Rapport MLRF

BAZAN Corentin
ROSARD Simon

Classification sur la base de données CIFAR-10

5 juillet 2023

Table des matières

1 Lien Repo GIT

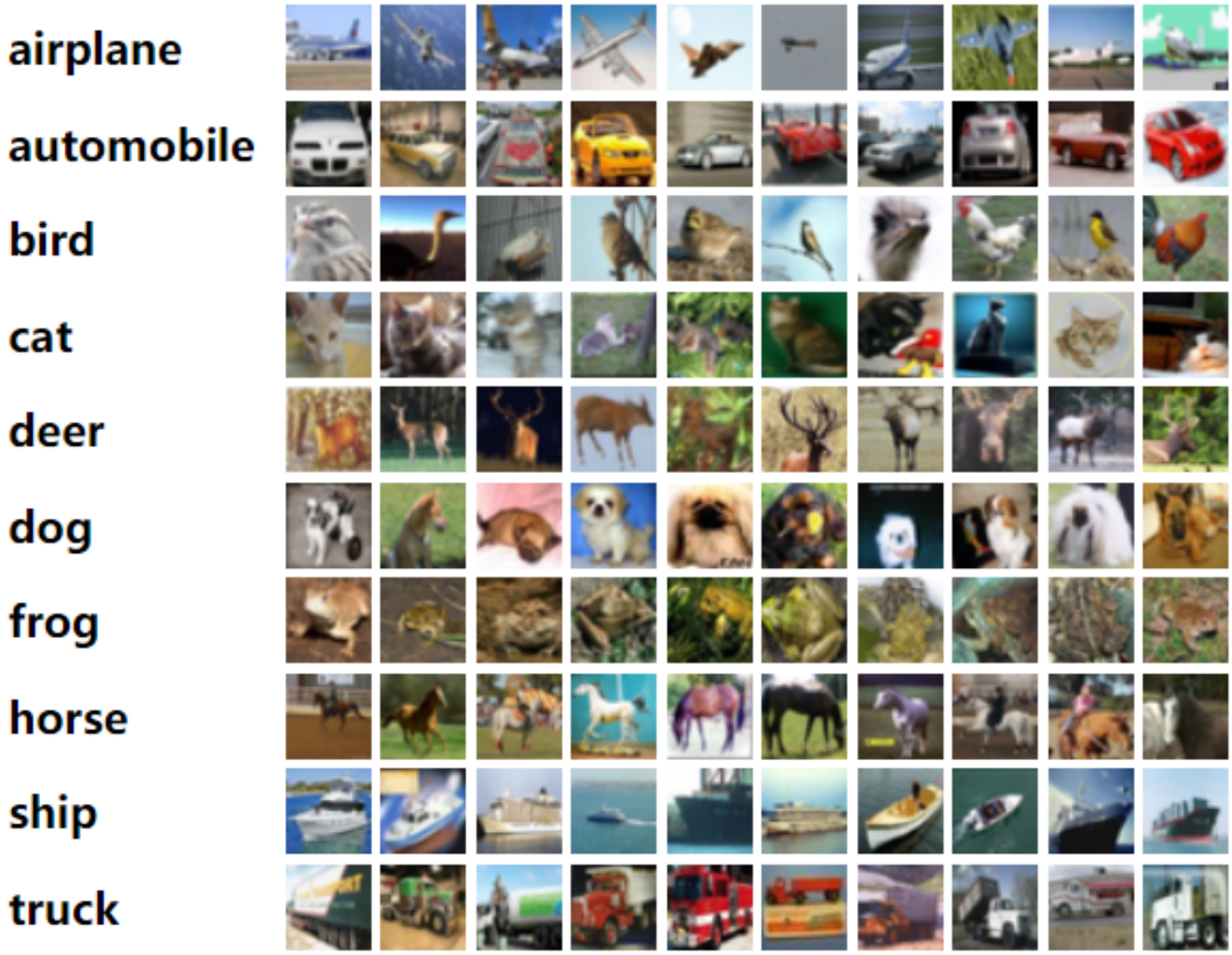
Lien : https://github.com/FullNebulon/mlrf

2 Introduction

L'objectif principal de ce projet est de classifier les images du dataset CIFAR-10, un ensemble bien connu et largement utilisé dans le domaine de l'apprentissage automatique et de la vision par ordinateur. Nous utiliserons 3 modèles différents et comparerons les résultats obtenus.

3 Présentation du dataset

Le dataset CIFAR-10 est composé de 60000 images en couleur de taille 32x32, réparties en 10 classes, avec 6000 images par classe. Les classes sont totalement mutuellement exclusives, allant des voitures aux chats, en passant par les avions et les oiseaux. Voici quelques exemples des images que l'on peut trouver au sein de ce dataset :



4 Etat de l'art

Depuis son introduction en 2009, le dataset CIFAR-10 a été un banc d'essai populaire pour de nombreux algorithmes de classification d'images. Diverses approches ont été explorées, des techniques traditionnelles d'apprentissage automatique aux architectures de réseaux neuronaux profonds.

Au départ, les méthodes traditionnelles d'apprentissage automatique, telles que les SVM (Support Vector Machines), le k-NN (k-Nearest Neighbors) et les forêts aléatoires, étaient fréquemment utilisées. Ces techniques reposent souvent sur l'extraction manuelle de caractéristiques des images, comme les histogrammes de couleurs, le SIFT (Scale-Invariant Feature Transform), le HOG (Histogram of Oriented Gradients), pour ensuite les utiliser comme entrées de ces algorithmes de classification.

Cependant, avec l'avènement des réseaux de neurones convolutifs (Convolutional Neural Networks, CNNs) et l'explosion de l'apprentissage profond, l'approche a évolué. Les CNNs ont prouvé leur efficacité en matière de vision par ordinateur, étant capables d'apprendre automatiquement des caractéristiques pertinentes à partir des images brutes, ce qui élimine le besoin d'une étape d'extraction de caractéristiques manuelle. Des architectures CNN comme LeNet, AlexNet, VGGNet, GoogLeNet et ResNet ont toutes été appliquées avec succès sur le CIFAR-10, avec une amélioration progressive des performances au fil du temps.

Les techniques d'augmentation des données, comme les rotations, les décalages, les inversions et les zooms aléatoires, sont également couramment utilisées pour augmenter la taille du jeu de données d'entraînement et réduire le surapprentissage. De plus, des techniques de régularisation comme le dropout et la normalisation par lots ont été appliquées pour améliorer les performances du modèle.

Plus récemment, l'utilisation de l'apprentissage par transfert a gagné en popularité. Cette approche consiste à pré-entraîner un modèle sur une tâche plus vaste, comme la classification d'ImageNet, puis à affiner ce modèle sur le CIFAR-10. Cela permet d'exploiter les caractéristiques apprises sur un ensemble de données plus grand et plus diversifié, améliorant ainsi les performances sur le CIFAR-10.

Aujourd'hui, les meilleures performances sur le CIFAR-10 étaient obtenues grâce à des architectures de réseaux neuronaux convolutifs profonds, parfois combinées à des techniques d'augmentation des données et de régularisation, pour une précision d'environ 99,7% pour le meilleur modèle.

5 Méthodes d'extraction et modèles utilisés

Pour ce projet, nous avons utilisé 3 méthodes d'extraction de features ainsi que 3 modèles de classification, pour un total de 9 modèles. Les 3 méthodes d'extraction de features sont les suivantes :

— Vecteur aplati des images
— Extraction HOG
— Extractions de couleur au format HSV

Les 3 modèles de classification utilisés sont les suivants :
— Régression logistique One VS All
— Random forest
— Classification par descente de gradient stochastique (SGD Classifier)

6 Extraction de couleurs au format HSV

L'extraction de couleur est une étape courante du traitement d'images et de la vision par ordinateur, surtout lorsqu'on veut utiliser les couleurs comme caractéristiques d'une image. Voici une description générale des étapes couramment utilisées lors de l'extraction de couleurs :

— Conversion de couleur : La plupart des images sont en format RGB, il a donc fallu les convertir au format HSV.

— Quantification de couleur : Une fois l'image convertie dans l'espace colorimétrique approprié, les valeurs de couleur continues sont souvent quantifiées en un nombre plus petit de bacs ou de niveaux de couleur. Dans notre cas, chaque canal couleur a été divisé en 8 bacs, toutes les couleurs tombant dans le même bac de valeurs seront traitées comme identiques.

— Calcul d'histogramme : Pour chaque pixel de l'image, on compte dans quel bac tombent ses valeurs de couleur, et on incrémente le compte correspondant. Cela donne un histogramme qui est une représentation de la distribution des couleurs dans l'image.

— Normalisation : Enfin, l'histogramme est souvent normalisé pour que sa somme ou sa norme soit égale à 1. Cela rend la représentation de couleur invariante à la taille de l'image : une grande image et une petite image avec les mêmes couleurs auront le même histogramme. Dans notre cas, ce n'est pas utile car toutes les images sont au format 32x32. Néanmoins, cette normalisation permet tout de même d'améliorer les performances.

Ces étapes fournissent une représentation simple et compacte de la couleur d'une image, qui peut être utilisée pour des tâches comme la classification d'images, la recherche d'images, la détection d'objets, etc.

6.1 Random Forest

6.1.1 Recherche des hyperparamètres

Pour ce modèle, nous avons réalisé un grid search sur 4 hyperparamètres :

— n_estimators ([50, 100]) : Il s'agit du nombre d'arbres dans la forêt. Plus il y a d'arbres, plus le modèle sera robuste et moins susceptible d'être sujet au surapprentissage, à défaut d'augmenter le temps de calcul.

— max_depth ([None, 10, 20]) : Il s'agit de la profondeur maximale de chaque arbre. Une profondeur trop élevée peut conduire à un modèle susceptible de surapprendre. Une profondeur plus faible peut conduire à un modèle plus simple, mais peut également entraîner un sous-apprentissage.

— min_samples_split ([5, 10]) : Il s'agit du nombre minimum d'échantillons requis pour diviser un nœud interne. C'est-à-dire que si un nœud a moins d'échantillons que min_samples_split, il ne sera pas divisé, et l'algorithme arrêtera de chercher des divisions pour ce nœud.

— min_samples_leaf ([1, 2, 4]) : C'est le nombre minimum d'échantillons requis pour être à un nœud feuille.

Avec ces choix de valeurs, nous avons obtenue la meilleure combinaison suivante :

— 'n_estimators' : 100

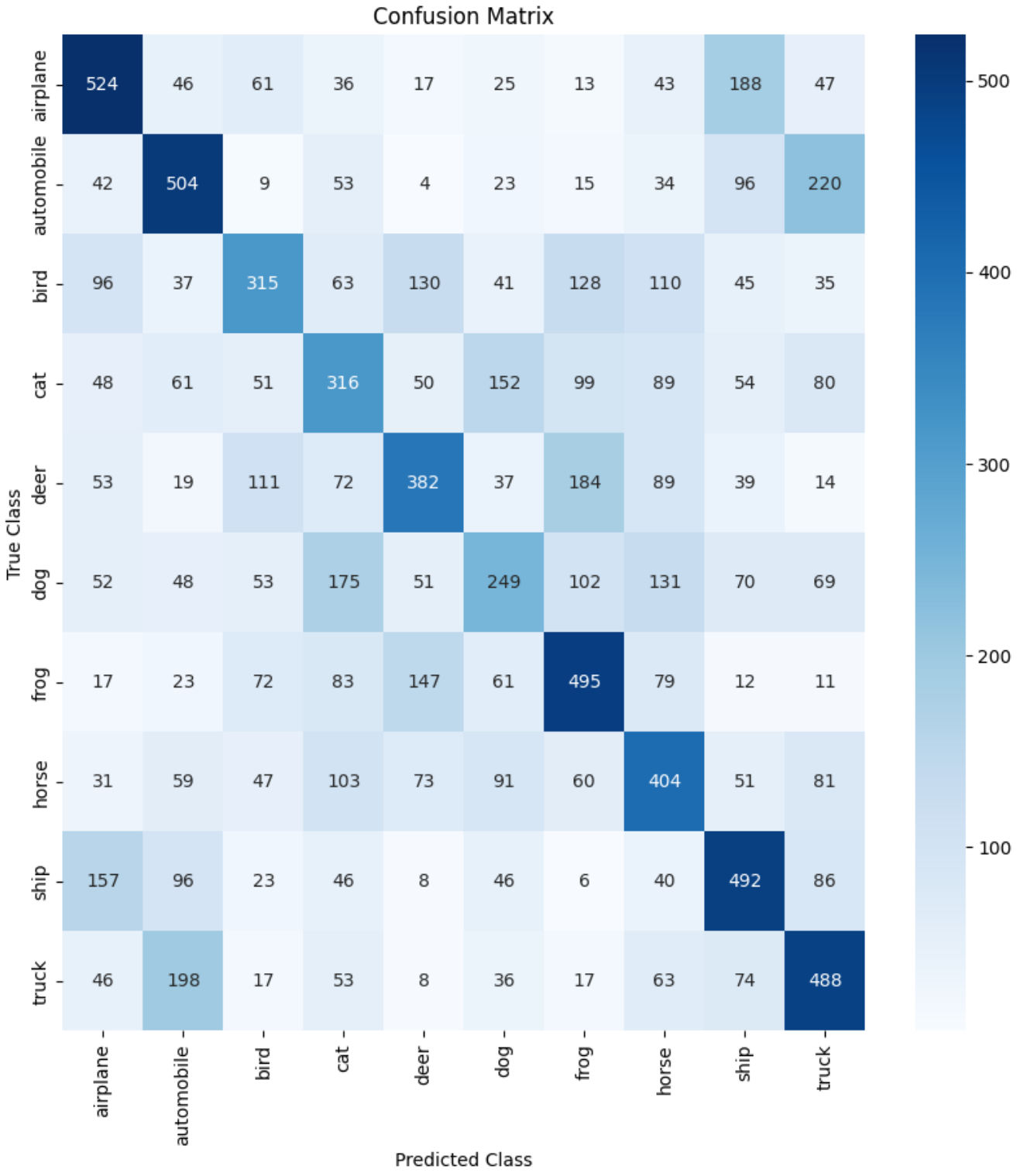
— 'max_depth' : None

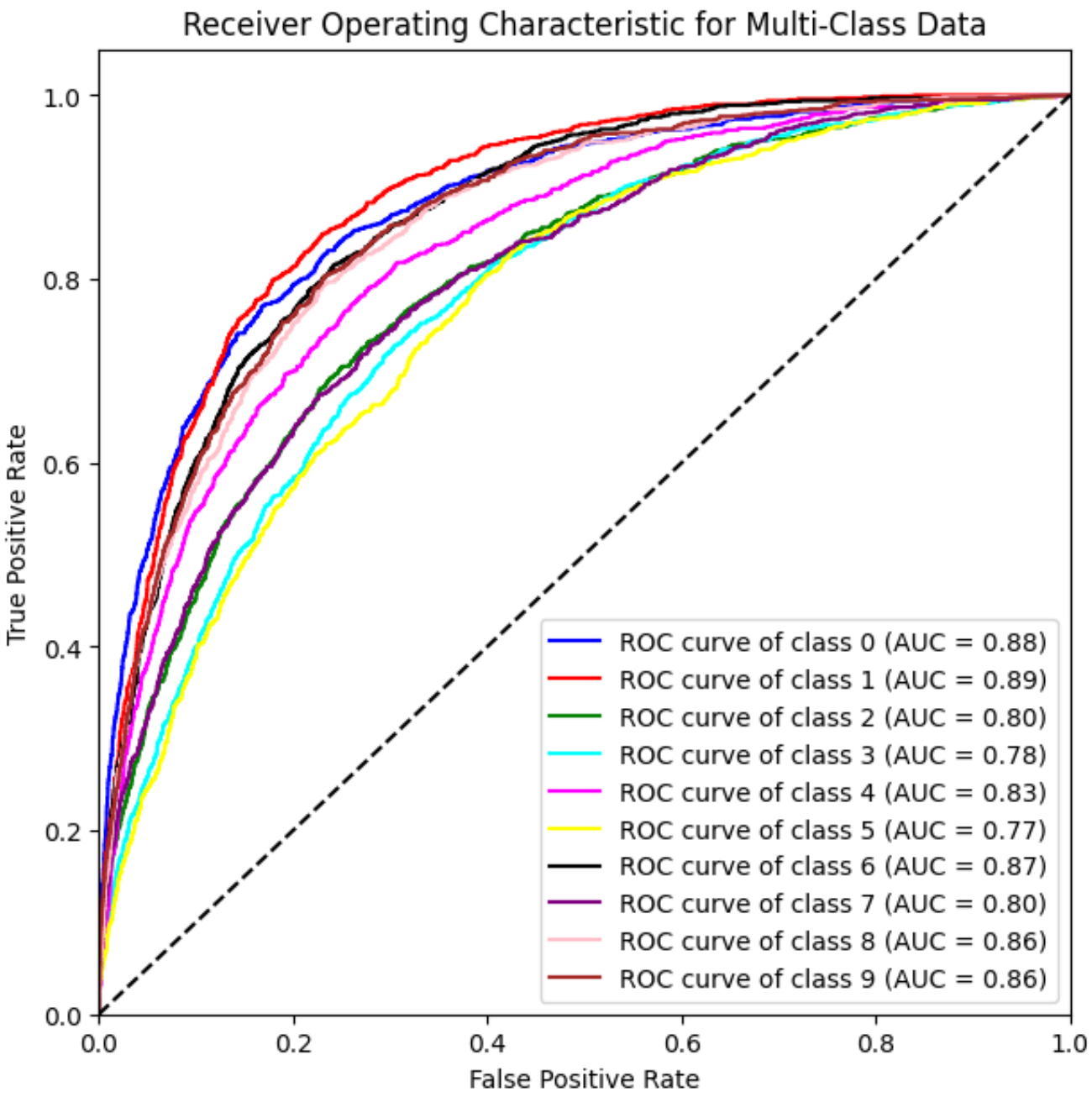
— 'min_samples_split' : 5

— 'min_samples_leaf' : 1

6.1.2 Précision obtenue et visualisation

Pour ce modèle, nous avons obtenue une précision de 41,72%, avec la matrice de confusion et les courbes ROC suivantes :


Confusion Matrix
True Class
Predicted Class
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
524 46 61 36 17 25 13 43 188 47
42 504 9 53 4 23 15 34 96 220
96 37 315 63 130 41 128 110 45 35
48 61 51 316 50 152 99 89 54 80
53 19 111 72 382 37 184 89 39 14
52 48 53 175 51 249 102 131 70 69
17 23 72 83 147 61 495 79 12 11
31 59 47 103 73 91 60 404 51 81
157 96 23 46 8 46 6 40 492 86
46 198 17 53 8 36 17 63 74 488
500
400
300
200
100



6.2 Régression logistique

6.2.1 Recherche des hyperparamètres

Pour ce modèle, nous avons réalisé un grid search sur 2 hyperparamètres :

— C ([0.01, 0.1, 1, 10]) : Dans les modèles de régression logistique, C est le paramètre de régularisation inverse. La régularisation est une technique utilisée pour prévenir le surapprentissage en ajoutant une pénalité à la complexité du modèle. Une valeur plus petite pour C spécifie une régularisation plus forte, c'est-à-dire une plus grande pénalité pour la complexité du modèle, ce qui peut aider à prévenir le surapprentissage.

— penalty (['l1', 'l2']) : Cet hyperparamètre spécifie le type de pénalité de régularisation à utiliser. Les options typiques pour penalty sont 'l1', 'l2'. 'l1' correspond à la régularisation Lasso, qui tend à produire des modèles

avec moins de coefficients non nuls. 'l2' correspond à la régularisation Ridge, qui tend à réduire tous les coefficients vers zéro, mais n'élimine pas complètement les caractéristiques.

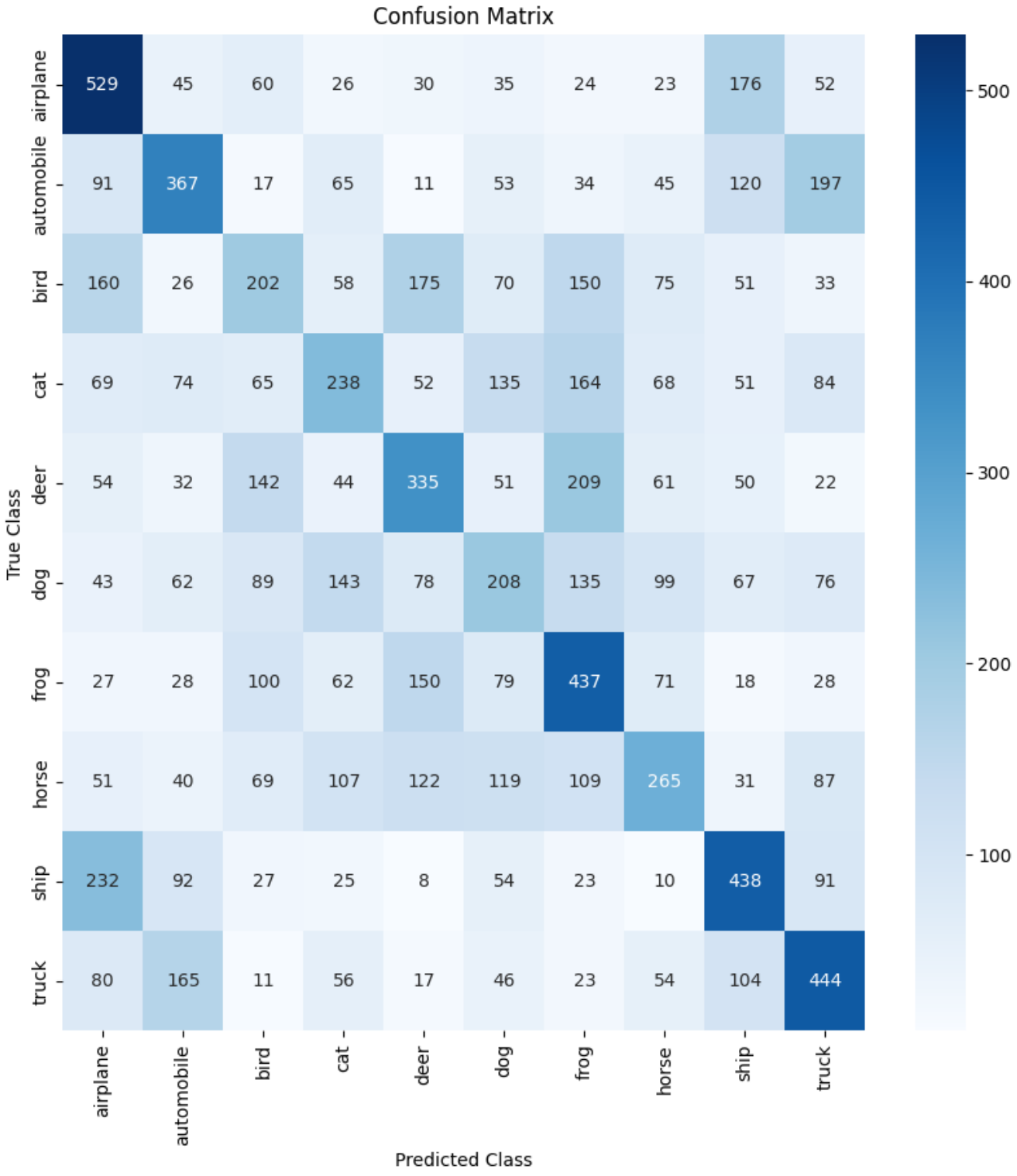
Avec ces choix de valeurs, nous avons obtenue la meilleure combinaison suivante :

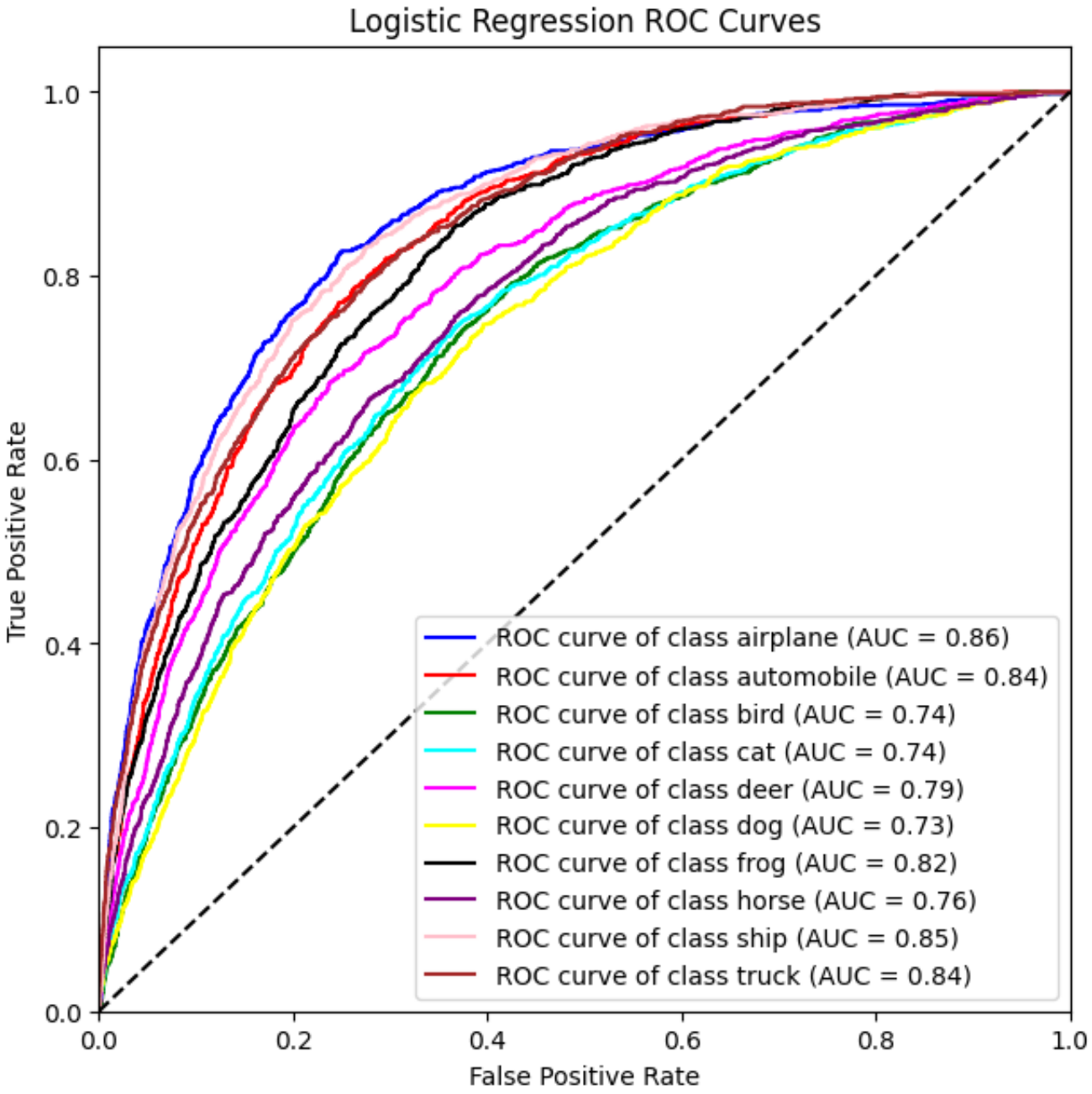
— C : 1

— penalty : L2

6.2.2 Précision obtenue et visualisation

Pour ce modèle, nous avons obtenue une précision de 34,63%, avec la matrice de confusion et les courbes ROC suivantes :


Confusion Matrix
True Class
Predicted Class
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
529 45 60 26 30 35 24 23 176 52
91 367 17 65 11 53 34 45 120 197
160 26 202 58 175 70 150 75 51 33
69 74 65 238 52 135 164 68 51 84
54 32 142 44 335 51 209 61 50 22
43 62 89 143 78 208 135 99 67 76
27 28 100 62 150 79 437 71 18 28
51 40 69 107 122 119 109 265 31 87
232 92 27 25 8 54 23 10 438 91
80 165 11 56 17 46 23 54 104 444
500
400
300
200
100



6.3 SGD Classifier

6.3.1 Recherche des hyperparamètres

Pour ce modèle, nous avons réalisé un grid search sur 2 hyperparamètres :

— loss (['hinge', 'log']) : Il s'agit de la fonction de perte à utiliser. 'hinge' est la fonction de perte standard pour les SVM, tandis que 'log' donne une régression logistique, une approche probabiliste de la classification binaire.

— alpha ([0.001, 0.01, 0.1, 1]) : C'est essentiellement le contraire du paramètre C dans une régression logistique, c'est-à-dire qu'une valeur plus élevée de alpha signifie une régularisation plus forte (plus de pénalité sur la complexité du modèle), ce qui peut aider à prévenir le surapprentissage.

— penalty (['l1', 'l2']) : Semblable à la régression logistique, cet hyperparamètre spécifie le type de pénalité de régularisation à utiliser.

Avec ces choix de valeurs, nous avons obtenue la meilleure combinaison suivante :

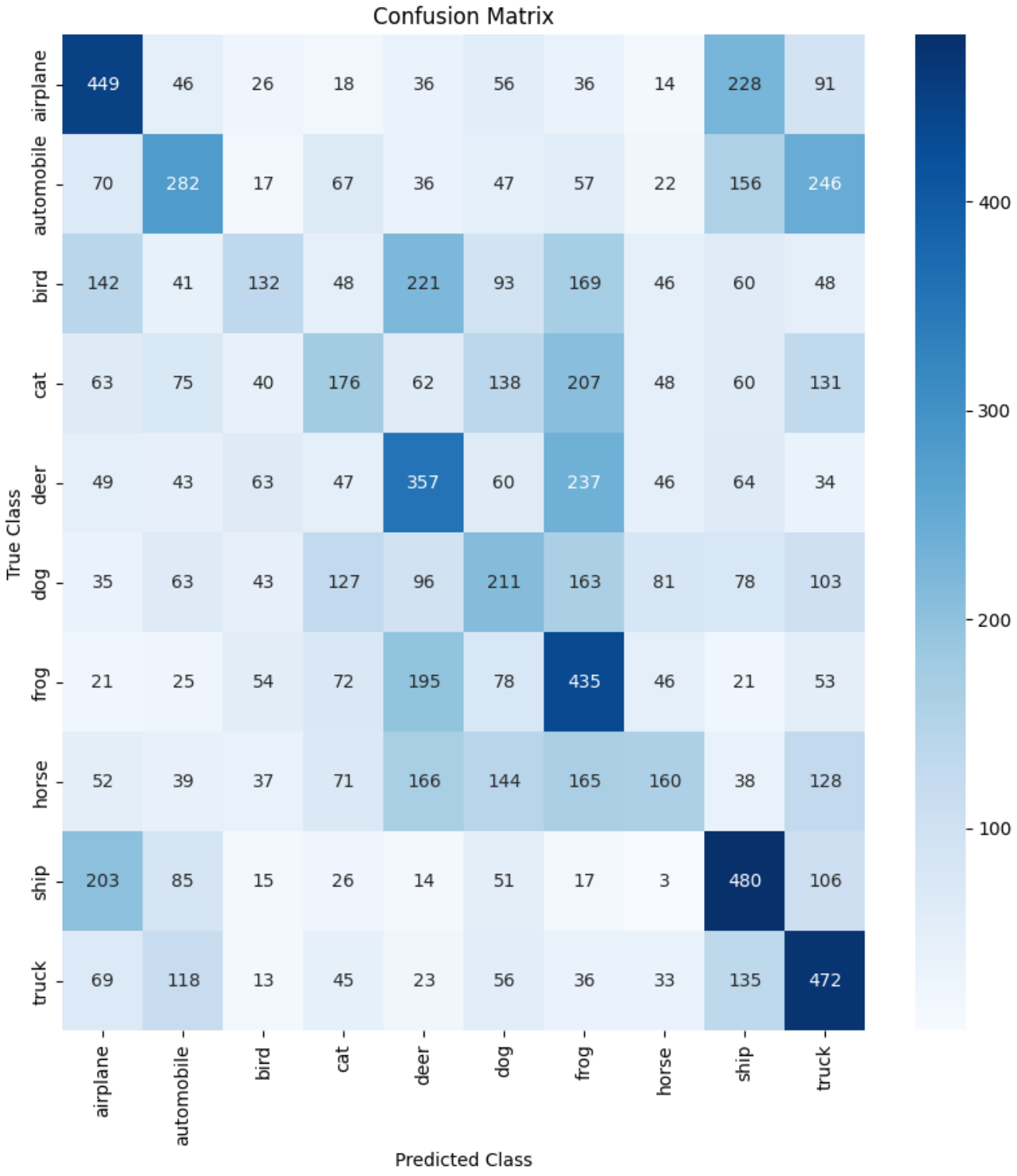
— loss : log

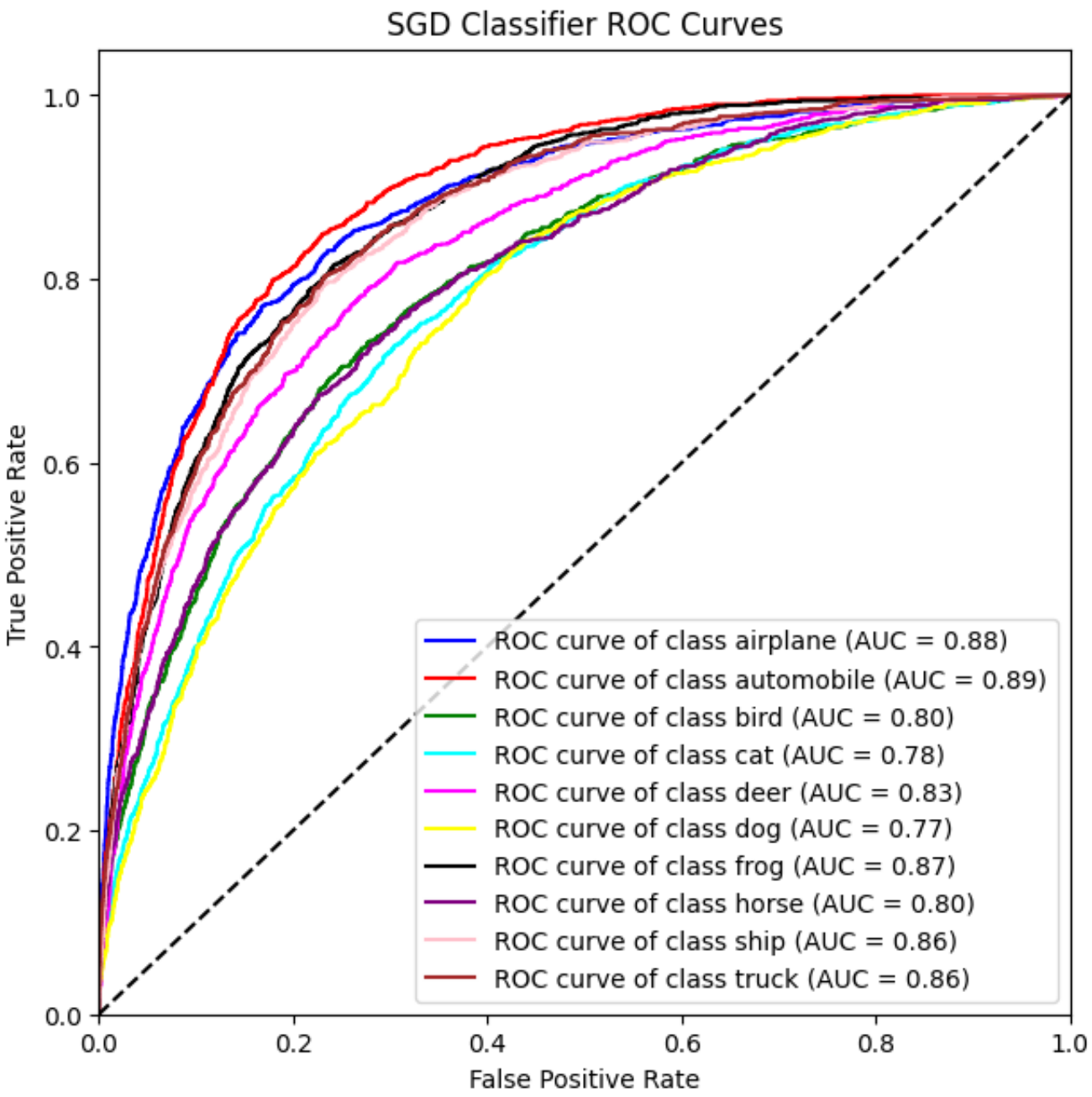
— alpha : 0,001

— penalty : L2

6.3.2 Précision obtenue et visualisation

Pour ce modèle, nous avons obtenue une précision de 31,54%, avec la matrice de confusion et les courbes ROC suivantes :


Confusion Matrix
True Class
Predicted Class
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
449 46 26 18 36 56 36 14 228 91
70 282 17 67 36 47 57 22 156 246
142 41 132 48 221 93 169 46 60 48
63 75 40 176 62 138 207 48 60 131
49 43 63 47 357 60 237 46 64 34
35 63 43 127 96 211 163 81 78 103
21 25 54 72 195 78 435 46 21 53
52 39 37 71 166 144 165 160 38 128
203 85 15 26 14 51 17 3 480 106
69 118 13 45 23 56 36 33 135 472
400
300
200
100



6.4 Interprétation des résultats

Les résultats obtenus ont montré que nos modèles ont du mal à différencier certaines classes, comme le montrent les erreurs fréquentes dans la matrice de confusion.

En particulier, nous avons constaté que nos modèles confondent souvent les bateaux et les avions, ainsi que les avions et les oiseaux. Il y a également une confusion notable entre les camions et les voitures. Cette difficulté à différencier ces paires spécifiques peut être attribuée à plusieurs facteurs potentiels.

Un facteur clé que nous avons identifié est notre méthode d'extraction de caractéristiques, qui se concentre sur les couleurs présentes dans les images. Les bateaux et les avions, par exemple, apparaissent souvent en bleu dans les images en raison de leur association avec le ciel ou l'eau. De même, les avions et les oiseaux peuvent tous deux être associés au ciel bleu, ce qui peut expliquer pourquoi ils sont fréquemment confondus. Pour les camions et les voitures, ils partagent souvent des couleurs similaires en raison de leur contexte similaire et de leur fabrication.

Il est clair que bien que l'extraction de la couleur soit une caractéristique utile, elle n'est pas suffisante pour différencier de manière fiable ces classes d'images. Les formes, les textures et le contexte spatial sont d'autres caractéristiques visuelles importantes qui pourraient aider à améliorer la précision des différents modèles.

7 Extraction HOG (Histogram of Gradients)

La méthode Histogram of Oriented Gradients est une technique qui compte le nombre d'occurrences de dégradés d'orientation d'image (directions de plus grand changement d'intensité lumineuse) dans des parties localisées de l'image. Les étapes du calcul des caractéristiques HOG sont les suivantes :

Normalisation de l'image : Avant de calculer les gradients, l'image est généralement normalisée pour réduire les effets d'éclairage.

Calcul du Gradient d'Image : La première étape du calcul HOG consiste à calculer le gradient d'image. Cela signifie que pour chaque pixel, nous voulons regarder les pixels qui l'entourent et trouver la direction de plus grand changement d'intensité lumineuse (ou de couleur).

Encodage des gradients : Les gradients calculés sont ensuite encodés pour chaque pixel. L'encodage le plus courant est l'encodage de l'orientation du gradient et de l'amplitude (ou de la magnitude).

Création des histogrammes orientés : L'image est divisée en petites cellules (8x8 pixels chacune dans notre cas), et pour les pixels dans chaque cellule, un histogramme de directions de gradient est compilé.

Normalisation : Les histogrammes de cellules sont ensuite groupés en blocs plus grands (1 bloc = 2x2 cellules dans notre cas) et les histogrammes dans chaque bloc sont normalisés. La normalisation réduit l'effet des variations d'éclairage.

Création du descripteur HOG : Le descripteur HOG final de l'image est alors le vecteur de tous les histogrammes normalisés des blocs de l'image.

A noter donc que, dans notre cas, comme on travaille avec des images de 32x32 pixels, une image contient (32/8)x(32/8) = 4x4 = 16 cellules. Et le nombre total de blocs est (4/2)x(4/2) = 2x2 = 4 blocs, puisque chaque bloc contient 2x2 cellules.

7.1 Random Forest

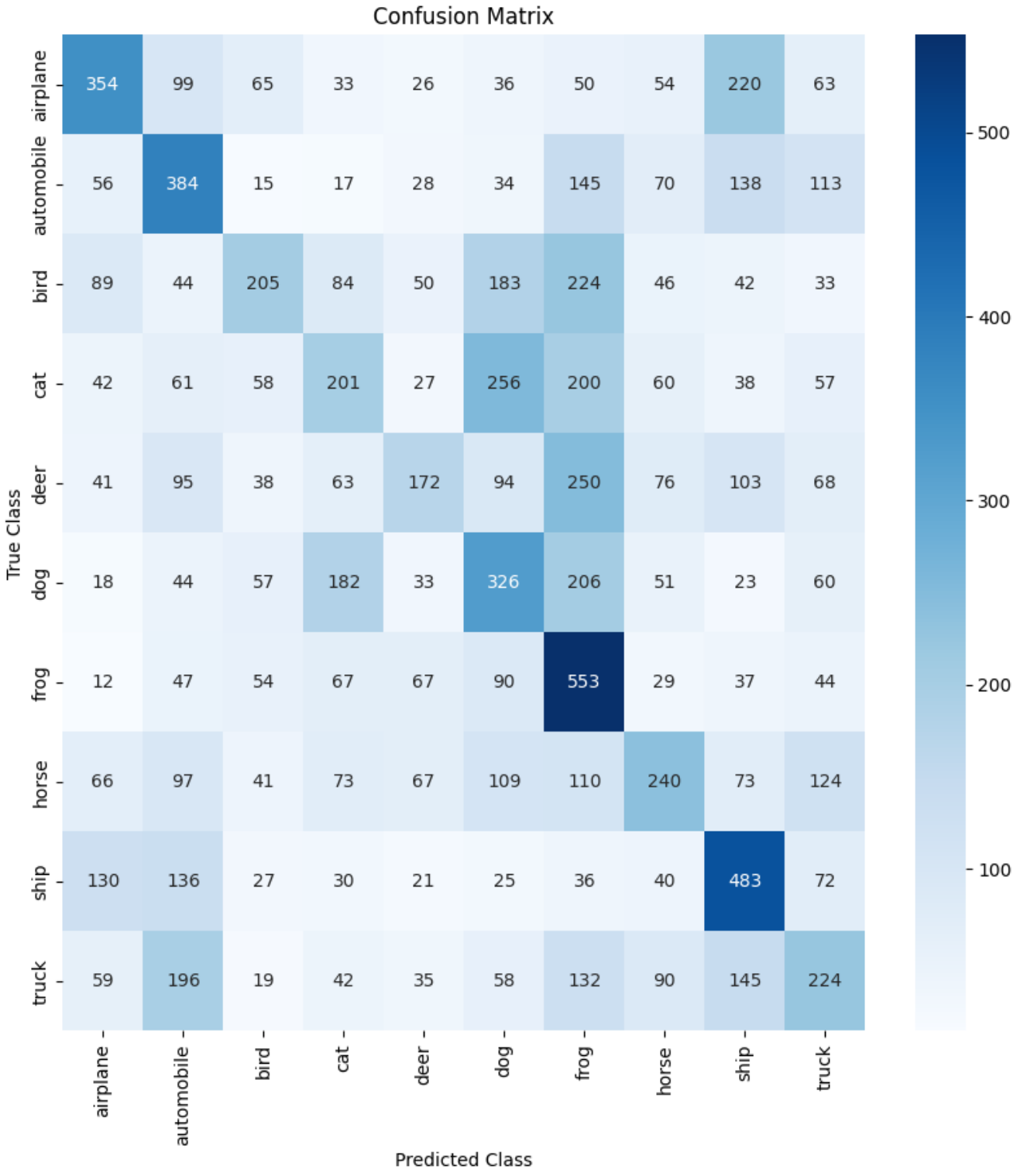
7.1.1 Recherche des hyperparamètres

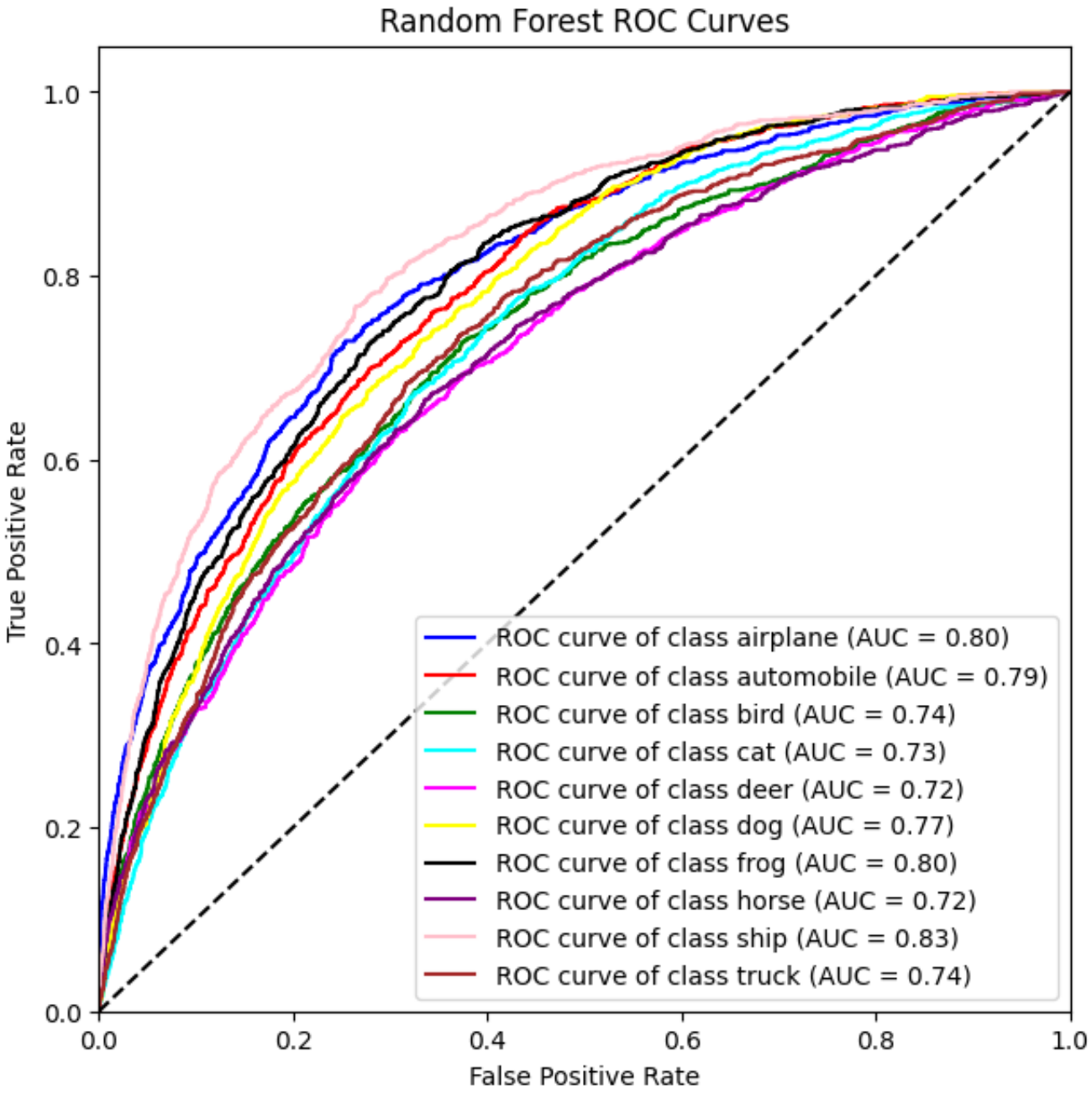
Avec les mêmes choix de valeurs que pour le précédent modèle Random Forest, nous avons obtenue la meilleure combinaison suivante :

- 'n_estimators' : 100
- 'max_depth' : 20
- 'min_samples_split' : 10
- 'min_samples_leaf' : 2

7.1.2 Précision obtenue et visualisation

Pour ce modèle, nous avons obtenue une précision de 31,42%, avec la matrice de confusion et les courbes ROC suivantes :


Confusion Matrix
True Class
Predicted Class
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
354 99 65 33 26 36 50 54 220 63
56 384 15 17 28 34 145 70 138 113
89 44 205 84 50 183 224 46 42 33
42 61 58 201 27 256 200 60 38 57
41 95 38 63 172 94 250 76 103 68
18 44 57 182 33 326 206 51 23 60
12 47 54 67 67 90 553 29 37 44
66 97 41 73 67 109 110 240 73 124
130 136 27 30 21 25 36 40 483 72
59 196 19 42 35 58 132 90 145 224
500
400
300
200
100



7.2 Régression logistique

7.2.1 Recherche des hyperparamètres

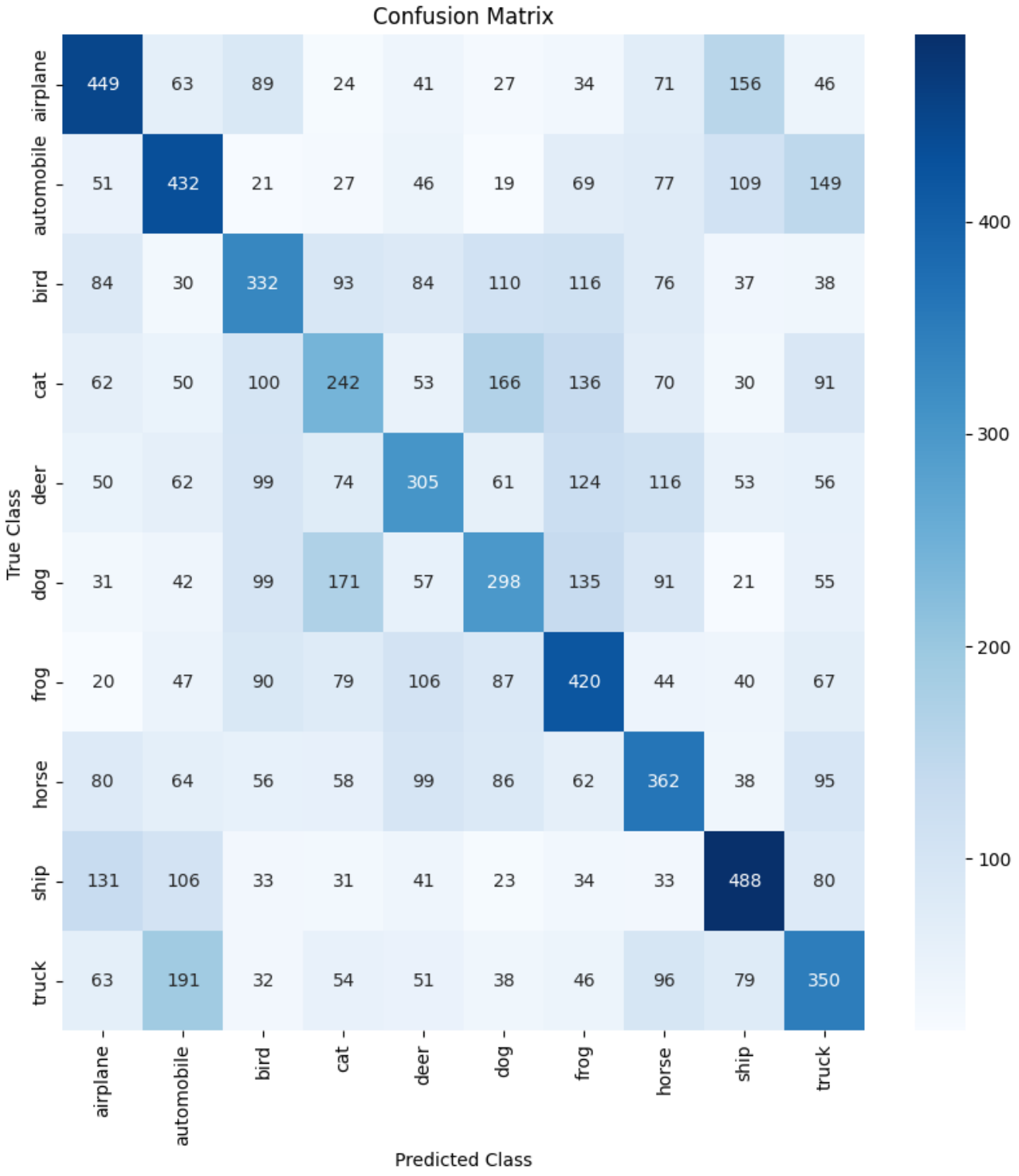
Avec les mêmes choix de valeurs que pour le précédent modèle de régression logistique, nous avons obtenue la meilleure combinaison suivante :

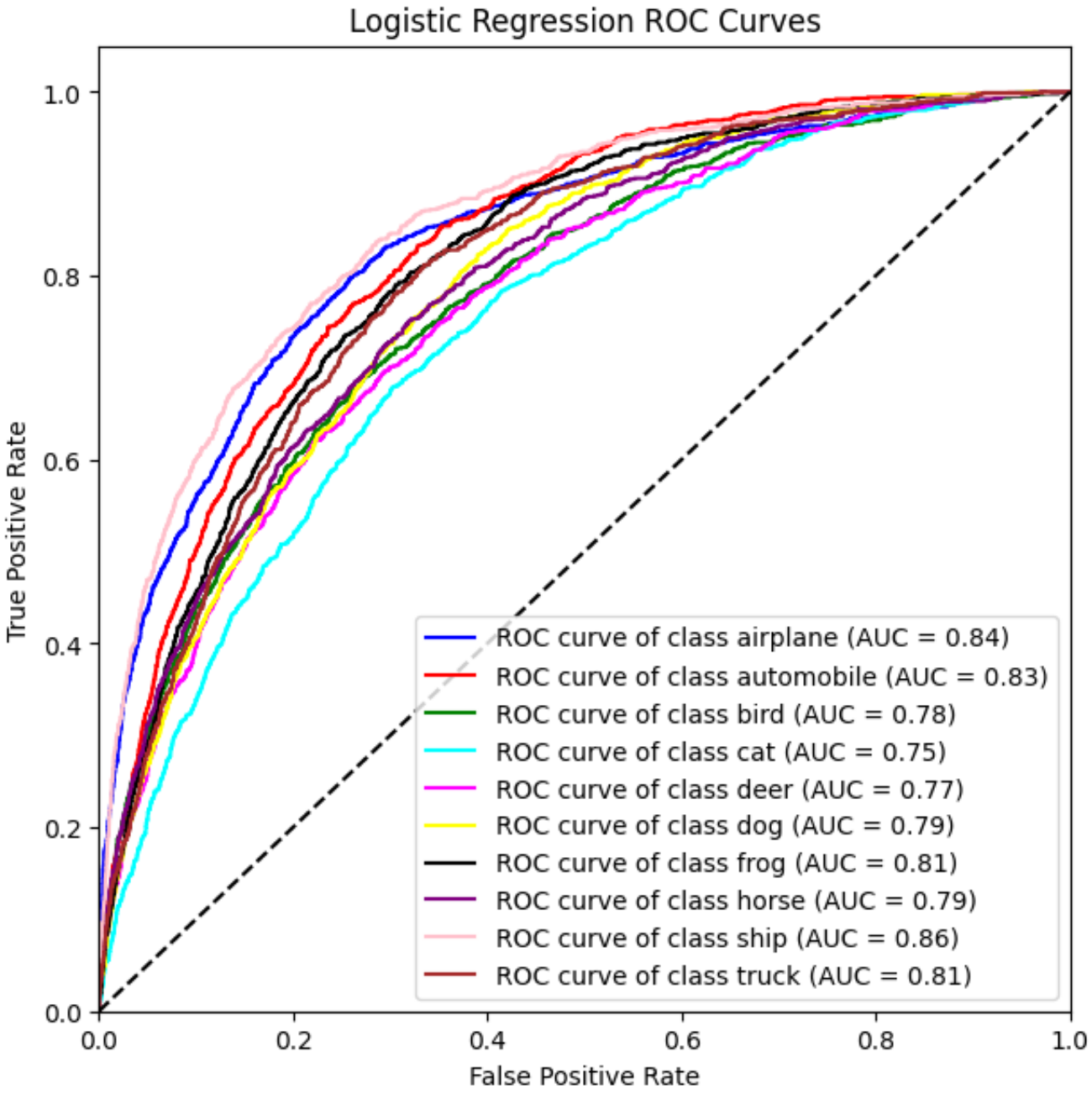
— C : O,1

— penalty : L2

7.2.2 Précision obtenue et visualisation

Pour ce modèle, nous avons obtenue une précision de 36,78%, avec la matrice de confusion et les courbes ROC suivantes :


Confusion Matrix
True Class
Predicted Class
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
449 63 89 24 41 27 34 71 156 46
51 432 21 27 46 19 69 77 109 149
84 30 332 93 84 110 116 76 37 38
62 50 100 242 53 166 136 70 30 91
50 62 99 74 305 61 124 116 53 56
31 42 99 171 57 298 135 91 21 55
20 47 90 79 106 87 420 44 40 67
80 64 56 58 99 86 62 362 38 95
131 106 33 31 41 23 34 33 488 80
63 191 32 54 51 38 46 96 79 350
400
300
200
100



7.3 SGD Classifier

7.3.1 Recherche des hyperparamètres

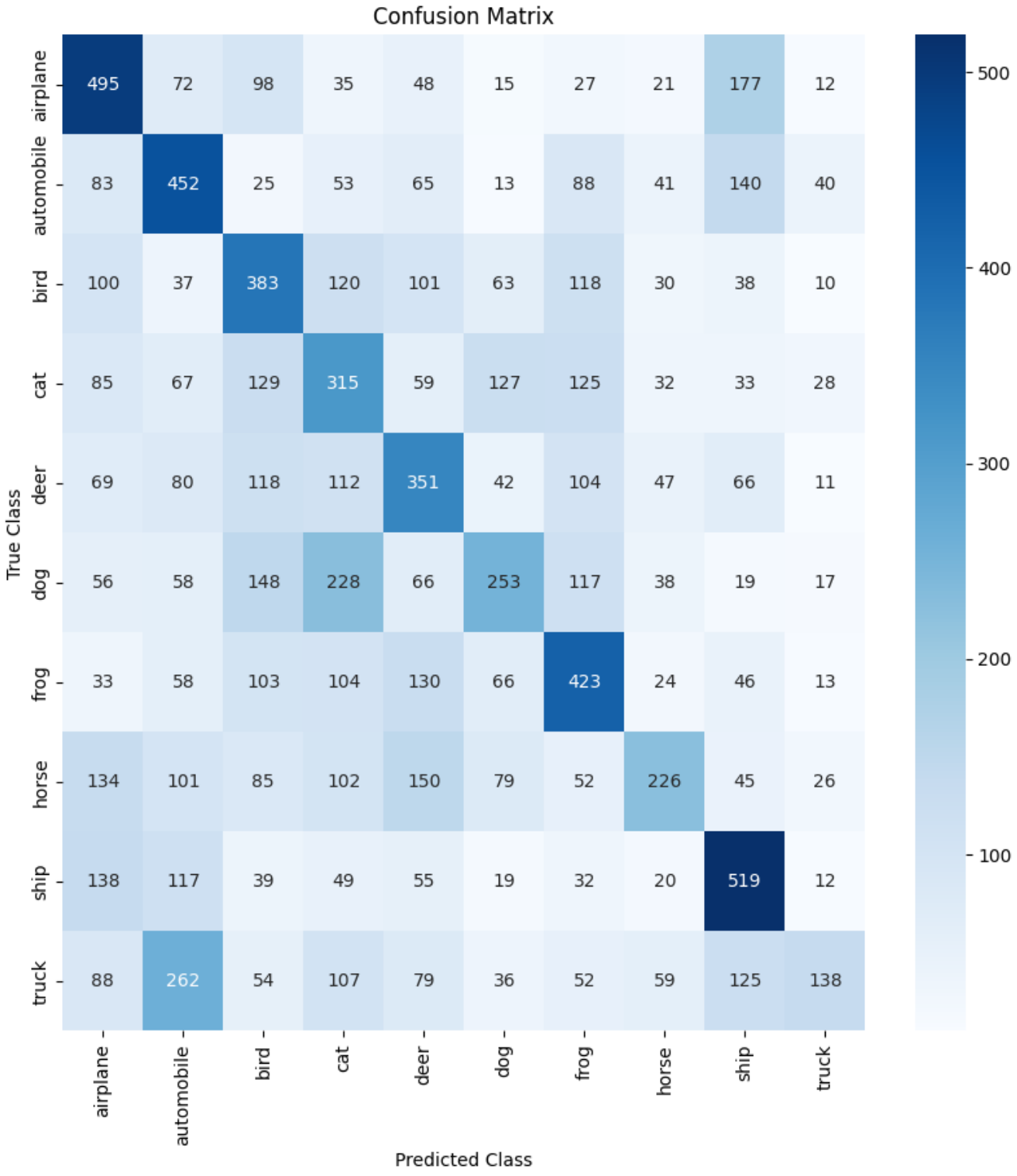
Avec les mêmes choix de valeurs que pour le précédent modèle SGDC, mais sans paramètre de pénalité, nous avons obtenue la meilleure combinaison suivante :

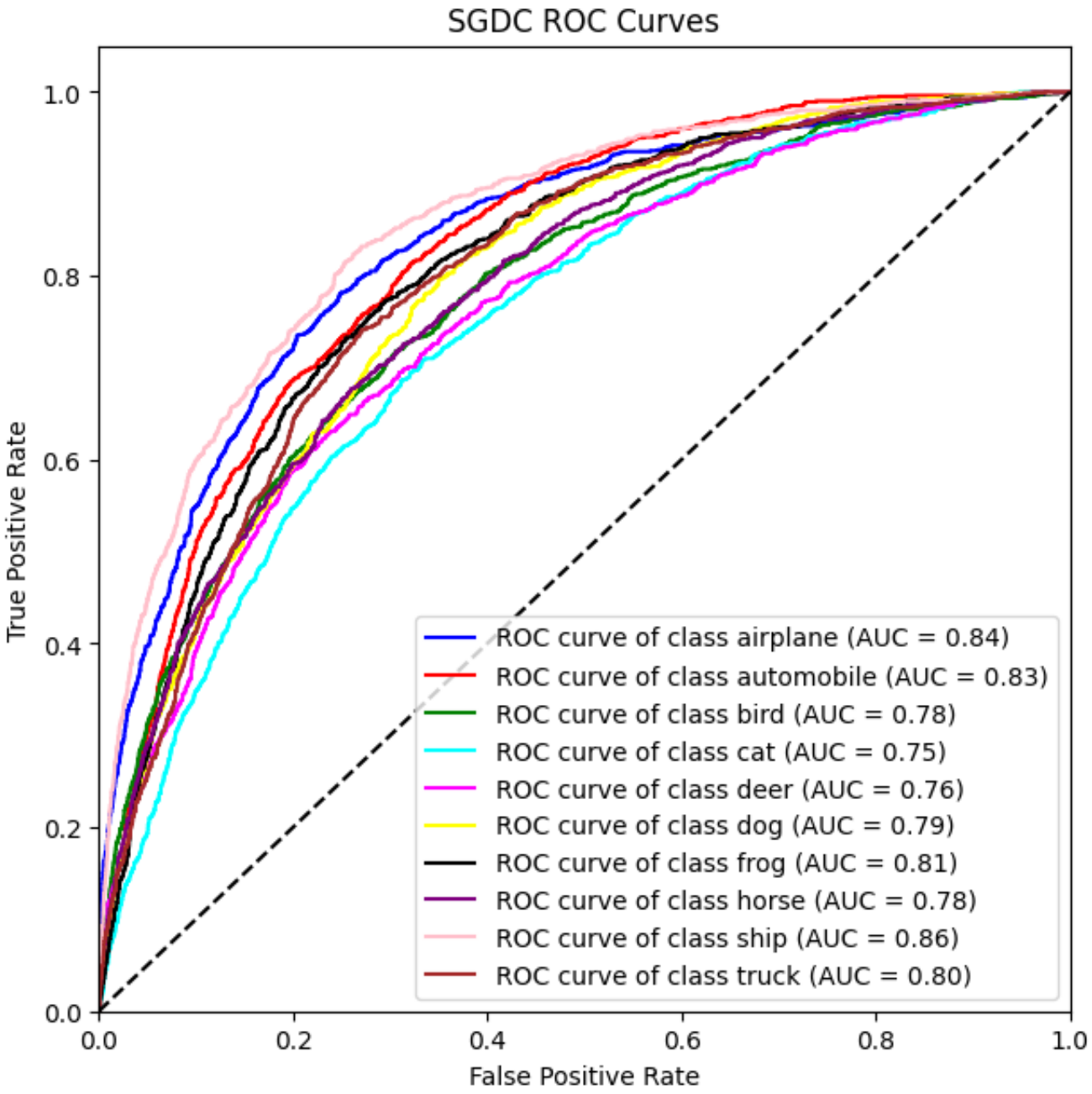
— loss : log

— alpha : 0,001

7.3.2 Précision obtenue et visualisation

Pour ce modèle, nous avons obtenue une précision de 35,55%, avec la matrice de confusion et les courbes ROC suivantes :


Confusion Matrix
True Class
Predicted Class
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
495 72 98 35 48 15 27 21 177 12
83 452 25 53 65 13 88 41 140 40
100 37 383 120 101 63 118 30 38 10
85 67 129 315 59 127 125 32 33 28
69 80 118 112 351 42 104 47 66 11
56 58 148 228 66 253 117 38 19 17
33 58 103 104 130 66 423 24 46 13
134 101 85 102 150 79 52 226 45 26
138 117 39 49 55 19 32 20 519 12
88 262 54 107 79 36 52 59 125 138
500
400
300
200
100



7.4 Interprétation des résultats

On remarque des résultats assez similaires que précedemment pour la régression logistique et le SGDC, avec des confusions similaires qui pourraient s'expliquer de la même manière, à la différence qu'ici, la similarité entre avion et bateau ne se fait pas sur les couleur mais sur la variation d'intensité de couleurs qui est proche. En revanche, on observe une chute de précision d'environ 10% pour le modèle Random Forest que nous n'avons pas été capable d'expliquer.

8 Extraction sous forme de vecteurs aplatis

Ici, il ne s'agit en réalité pas d'une extraction à proprement parler puisqu'on utilise la données brutes que l'on a converti sous forme de vecteur.

8.1 Random Forest

8.1.1 Recherche des hyperparamètres

Avec les mêmes choix de valeurs que pour les précédents modèles Random Forest, nous avons obtenue la meilleure combinaison suivante :

— 'n_estimators' : 100

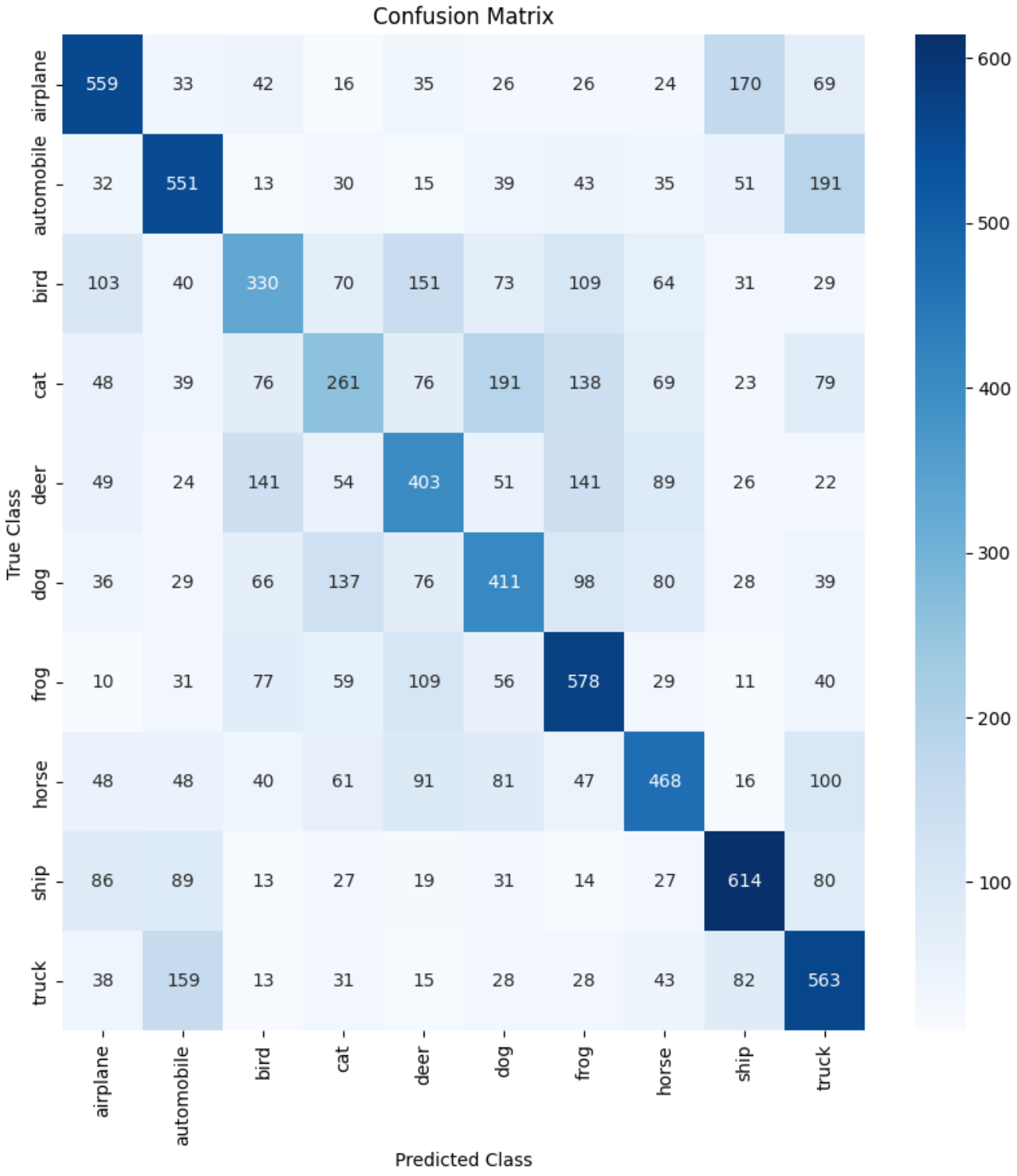
— 'max_depth' : None

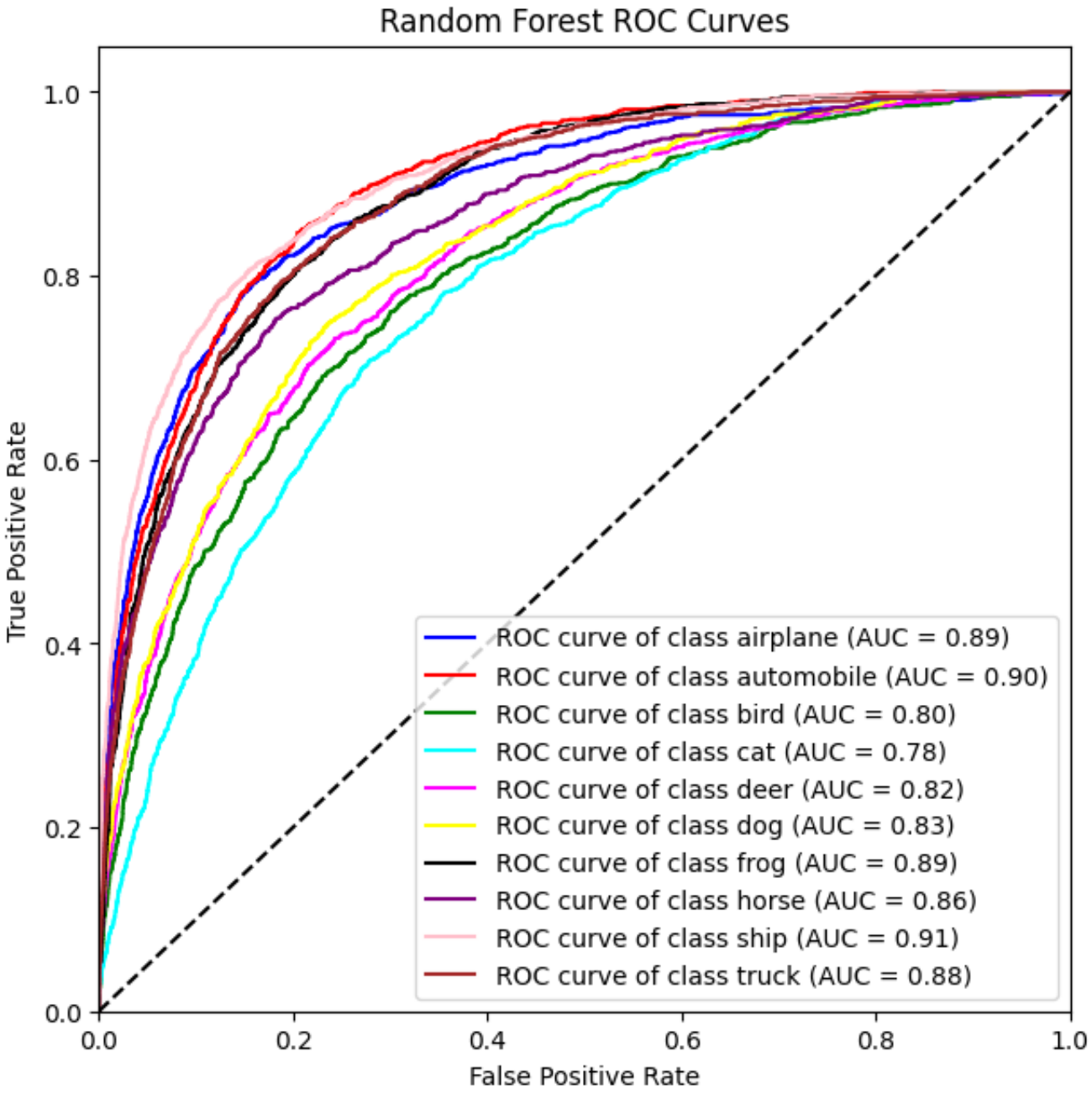
— 'min_samples_split' : 5

— 'min_samples_leaf' : 2

8.1.2 Précision obtenue et visualisation

Pour ce modèle, nous avons obtenue une précision de 47,38%, avec la matrice de confusion et les courbes ROC suivantes :


Confusion Matrix
True Class
Predicted Class
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
559 33 42 16 35 26 26 24 170 69
32 551 13 30 15 39 43 35 51 191
103 40 330 70 151 73 109 64 31 29
48 39 76 261 76 191 138 69 23 79
49 24 141 54 403 51 141 89 26 22
36 29 66 137 76 411 98 80 28 39
10 31 77 59 109 56 578 29 11 40
48 48 40 61 91 81 47 468 16 100
86 89 13 27 19 31 14 27 614 80
38 159 13 31 15 28 28 43 82 563
600
500
400
300
200
100



8.2 Régression logistique

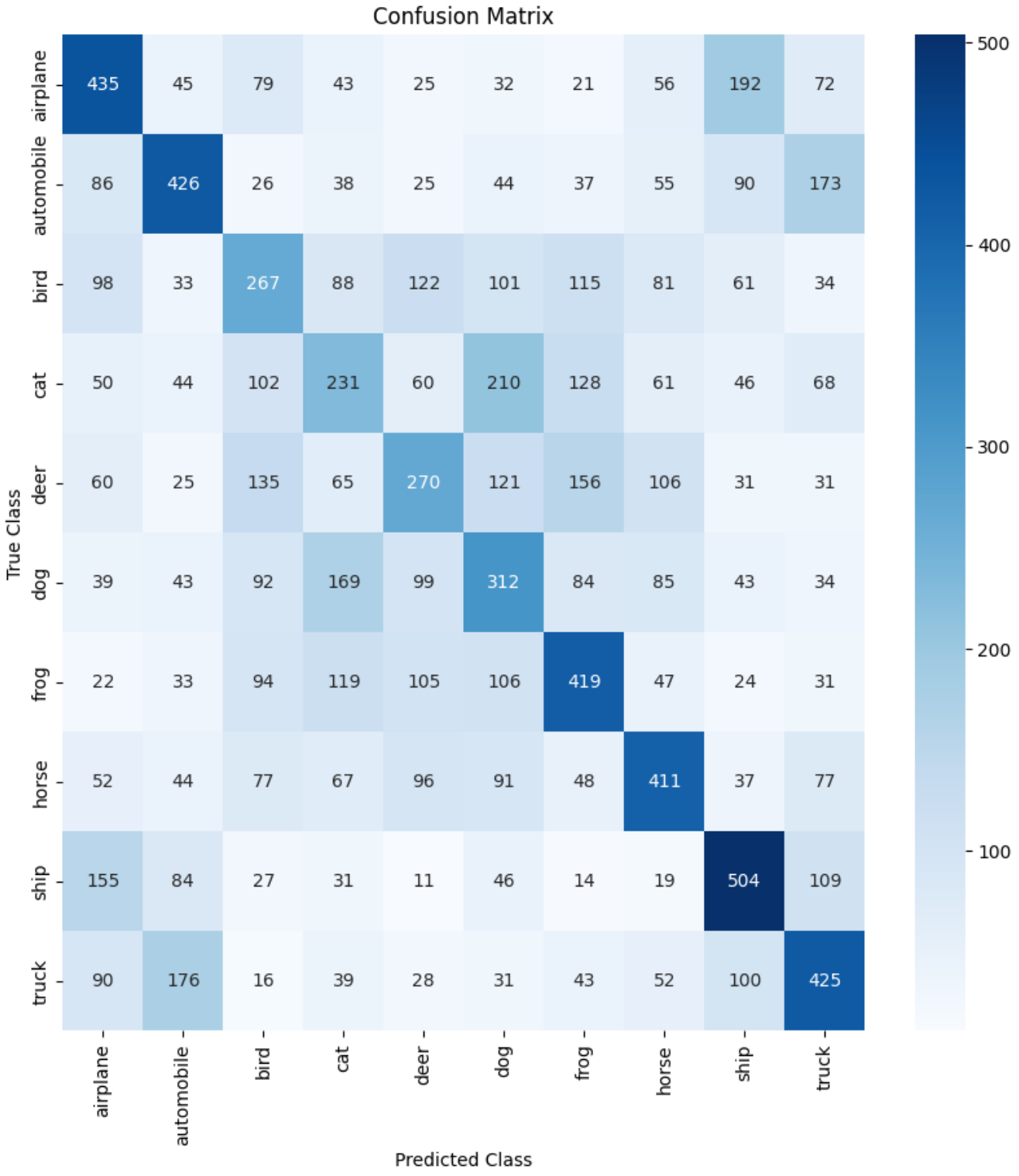
8.2.1 Recherche des hyperparamètres

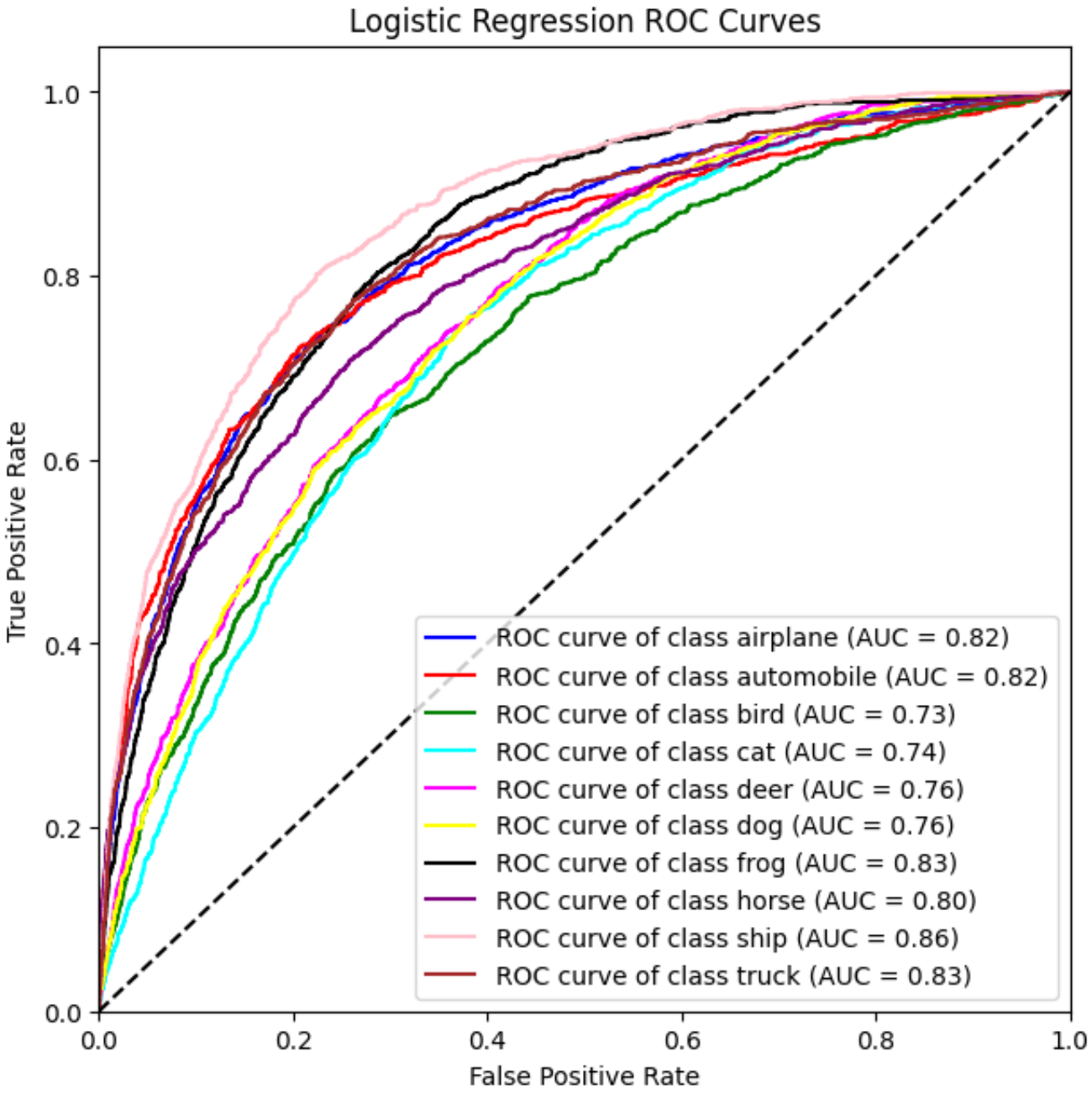
Avec les mêmes choix de valeurs que pour les précédents modèles de régression logistique, mais cette fois-ci sans pénalité, nous avons obtenue la meilleure combinaison suivante :

— C : O,1

8.2.2 Précision obtenue et visualisation

Pour ce modèle, nous avons obtenue une précision de 37%, avec la matrice de confusion et les courbes ROC suivantes :


Confusion Matrix
True Class
Predicted Class
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
435 45 79 43 25 32 21 56 192 72
86 426 26 38 25 44 37 55 90 173
98 33 267 88 122 101 115 81 61 34
50 44 102 231 60 210 128 61 46 68
60 25 135 65 270 121 156 106 31 31
39 43 92 169 99 312 84 85 43 34
22 33 94 119 105 106 419 47 24 31
52 44 77 67 96 91 48 411 37 77
155 84 27 31 11 46 14 19 504 109
90 176 16 39 28 31 43 52 100 425
500
400
300
200
100



8.3 SGD Classifier

8.3.1 Recherche des hyperparamètres

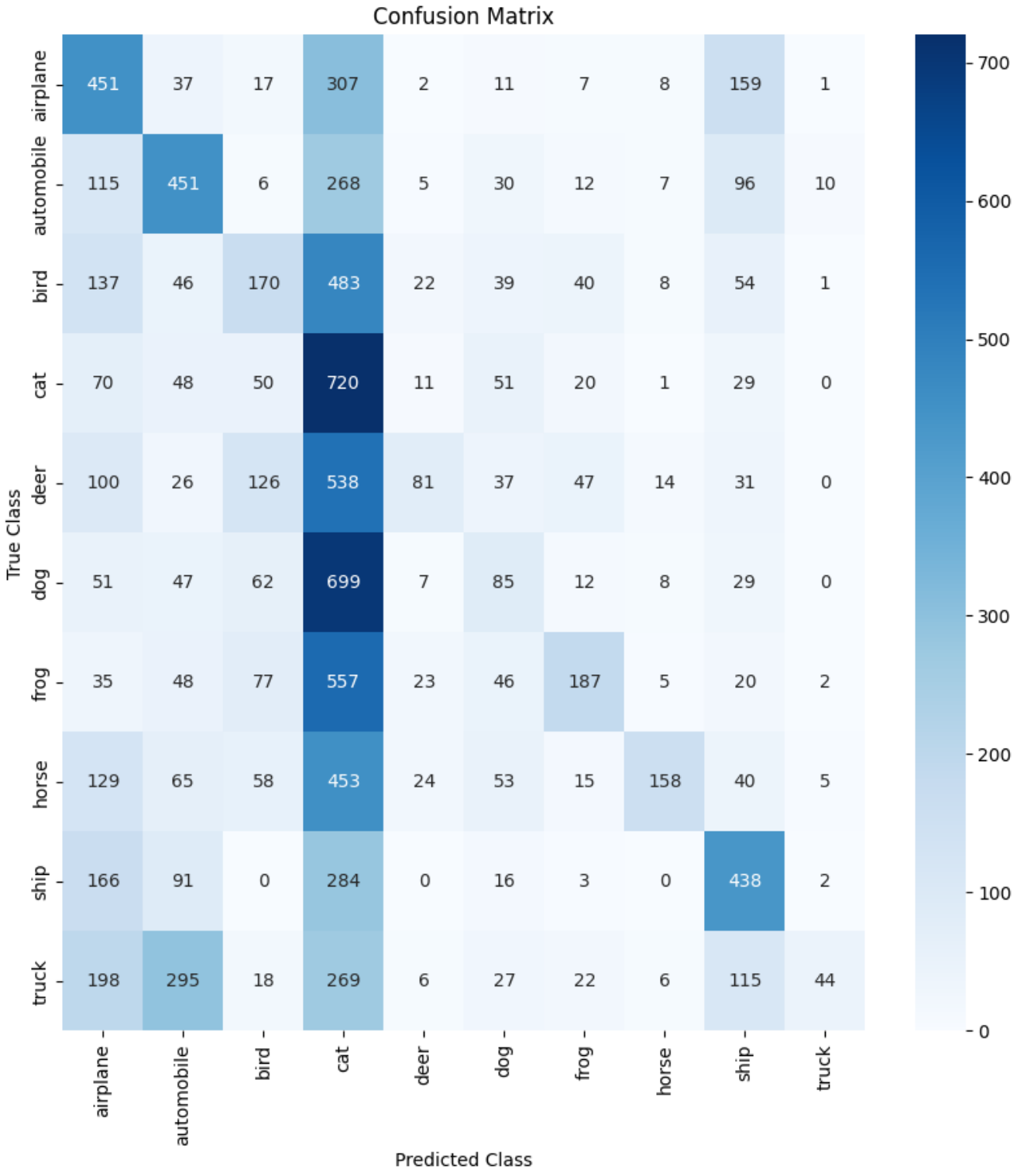
Avec les mêmes choix de valeurs que pour le précédent modèle SGDC, nous avons obtenue la meilleure combinaison suivante :

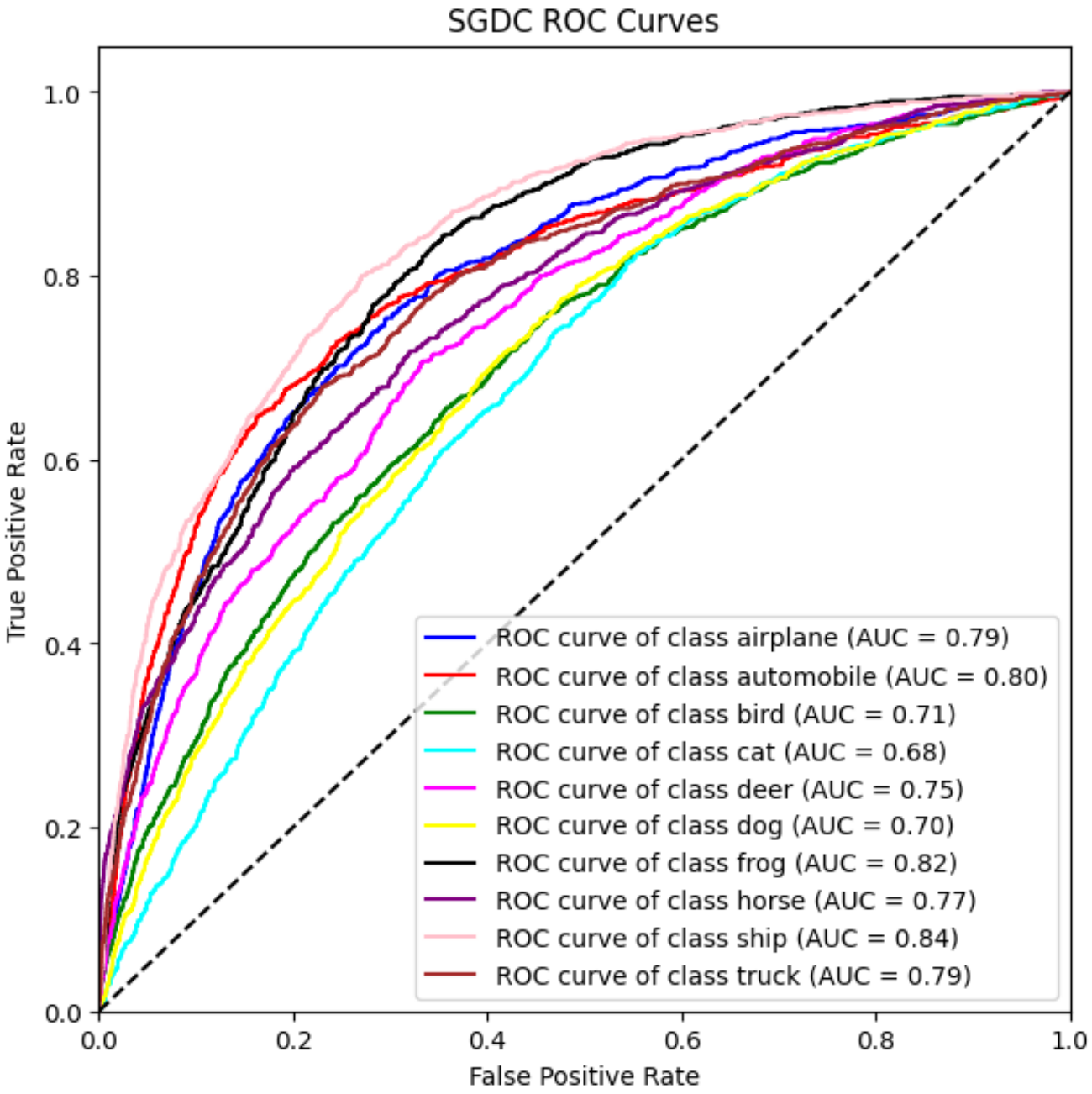
— loss : log

— alpha : 1

8.3.2 Précision obtenue et visualisation

Pour ce modèle, nous avons obtenue une précision de 27,85%, avec la matrice de confusion et les courbes ROC suivantes :


Confusion Matrix
True Class
Predicted Class
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
451 37 17 307 2 11 7 8 159 1
115 451 6 268 5 30 12 7 96 10
137 46 170 483 22 39 40 8 54 1
70 48 50 720 11 51 20 1 29 0
100 26 126 538 81 37 47 14 31 0
51 47 62 699 7 85 12 8 29 0
35 48 77 557 23 46 187 5 20 2
129 65 58 453 24 53 15 158 40 5
166 91 0 284 0 16 3 0 438 2
198 295 18 269 6 27 22 6 115 44
700
600
500
400
300
200
100
0



8.4 Interprétation des résultats

On remarque des résultats assez similaires que précedemment pour la régression logistique. En revanche, on observe une grosse perte de précision pour SGDC, et, en observant la matrice de confusion, on s'aperçoit que le modèle prédit très souvent la classe "chat", peu importe la classe à prédire, ce qui se remarque aussi sur la courbe ROC. D'autre part, on obtient un réel gain de précision avec le modèle Random Forest.

9 Résumé des précisions

	HSV	HOG	Donnée brute
Random Forest	41,72%	31,42%	47,38%
Régression logistique	34,63%	36,78%	37%
SGD Classifier	31,54%	35,55%	Cell 27,85%

10 Conclusion

En conclusion, l'étude a permis d'évaluer plusieurs modèles de classification d'images pour le jeu de données CIFAR-10, en utilisant diverses techniques d'extraction de caractéristiques : les données brutes, l'histogramme des teintes saturées en valeurs (HSV) et l'histogramme des gradients orientés (HOG). Trois types de modèles de machine learning ont été examinés : Random Forest, Régression logistique One VS All et SGD Classifier.

De manière surprenante, le modèle qui a produit la meilleure performance est celui qui a utilisé les données brutes, sans aucun prétraitement. En particulier, le modèle Random Forest a atteint une précision de 47,38%, nettement supérieure à celle obtenue avec les autres méthodes de prétraitement et modèles, qui ont généralement atteint une précision d'environ 30 à 35%.

Cela suggère que, dans ce cas particulier, les méthodes de prétraitement de données HSV et HOG n'ont pas réussi à capturer des caractéristiques significatives des images pour améliorer les performances de la classification. Il est possible que les modèles bénéficieraient de méthodes de prétraitement ou d'extraction de caractéristiques différentes, ou peut-être d'une combinaison de plusieurs méthodes.

Il convient également de noter que bien que la performance du modèle Random Forest utilisant des données brutes soit la plus élevée parmi les modèles testés, une précision de 47,38% est encore relativement faible, indiquant que la tâche de classification est complexe et qu'il reste une marge d'amélioration significative.

Enfin, cette étude montre l'importance de l'expérimentation en apprentissage automatique. Il n'est pas toujours évident à l'avance quelles techniques fonctionneront le mieux pour un problème donné, et souvent, des approches simples peuvent surpasser des méthodes plus sophistiquées, ce qui n'est au premier abord pas évident. Il est donc important d'explorer une variété de techniques et d'approches pour chaque tâche.