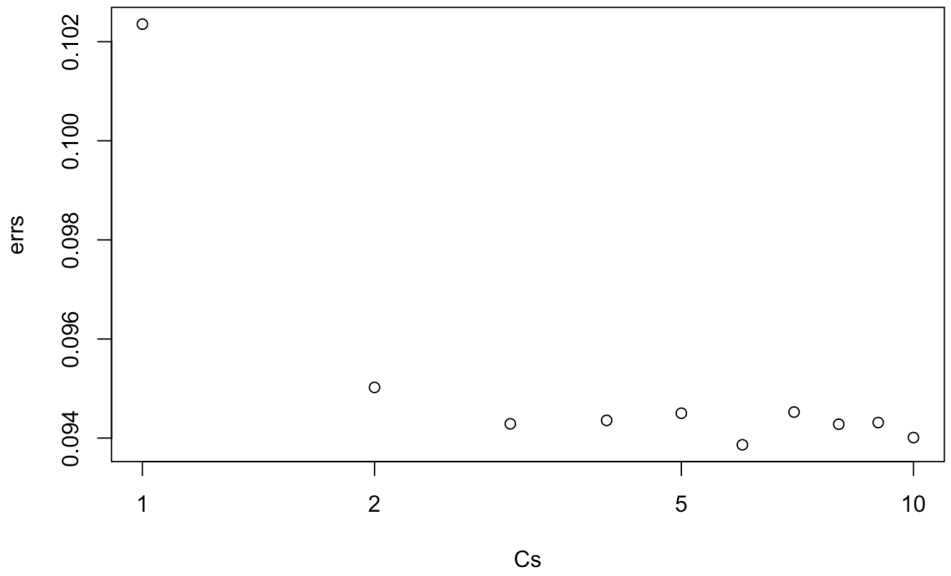

FIGURE 10 – Évolution de l'erreur (logarithmique) en fonction de C

Finalement, nous avons soumis le modèle ci-dessous.

```
ksvm(y~.,data=data.train ,scaled=TRUE ,type="eps-svr",
                kernel="rbfdot" ,C=3 ,epsilon=1e-09, sigma=1e-08)
```

3 Données Construction

Ce dataset est constitué de 104 variables et de 250 observations proposant un problème de regression pour prédire les valeurs.

3.1 Analyse des données

Dans un premier temps, nous avons cherché à comprendre les données. Pour cela nous avons effectué une ACP et visualisé la matrice de corrélation.

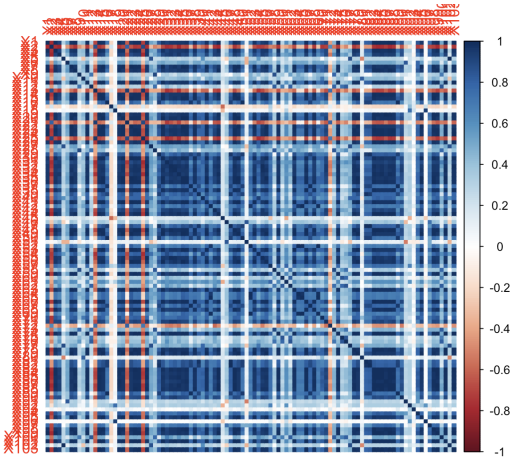

FIGURE 11 – Matrice de corrélation du dataset construction

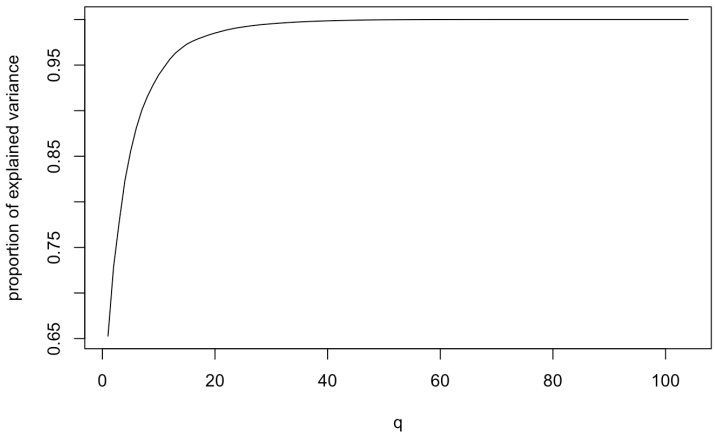

FIGURE 12 – ACP du dataset construction

A la suite de cette première étape, nous pouvons constater que nous allons pouvoir réduire la dimension de ce dataset. En effet, la matrice de corrélation révéle que de nombreuses variables sont corrélées. De plus, en réalisant l'ACP, on observe que les 20 première composantes détiennent environ 98% de l'information détenu par les données. Nous allons donc chercher à diminuer le nombre de variable utilisé avant d'appliquer nos modèles. Il faudra néanmoins faire atteention à garder un modèle souple puisque nous diponsons de seulement 250 observations.

3.2 Classifieurs

Le nombre de prédicteur étant important, l'utilisation de méthode pour réduire leur nombre est nécessaire. Pour ce faire, les méthodes de selection de variable en propagation avant et arrière avec les indicateurs de AIC et BIC sont pertinantes. Nous avons donc cherché à effectuer une séléction des meilleurs prédicteurs en leur appliquant des transformation (carré, cube...) afin de déterminer le prédicteur avec le plus d'information. Par la suite, ce sont les modèles de régression sur des variables complexes qui sont exploré. Pour ce faire, une première selection des variables est réalisé pour chaque fonction de transformation utilisé. Cette première étape est nécessaire pour que la regression puisse se faire sans avoir plus de variable prédictive que de valeurs connu. Ensuite, l'ensemble des variables conservé est à nouveau selectionné dans un régression les regroupants toutes. A chaque étape un validation croisé en 5 blocs permet de générer 5 sous-ensemble de variable selection. De chaque validation croisé, on choisira de

conserver les variables apparaissant X fois (de 1 à 5) dans les 5 selections. A l'issu de ce rocessus nous avons décidé de garder les variables apparaissant 5 fois.

Les variables étaient : $X92^4$, $X45^3$, $X48^3$, $X57^3$, $X57^2$, $X93$, $X57$et$X55$.

Lors de la réalisation de la validation croisé, nous arrivions à une MSE moyenne de 700. Cependant, malgré la cross validation réalisée pour la sélection des variables et pour l'entraînement de la régression linéaire ; la MSE lors du test sur Maggle a explosé. Cela très certainement dû à de l'overfitting car nous avons réduit la dimension à 8 variables sachant que l'une d'entre elle revenait plusieurs fois avec des transformations différentes.

Ainsi, nous avons décidé d'utiliser les résultats de l'ACP pour réduire la dimension en utilisant les 20 premières composantes principales. Comme pour le dataset robotics, nous avons cherché le modèle le plus efficace et deux modèles se sont démarqués. Il s'agit de ksvm et de regression random forest. Nous avons cherché à optimiser les paramètres avec ces deux modèles. C'est finalement regression random forest qui s'est avérée la plus efficace. Comme pour ksvm, il existe en R une fonction permettant de tester les différents hyperparamètres. Vous pouvez voir ci-dessous la démarche effectuée ainsi que le modèle soumis.

```
# Créer la grille avec les colonnes mtry et ntree
grid <- expand.grid(mtry = c(2, 4, 6, 8), ntree = seq(1, 1000, length.out = 10))

# Définir la fonction de contrôle
ctrl <- trainControl(method = "cv", number = 10)

# Effectuer la recherche sur la grille
rf_grid <- train(y ~ ., data = train, method = "rf", trControl = ctrl,

tuneGrid = grid)

# Accéder aux meilleurs hyperparamètres
best_ntree <- rf_grid$bestTune$ntree
best_mtry <- rf_grid$bestTune$mtry

# Entraîner le modèle final avec les meilleurs hyperparamètres
final_model <- randomForest(y ~ ., data = train, ntree = 200, mtry = best_mtry)
```

4 Données Images (BONUS)

Ce dataset est composé de 1782 images en couleur (au format JPEG) représentant des voitures, des fruits et des chiens. Les images sont de tailles variables. La tâche consiste à prédire le contenu de nouvelles images appartenant à l'un de ces trois types.

4.1 Analyse du dataset

La base de données contient des images de tailles variées, avec 32.25% d'images de voitures, 26.32% d'images de chiens et 37.43% d'images de fruits.

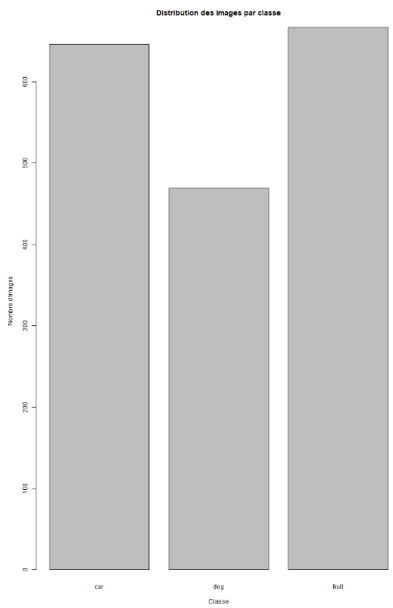

FIGURE 13 – Distribution des images par classe

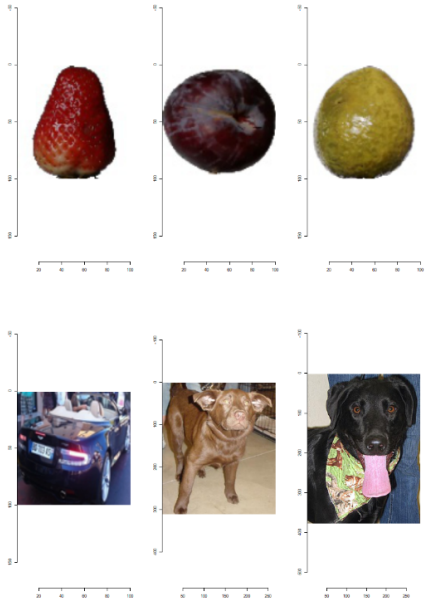

FIGURE 14 – Images aléatoires de la base de données.

TABLE 2 – Distribution de la largeur et de la hauteur des images

	hauteur	**largeur**
Min.	50.0	43.0
1st Qu.	100.0	100.0
Median	100.0	100.0
Mean	157.7	151.4
3rd Qu.	158.8	135.2
Max.	492.0	493.0

4.2 Augmentation de Données

Une technique employée dans ce projet est l'Augmentation de Données, une méthode utile dans la construction de modèles permettant d'élargir la taille de l'ensemble d'entraînement sans acquérir de nouvelles images. L'objectif est d'instruire le modèle non seulement avec l'image originale, mais aussi avec des modifications de l'image telles que la rotation, l'inversion, le zoom, le recadrage, etc. Cela s'avère très utile pour prévenir le surajustement et pour obtenir de nouvelles images, car la base de données utilisée dans le projet est limitée en nombre.

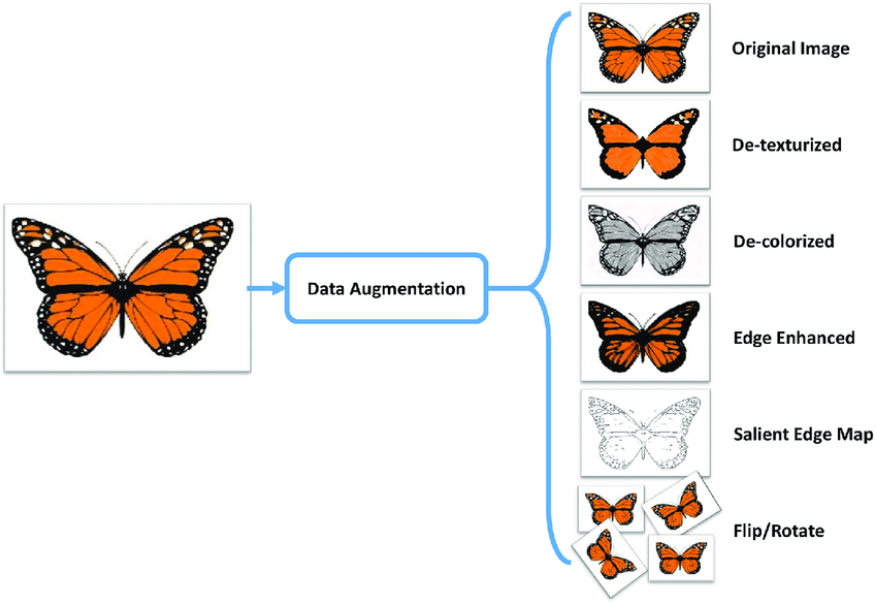

FIGURE 15 – Illustration de l'augmentation de données

En utilisant la fonction `image_data_generator` du package `keras`, nous pouvons définir tous les paramètres de la génération de nouvelles images modifiées :

```
train_data_gen <- image_data_generator(
    rescale = 1/255, # Mise à l'échelle de la valeur des pixels
    horizontal_flip = TRUE, # Retournement horizontal de l'image
    vertical_flip = TRUE, # Retournement vertical de l'image
    rotation_range = 45, # Rotation de l'image de 0 à 45 degrés
    zoom_range = 0.25, # Plage de zoom avant ou arrière
    validation_split = 0.2 # 20 % des données en tant que données de validation
)
```

4.3 Classificateur

Le classifieur de réseaux neuronaux utilisé était le Convolutional Neural Network. Pour expliquer brièvement le fonctionnement d'un CNN, il faut d'abord comprendre que, pour extraire des caractéristiques de l'image, un noyau est utilisé, c'est-à-dire une matrice de taille spécifique, telle qu'une matrice 3x3. Le noyau se déplace à travers l'image, capturant des caractéristiques et créant de nouvelles caractéristiques convolutives. Le résultat est le score de la caractéristique convolutive, qui est plus élevé lorsque la section de l'image est plus similaire au noyau.

Pour mettre en évidence les caractéristiques importantes et réduire la dimension, le Max pooling est utilisé, sélectionnant uniquement la valeur maximale dans une fenêtre donnée. Ensuite, la couche de mise à plat (flattening) convertit la matrice 2D extraite en une matrice 1D pour continuer à utiliser des couches totalement connectées.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 64, 64, 16)        448
 max_pooling2d (MaxPooling2D)  (None, 32, 32, 16)      0
 flatten (Flatten)           (None, 16384)             0
 dense (Dense)               (None, 16)                262160
 Sortie (Dense)              (None, 3)                 51
=================================================================
Total params: 262659 (1.00 MB)
Trainable params: 262659 (1.00 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Pour la compilation, la fonction de perte utilisée était `sparse_categorical_crossentropy` et l'optimiseur était l'algorithme `adam` avec un taux d'apprentissage de 0.01.

```
model %>%
  compile(
    loss = "sparse_categorical_crossentropy",
    optimizer = optimizer_adam(lr = 0.01),
    metrics = "accuracy"
  )
```

Et enfin, le modèle a été entraîné pendant 50 époques

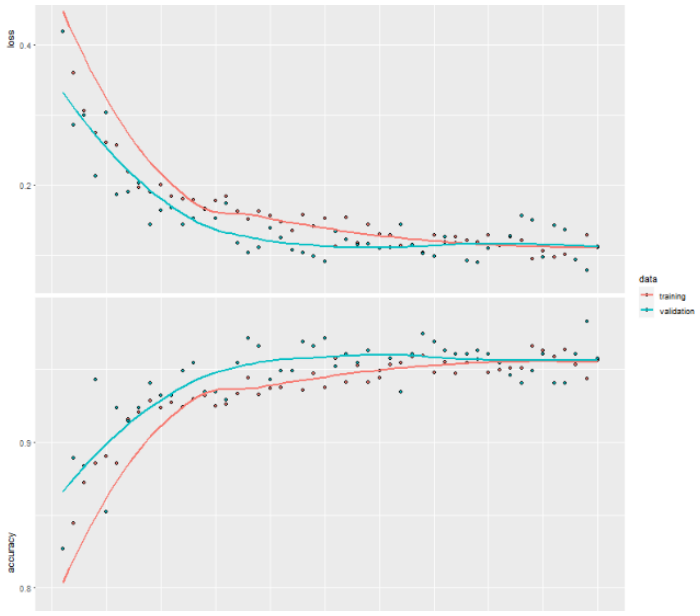

FIGURE 16 – Historique de l'entraînement du modèle

En conclusion, notre modèle a atteint une précision d'environ 96 % en moyenne, aussi bien dans les données de validation que dans l'apprentissage, démontrant que la technique d'augmentation d'images a donné de bons résultats.

5 Conclusion

En conclusion, nous avons toujours adopté une méthodologie axée sur une analyse des données avant de commencer à élaborer des modèles. Cette étape d'analyse nous a permis de comprendre les données et de ne pas appliquer sans réfléchir tous les modèles vus en cours. Enfin ce projet nous a permis de travailler en équipe et de savoir être résilient. En effet, notamment pour le dataset Construction, nous n'avions pas anticipé l'overfitting dû au faible nombre de données. Ainsi, nous avons dû nous adapter une fois un premier test réalisé afin de fournir un meilleur résultat.