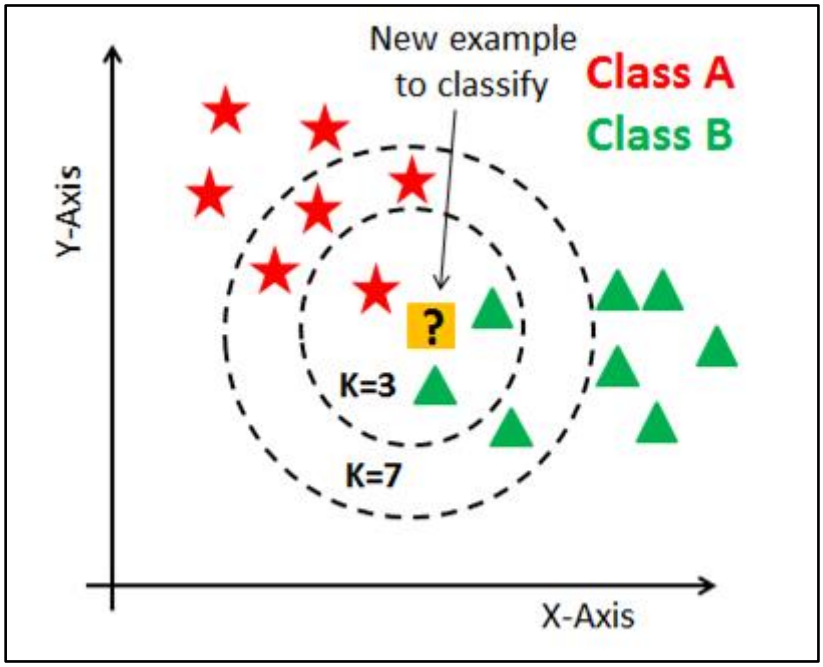
Programmation Orientée Objet

Algorithme des K plus proches voisins (Kppv)

Joshua BRUN & Pierre NICOLAS



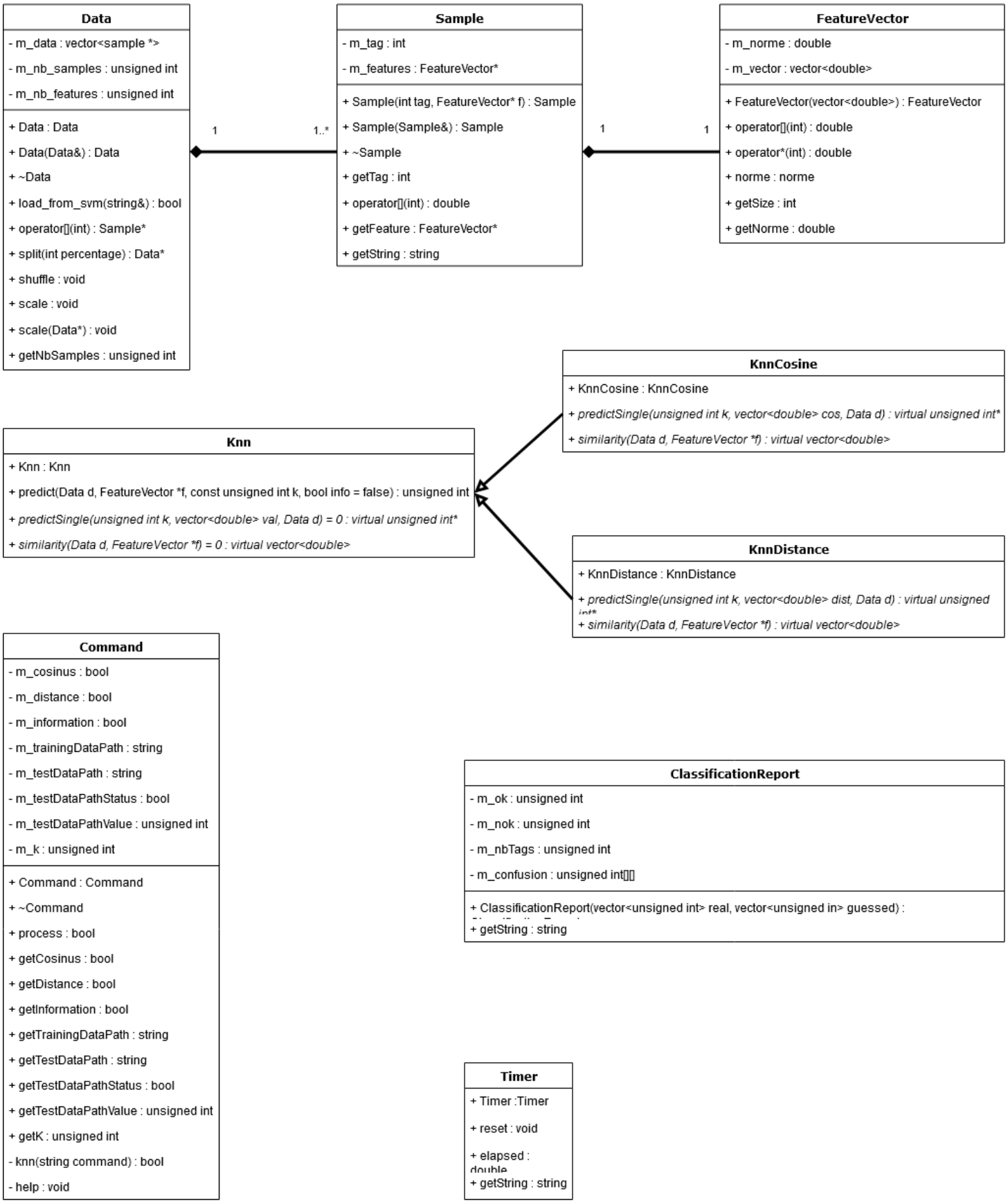
Introduction

Pour le projet, nous avons repris le diagramme de classe proposé en ajoutant des classes. Par exemple, une classe KnnDistance pour effectuer un autre type de calcul du Kppv ou une classe Command pour réaliser l'IHM (Interface Homme Machine) en ligne de commande.

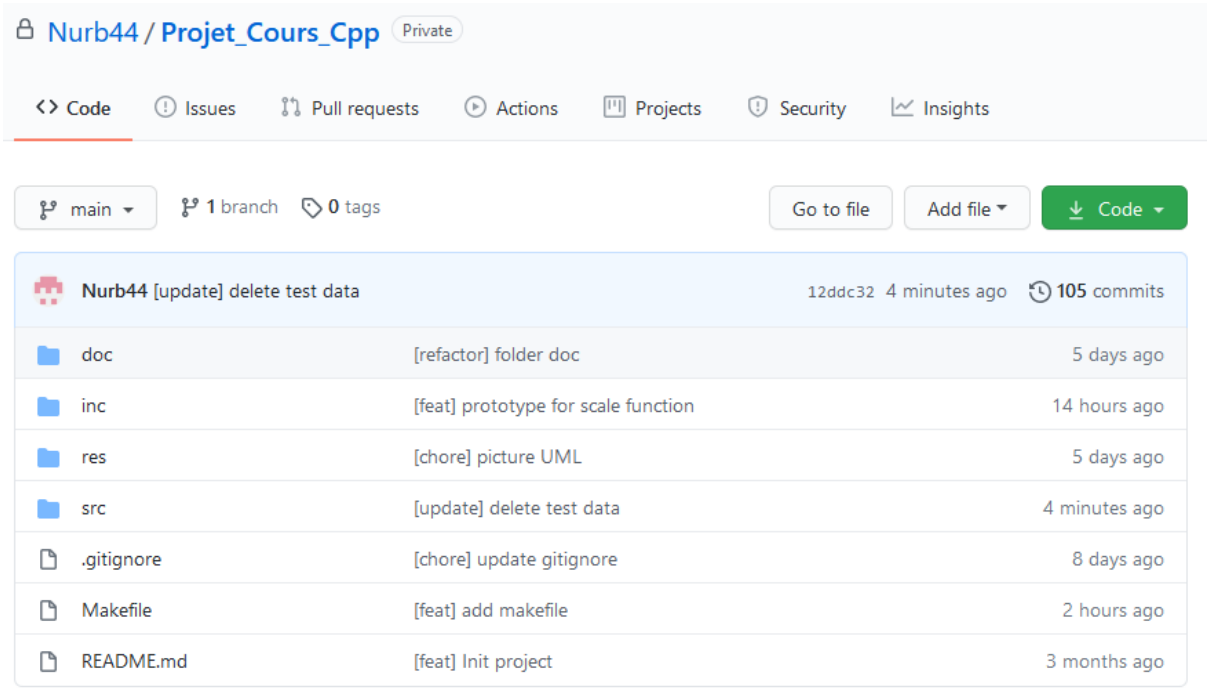
Nous nous sommes répartis les tâches en fonction de nos préférences, Pierre a réalisé les classes : Data, Sample, FeatureVector, ClassificationReport et Timer et Joshua les classes : Knn, KnnCosine, KnnDistance, Command et le main.

Pour réaliser le projet, nous avons créé un projet GIT sous GitHub, ainsi nous avons pu travailler en équipe sans difficulté.

Diagramme de classe



Utilisation de Git pour le projet



Utilisation du gestionnaire de version Git avec GitHub pour pouvoir avancer chacun de son côté et mettre en commun facilement.

Comparaison similarité cosinus et distance

Mise à l'échelle

Échantillon d'apprentissage : 900
Échantillon à deviner : 100
Nombre de test : 5000 indépendants (500 par valeur de k)

Le scale permet de recentrer les valeurs entre 0 et 1 pour éviter une disproportionnalité entre les différentes features. Le test ci-dessous, montre l'avantage d'utiliser le scale en montrant la précision moyenne gagnée.

k	1	2	3	4	5	6	7	8	9	10
Δ%	2.02	1.61	2.00	0.43	1.18	1.45	2.02	1.74	1.95	1.99

On gagne donc en moyenne presque 2% de précision. Il est donc très intéressant d'utiliser la mise à l'échelle.

Similarité cosinus

Échantillon d'apprentissage : 900
Échantillon à deviner : 100
Nombre de test : 80 indépendants (80 par valeur de k)
Base de donnée : Mise à l'échelle
Métjode : KnnCosine

Le test ci-dessous montre l'efficacité de l'algorithme KnnCosine en fonction du k utilisé.

k	1	2	3	4	5	6	7	8	9	10
%	93.28	92.37	93.40	93.03	93.66	92.58	93.42	92.76	92.75	92.10

Avec une base de données de 900 échantillons, le meilleur k à choisir pour l'algorithme KnnCosine est 5. On observe une baisse de la réussite sur des valeurs de k plus élevé.

Similarité distance

Échantillon d'apprentissage : 900
Échantillon à deviner : 100
Nombre de test : 80
Base de donnée : Mise à l'échelle
Méthode : KnnDistance

Le test ci-dessous montre l'efficacité de l'algorithme KnnDistance en fonction du k utilisé.

k	1	2	3	4	5	6	7	8	9	10
%	94.90	93.01	94.18	93.36	93.73	93.46	94.00	93.37	93.33	92.87

Avec une base de données de 900 échantillons, le meilleur k à choisir pour l'algorithme KnnDistance est 1. On observe une baisse de la réussite sur des valeurs de k plus élevé.

Conclusion

Grâce à la mise à l'échelle, nos prédictions ont de meilleurs taux de réussite, ce qui améliore notre programme. Cependant, les fonctions "scale" disponibles sont des prototypes, elles sont vouées à être améliorées.

Les deux types de calculs, similarité cosinus et distance, sont opérationnels. En testant avec un k entre 1 et 10, nous remarquons que le calcul de distance à un meilleur pourcentage de prédiction (94.90%) pour un k = 1. Dans les deux cas, la prédiction est moins précise lorsque k est plus élevé.

En amélioration, nous pourrions remplacer la classe Command par la librairie CLI11 qui permet d'avoir une interface en ligne de commandes et ainsi peaufiner notre projet. De plus, nous pourrions écrire les commentaires sous le format de Doxygen et ainsi générer automatiquement une documentation du code en HTML et le diagramme de classe. Nous pouvons également ajouter un fichier de log et pondérer le choix du tag en fonction du plus proche.

Capture de notre interface :

```
WELCOME TO THE KNN
If you need help, enter "help"
>>knn -cd "D:\Cours\C++\Projet_Cours_Cpp\res\digits.svm" "5.svm" 5

COSINUS
In 50 tests, 47 results are correct(94.00%), while 3 are incorrect(6.00%).
Here is confusion matrix (R:Real/G:Guessed) :
R/G     0       1       2       3       4       5       6       7       8       9
0 |     7       0       0       0       0       0       0       0       0       0
1 |     0       5       0       0       0       0       0       0       0       0
2 |     0       0       2       0       0       0       0       0       0       0
3 |     0       0       0       5       0       0       0       0       0       0
4 |     0       0       0       0       3       0       0       0       0       0
5 |     0       0       0       0       0       7       1       0       0       0
6 |     0       0       0       0       0       0       7       0       0       1
7 |     0       0       0       0       1       0       0       4       0       0
8 |     0       0       0       0       0       0       0       0       2       0
9 |     0       0       0       0       0       0       0       0       0       5


DISTANCE
In 50 tests, 46 results are correct(92.00%), while 4 are incorrect(8.00%).
Here is confusion matrix (R:Real/G:Guessed) :
R/G     0       1       2       3       4       5       6       7       8       9
0 |     7       0       0       0       0       0       0       0       0       0
1 |     0       5       0       0       0       0       0       0       0       0
2 |     0       0       2       0       0       0       0       0       0       0
3 |     0       0       0       4       0       0       0       1       0       0
4 |     0       0       0       0       3       0       0       0       0       0
5 |     0       0       0       0       0       7       1       0       0       0
6 |     0       0       0       0       0       0       7       0       0       1
7 |     0       0       0       0       1       0       0       4       0       0
8 |     0       0       0       0       0       0       0       0       2       0
9 |     0       0       0       0       0       0       0       0       0       5

Time to work: 10.631755s
>>
```