Institut National des Sciences Appliquées
de Rennes

TSExplanation
Rapport Final

Adrien Buridant
Yohan Couanon
Isabelle Guillou
Tangi Mendès
Lisa Relion

Responsables de projet :
Laurence Rozé (INSA, INRIA, IRISA)
Maël Guillemé (ENERGIENCY)

Septembre 2018 - Mai 2019
INSA de Rennes

Table des matières

Introduction

Ce rapport final vient clôturer le projet TSExplanation. Pour rappel, l'objectif de ce projet était de réaliser un interpréteur de classification de séries temporelles. Les détails de ce sujet ont été abordés dans les rapports de pré-étude, de spécification et de conception.
La Figure 1 rappelle le fonctionnement général de l'outil TSExplanation :

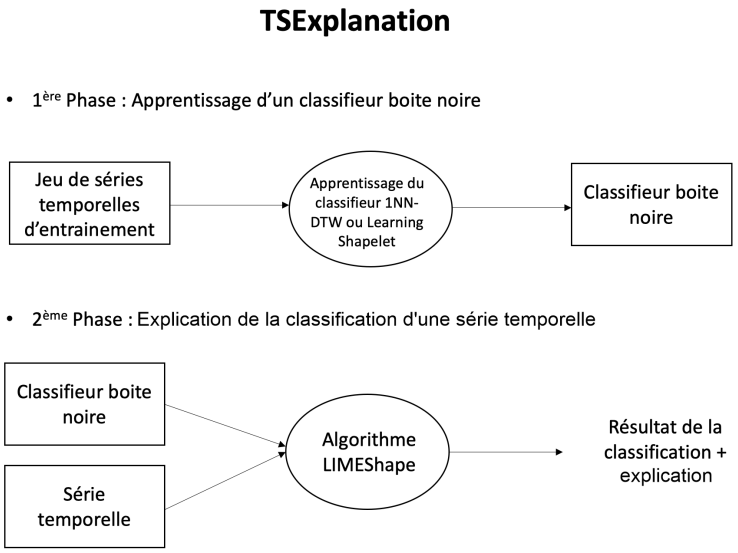


FIGURE 1 – Illustration du fonctionnement de TSExplanation

Afin de mener à bien ce projet, le travail à réaliser a été divisé en trois parties : l'apprentissage et la sauvegarde d'un classifieur de séries temporelles, l'adaptation de l'algorithme LIME pour les séries temporelles et enfin la réalisation d'une interface graphique pour faciliter le travail de l'utilisateur.

Dans ce rapport, nous présenterons l'outil TSExplanation et ce qui a été réalisé tout au long de ce projet. Nous ferons donc le point sur les avancées faites dans les trois parties majeures du projet (rappelées précédemment), en détaillant les points techniques les plus importants pour chacune d'entre elles ainsi que les tests qui ont été fait pour s'assurer du bon fonctionnement de l'outil. Nous ferons également un récapitulatif des changements effectués par rapport à ce qui avait été annoncé dans les rapports précédents, notamment les rapport de spécification et de conception. Un bilan sera fait sur le déroulement du projet et sur son adéquation avec ce qui avait été mis en place lors de la phase de planification. Enfin, ce rapport sera suivi par des annexes, composées de la documentation utilisateur de l'outil TSExplanation.

1 État de finalisation du projet

Dans cette partie, l'état d'avancement de chacune des trois parties du projet sera présenté. Pour chaque partie, les différents points techniques seront abordés et nous préciserons si les objectifs fixés ont été atteints et si certains points peuvent encore être améliorés.

Dans le rapport de planification, le découpage des tâches avait été présenté en détails. Voici un rappel de ce découpage :

- Générateur de classifieurs de série temporelle (entraînement et sauvegarde)
  - Classifieur 1NN
  - Classifieur Learning Shapelet
  - Sauvegarde
- Adaptation de LIME aux séries temporelles
  - Classe TimeSeriesDomainMapper
  - Classe IndexedTimeSeries
  - Classe TimeSeriesExplainer
- Interface Graphique
  - Agencement des composants graphiques
  - Gestion des évènements
  - Binding avec le Back-end
- Tests de validation

1.1 Apprentissage et sauvegarde d'un classifieur

La réalisation de cette partie du projet a consisté en l'implémentation de deux classes : l'une pour l'importation d'un jeu de séries temporelles d'entraînement et l'autre pour l'apprentissage d'un classifieur de séries temporelles à partir d'un jeu d'entraînement.
La Figure 2 présente le diagramme de classes qui avait été construit lors de la phase de conception. Les éléments en rouge sont ceux qui ont été supprimés et les éléments en vert sont ceux qui ont été ajoutés.

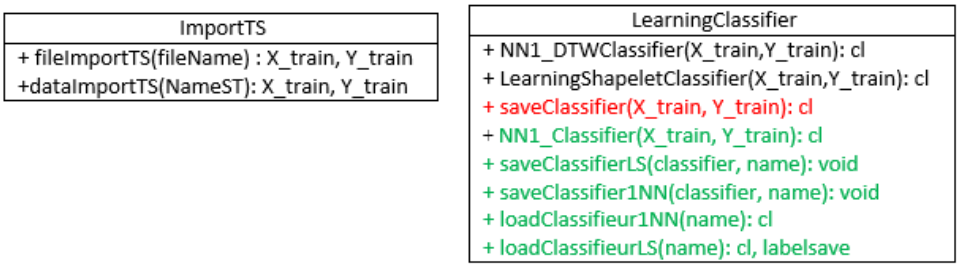


FIGURE 2 – Diagramme de classes pour les classifieurs

Ces deux classes ont bien été implémentées, et leurs différentes méthodes sont toutes opérationnelles. Avec TSExplanation, il est donc possible d'importer un jeu de séries temporelles depuis un fichier ou depuis la base de données UEA & UCR Time Series Classification Repository puis d'entraîner différents types de classifieurs de séries temporelles à partir de tels jeux de données.

1.2 LIME

Dans cette partie du projet, l'objectif était de reprendre l'algorithme LIME, un algorithme d'explication de classification d'images ou de texte, et de l'adapter pour pouvoir expliquer une classification de séries temporelles.

Dans le rapport de conception, il a été expliqué que deux classes allaient être récupérées depuis l'algorithme LIME, à savoir les classes *Lime_base* et *Explanation*. En revanche, trois classes devaient être entièrement implémentées afin d'adapter LIME aux séries temporelles :

— *TSDomainMapper*,
— *IndexedTS*,
— *TSExplainer*.

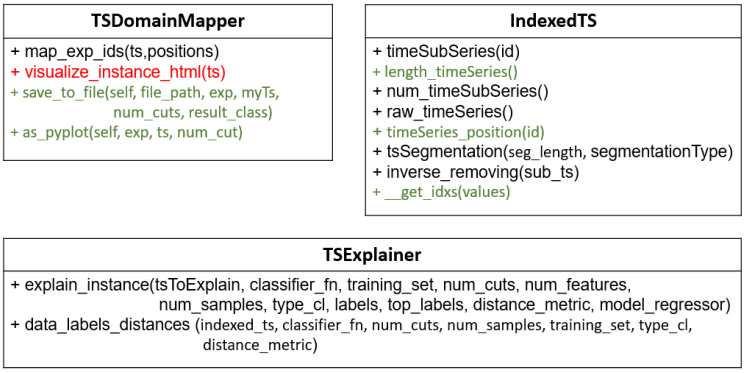


FIGURE 3 – Diagramme de classes de LIMEShape

Ces trois classes ont été implémentées, en subissant quelques modifications, comme le montre la Figure 3 et sont présentes dans le fichier *lime_timeseries.py*. Les éléments en rouge sont ceux qui ont été supprimés et les éléments en vert sont ceux qui ont été ajoutés.

1.3 Interface graphique

L'interface graphique du projet TSExplanation est composée de quatre parties distinctes. Son but est de faire le lien entre l'utilisateur et l'algorithme LIMEShape.

L'onglet "Classifieur" permet d'entraîner puis de sauvegarder un classifieur de type 1NN-DTW, Learning Shapelet ou 1NN avec la distance euclidienne. Cette partie fonctionne intégralement, même si le classifieur 1NN-DTW impose des temps de calculs très longs. Ce problème sera détaillé dans la partie *3.2 Changement de l'algorithme 1NN-DTW à 1NN distance euclidienne* de ce rapport.

L'onglet "ST" permet d'afficher une série temporelle donnée. Cette fonction a été réalisée assez tôt dans le projet, l'utilisateur peut donc afficher une série temporelle provenant d'un fichier ou de la base de séries temporelles de l'UCR.

L'onglet "Shapelet" permet d'afficher la distance entre une Shapelet et une série temporelle. Cette partie de l'application est également fonctionnelle. La fonction *minDistance* implémentée dans la classe principale de l'interface, comme le montre la Figure 10, renvoie la position où la distance entre les deux courbes est minimale. Ainsi il ne reste plus qu'à afficher la Shapelet à l'endroit associé sur la série temporelle.

L'onglet "LIME" est l'onglet principal de notre application. Il permet l'affichage de l'explication de la classification d'une série temporelle en faisant appel à l'algorithme LIMEShape que nous avons implémenté. Comme les trois autres, cette partie fonctionne très bien. L'utilisateur peut donc visualiser l'explication mais aussi la sauvegarder au format HTML.

1.4 Execution par ligne de commande

Certaines fonctionnalités de l'interface graphique sont accessibles par lignes de commande. L'apprentissage et la sauvegarde d'un classifieur ainsi que l'affichage de l'explication sont exécutables directement depuis l'invite de commande.

1.4.1 Commande : apprentissage et sauvegarde d'un classifieur

```
> python3 saveClassifier.py <inputFile> <classifierType> [-o OUTPUT] [-l]
```

Le script *saveClassifier.py* prend 2 paramètres : le nom du fichier de données et le type de classifieur. Si le fichier désiré est stocké en local, il faut le préciser en mettant le flag *-l* indiquant que c'est un ficher personnel qui doit être chargé. Si le flag *-o* n'est pas renseigné, le classifieur sera enregistré sous la forme *inputFile_ classifierType.sav*, sinon il prendra le nom indiqué, suivi du type de classifieur, *output_ classifierType.sav*.

```
> python3 saveClassifier.py GunPoint 1NN -o G1
```

Par exemple, la ligne de commande ci-dessus enregistrera un classifieur 1NN sous le nom G1, appris sur la base de données GunPoint. Le fichier G1_1NN.sav est alors créé et contient tous les paramètres significatifs du classifieur.

1.4.2 Commande : affichage de l'explication

```
> python3 TSExplanation.py <inputFile> <classifierFile>
[-i INDEX] [-f FEATURES] [-c CUTS] [-s SAMPLES] [-o OUTPUT] [-l]
```

Le script *TSExplanation.py* prend également 2 paramètres : le nom du fichier de données dans laquelle se trouve la série à expliquer et le fichier correspondant au classifieur souhaité. Plusieurs flags optionnels sont disponibles :

- -l : indique que le fichier d'entrée est en local,
- -i <INT> : spécifie la position de la série temporelle à expliquer dans le fichier d'entrée (défaut : 0),
- -f <INT> : spécifie le nombre d'attributs à prendre en compte dans l'explication (défaut : 10),
- -c <INT> : spécifie le nombre de sous séries temporelles. Plus le nombre est grand, plus la segmentation sera fine (défaut : 24),
- -s <INT> : spécifie le nombre de voisins à générer pour pouvoir fournir une explication (défaut : 1000),
- -o <STRING> : spécifie un nom d'enregistrement, si non renseigné, le script ne fera qu'afficher la courbe résultat.

```
> python3 TSExplanation.py GunPoint
../Classifier/SaveClassifierFiles/G1_1NN.sav -i 3 -c 30 -o explanation
```

Par exemple, la ligne de commande ci-dessus générera une explication de la 4ème (indice 3) série temporelle du jeu de données GunPoint à l'aide du classifieur G1_1NN.sav. La série temporelle sera divisée en 30 sous séries temporelle. Seules les 10 sous séries les plus significatives seront prises en compte. 1000 voisins seront générés. A l'issu du script, l'explication sera enregistrée sous le nom *explanation*.

2 Compte rendu des phases de test

Cette partie rendra compte des tests qui ont été effectués aussi bien sur le fonctionnement de TSExplanation que sur les résultats de notre solution.

2.1 Tests de fonctionnement

Les notebooks suivants sont des tests de nos fonctionnalités. Le premier permet d'importer une série temporelle, de l'afficher puis d'entraîner un classifieur, de le sauvegarder et de le charger. Le second notebook permet de générer le résultat du classifieur et l'explication.

Import des librairies

```
import importTS
import LearningClassifier
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score
from tslearn.preprocessing import TimeSeriesScalerMinMax
import matplotlib.pyplot as plt
```

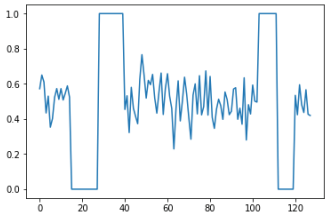
Import du jeu de série temporelle

- X_train : ST d'entrainement
- Y_train : labels associés aux ST d'entrainement
- X_test : ST de test
- Y_test : labels associés aux ST de test

```
X_train, Y_train, X_test, Y_test = importTS.dataImport("TwoPatterns")
```

Affichage de la première ST d'entrainement

```
l = X_train[0].ravel().tolist()
plt.plot(l, linestyle='-')
plt.show()
```



Apprentissage du classifier sur les séries d'entrainement.

```
clLS = LearningClassifier.learningShapeletClassifier(X_train, Y_train)
```

Score du classifier (nombre de classification correcte/ nombre de série temporelle de test)

```
predicted_labels = clLS.predict(X_test)
print("Correct classification rate :", accuracy_score(Y_test, predicted_labels))
```

```
Correct classification rate : 0.71475
```

Sauvegarde du classifier dans un fichier

```
LearningClassifier.saveClassifierLS(clLS, "LS")
```

Chargement du même classifier à partir du fichier sauvegardé

```
loadclLS, labelsave = LearningClassifier.loadClassifieurLS("_LS_.sav")
```

On reteste le classifier avec le même jeu de données test pour s'assurer que l'on est en présence du même classifier

```
newlabel = loadclLS.predict(X_test)
print("Correct classification rate :", accuracy_score(Y_test, labelsave.inverse_transform
(newlabel)))
```

```
Correct classification rate : 0.71475
```

FIGURE 4 – Notebook sur les imports des ST et sur l'apprentissage du classifieur

Import librairies

```
import sys
sys.path.insert(0, "../Classifier")
import importTS
import LearningClassifier
import lime_timeseries
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier as KNN
import matplotlib.pyplot as plt
```

```
Using Theano backend.
```

Import des séries temporelles d'entrainements et de tests

```
X_train, Y_train, X_test, Y_test = importTS.dataImport("TwoPatterns")
```

Construction des classifieurs 1NN et LearningShapelet

```
cl = LearningClassifier.NN1_Classifier(X_train, Y_train)
```

Paramètrages :

- myTs : ST à expliquer
- num_cuts : nombre de coupe dans la série temporelle
- num_features : nombre de sous séries temporelle significatives
- num_samples : nombre de voisins

```
index  = 0
num_cuts = 27
num_features = 10
num_samples = 1000
myTs = X_test[index].ravel()
```

Création de l'explication

```
myTSexp=lime_timeseries.TSExplainer()
exp = myTSexp.explain_instance(myTs,cl,X_train, num_cuts, num_features, num_samples)
```

Affichage du résultat :

- Liste des poids (numéro de la sous-série, poids)
- Graphique

FIGURE 5 – Notebook sur l'explication

```
print(exp.as_list())
exp.domain_mapper.as_pyplot(exp, myTs, num_cuts)
```

```
[(20, 0.2695211814765353), (15, 0.26291234586296464), (18, 0.25221330738491765), (16, 0.
19696322310911704), (19, 0.08152687741041448), (8, -0.06784755249108475), (12, -0.065341
1023126917), (17, 0.06187922270047791), (10, 0.06132784308467147), (22, 0.05035093347239
5734)]
[(20, 0.2695211814765353), (15, 0.26291234586296464), (18, 0.25221330738491765), (16, 0.
19696322310911704), (19, 0.08152687741041448), (8, -0.06784755249108475), (12, -0.065341
1023126917), (17, 0.06187922270047791), (10, 0.06132784308467147), (22, 0.05035093347239
5734)]
```

```
(<matplotlib.backends.backend_qt5agg.FigureCanvasQTAgg at 0x1218404c8>,
 <Figure size 864x576 with 1 Axes>)
```

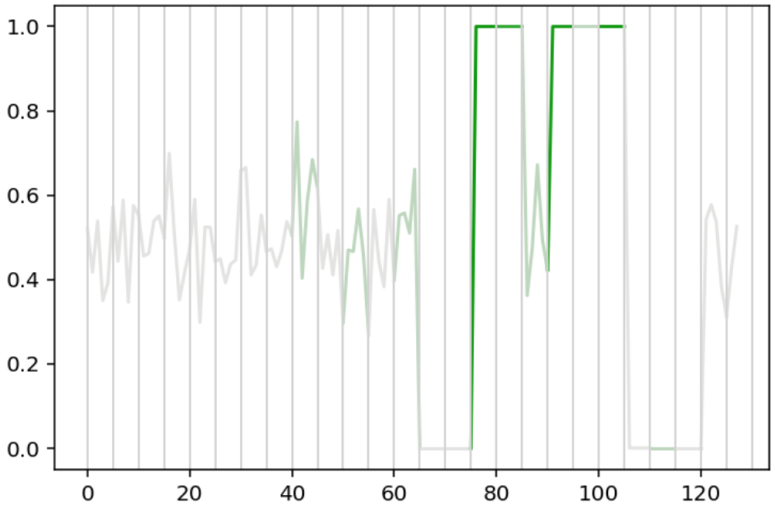


FIGURE 6 – Notebook sur l'explication

2.2 Tests sur les résultats obtenus

Dans un premier temps, pour tester la cohérence de nos résultats, nous avons créé un jeu de données simple avec 2 classes facilement identifiables. Chaque classe est identifiable avec un "pic" situé à deux endroits différents. Sur la Figure 7, deux séries temporelles représentant chacune une classe sont affichées. La série associée à la classe 1 est en rouge, la série associée à la classe 2 est en bleu. Le jeu d'entraînement contient 50 séries (25 par classe).

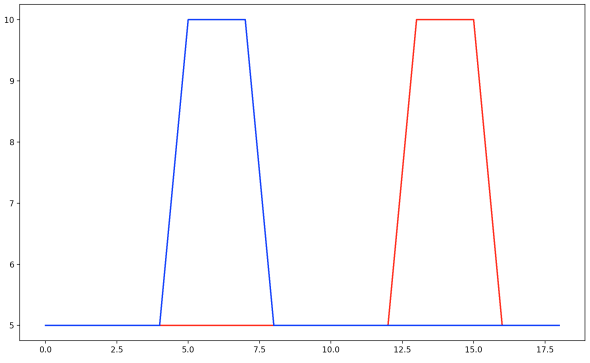


FIGURE 7 – Deux séries temporelles d'entraînement représentant chacune une classe

Nous avons fait apprendre un classifieur 1NN sur ce jeu d'entraînement. Puis nous avons voulu générer le résultat et l'explication de la classification d'une série non annotée. La Figure 8 ci-dessous présente le résultat :

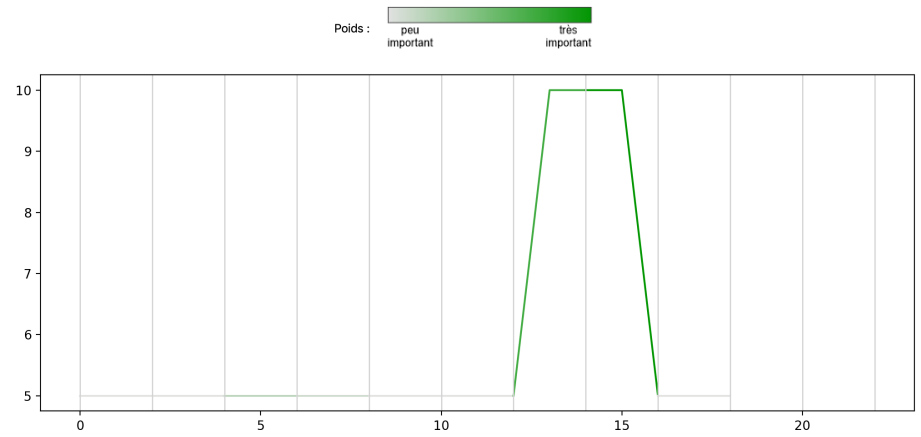


FIGURE 8 – Affichage du résultat de la classification et de l'explication du 1er exemple

On remarque dans un premier temps que la classification semble correcte. La série temporelle de test appartient bien à la classe 1. En ce qui concerne l'explication, les sous-séries qui ont permis d'aider à la classification se situent au niveau du pic (poids fort) et au niveau de l'absence de pic (poids plus faible). Pour cette série, l'explication est correcte.

On refait le test avec une autre série temporelle de test. La Figure 9 ci-dessous présente le résultat :

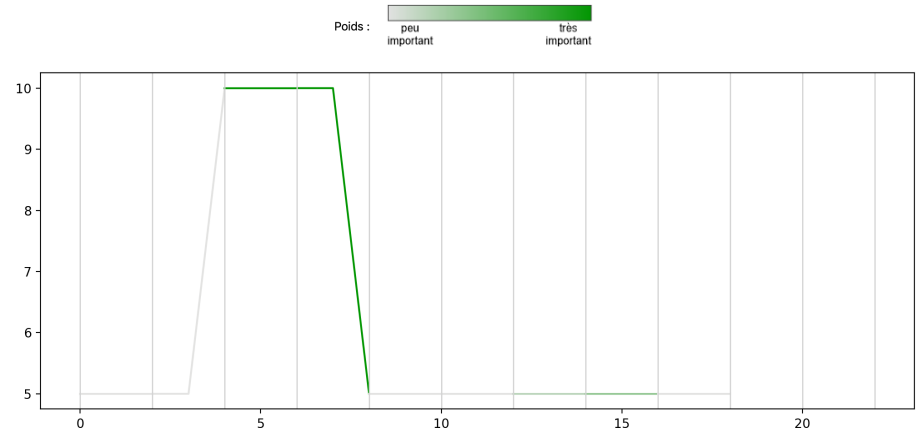


FIGURE 9 – Affichage du résultat de la classification et de l'explication du 2nd Exemple

On peut faire les mêmes observations que sur le test précédent. Ici aussi, le résultat et l'explication semblent justes.

Les tests effectués précédemment se basent sur un jeu de données très simpliste. Néanmoins cela permet d'avoir un premier retour sur nos résultats. Pour pouvoir tester sur des bases plus complexes, deux moyens seraient possibles :

— adapter l'algorithme FastShape. Cet algorithme permet de récupérer des Shapelets qui ont permis la classification. On peut alors comparer ces Shapelets avec les sous-séries significatives que nous obtenons,
— mettre en oeuvre une user-study, c'est à dire un sondage auprès d'utilisateurs pour avoir des retours sur notre travail.

Malheureusement par manque de temps, nous n'avons pas pu faire ces tests.

3 Rectifications et mises à jour depuis les rapports de spécifications et de conception

3.1 Découpage de la série temporelle

Lors du rapport de spécification, il était prévu de réaliser différents types de découpe de la série temporelle en pré-traitement. En effet, deux possibilités de découpe étaient présentées :

— une découpe "uniforme" à intervalle régulier,
— une découpe "intelligente" qui recherche s'il y a plusieurs motifs identiques dans la série temporelle et si c'est le cas, elle les découpe de la même manière.

Seul le premier type de découpe a été réalisé, faute de temps.

3.2 Changement de l'algorithme 1NN-DTW à 1NN distance euclidienne

Lors de la partie conception de ce projet, la classification des séries temporelles a été réalisée avec deux classifieurs différents. Les résultats avec le premier classifieur Learning Shapelet ont été concluants, mais lors des tests avec le second classifieur 1NN-DTW le temps d'exécution était trop long. Ce temps est dû à la complexité de ce classifieur que l'on peut estimer en O(9×10^9) pour 1000 séries temporelles, 1000 correspondant au nombre de voisins.
Le classifieur 1NN-DTW est toujours disponible dans l'interface graphique, mais pour obtenir des résultats concrets pour ce projet, il a été décidé de plutôt utiliser le classifieur 1NN couplé à la distance euclidienne. En effet, ce classifieur possède une complexité bien plus faible, ce qui permet d'avoir des résultats aux tests, et donc de pouvoir comparer ceux-ci avec ceux du classifieur Learning Shapelet. La complexité du classifieur avec la distance euclidienne est, quant à elle, autour de O(3×10^7) pour 1000 séries temporelles.

3.3 Interface graphique

Dans le rapport de conception, un diagramme de classes de l'interface graphique avait été réalisé. Cependant il a dû être modifié à plusieurs reprises pour répondre aux besoins du projet. La Figure 10 montre le diagramme mis à jour, les éléments en rouge sont ceux qui ont été supprimés et les éléments en vert sont ceux qui ont été ajoutés.

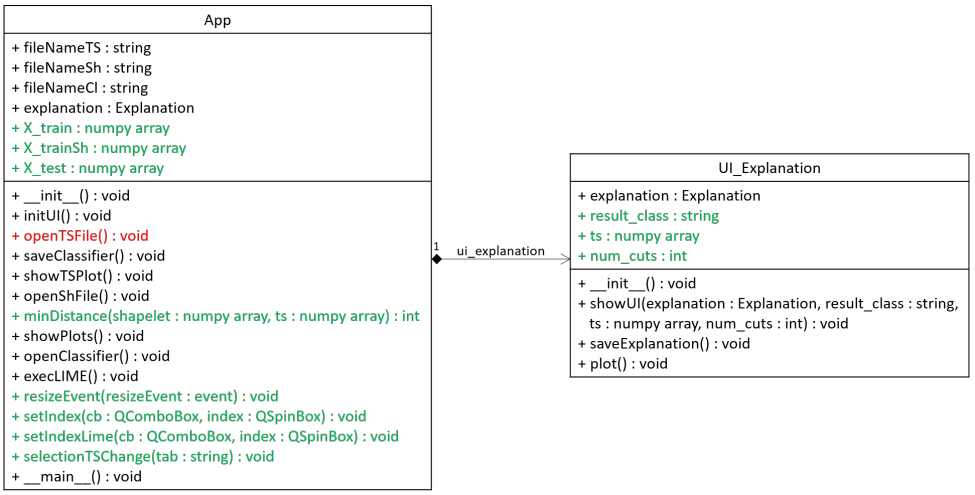


FIGURE 10 – Diagramme de classe de l'interface graphique

Plusieurs attributs ont été rajoutés afin de pouvoir transférer des données d'une fonction à une autre. Il a également fallu ajouter une fonction *minDistance* qui permet de trouver l'emplacement où la distance entre une Shapelet et une série temporelle est minimale. De plus, il a été nécessaire de créer des fonctions pour gérer les indices des séries temporelles et éviter toutes erreurs de bornes. Enfin, la fonction *openTSFile* a été remplacée par la fonction *selectionTSChange* pour prendre en compte les séries temporelles venant de la base de séries temporelles de l'UCR. Les quatre parties suivantes détaillent les modifications apportées à chaque onglet de l'application.

3.3.1 Entraînement et sauvegarde d'un classifieur

Dans l'onglet "Classifieur" de TSExplanation, quelques changements ont été effectués. Il est désormais possible d'entraîner et de sauvergarder un classifieur 1NN avec la distance euclidienne, en plus des classifieurs Learning Shapelet et 1NN-DTW déjà présents. De plus, il y a maintenant une fenêtre popup qui s'affiche pour informer l'utilisateur qui la sauvegarde du classifieur est terminé.

3.3.2 Affichage d'une série temporelle

Dans l'onglet "ST" de TSExplanation, une seule modification a été apportée. L'utilisateur peut à présent se servir de séries temporelles venant de la base de séries temporelles du site *TimeSeriesClassification.com*.

3.3.3 Affichage de la distance entre une série temporelle et une Shapelet

Dans l'onglet "Shapelet" de TSExplanation, l'affichage a été épuré pour mieux visualiser les informations essentielles. Ainsi les aperçus de la Shapelet et de la série temporelle ont été supprimés car peu utiles. De plus, l'utilisateur peut à présent se servir de séries temporelles venant de la base de séries temporelles du site *TimeSeriesClassification.com*.

3.3.4 Explication de la classification d'une série temporelle

L'onglet "LIME" de TSExplanation est celui qui a connu le plus de changements. Tout d'abord, comme dans les onglets précédents, l'utilisateur peut à présent se servir de séries temporelles venant de la base de séries temporelles du site *TimeSeriesClassification.com*. Les autres modifications proviennent des paramètres à saisir pour utiliser l'algorithme LIMEShape, ces derniers ont évolués au fur et à mesure de son implémentation. En effet, le nombre de classes à expliquer n'est plus utilisé. Au contraire, le nombre de voisins à prendre en compte, le nombre de segmentations de la série temporelle ainsi que le type de remplacement des sous-séries temporelles sont des paramètres qui ont été ajoutés à l'interface. Ce dernier paramètre n'a pour le moment qu'une modalité, le remplacement par des segments nuls, mais il pourra faire l'objet d'ajouts dans le cadre d'une future d'amélioration. En ce qui concerne la fenêtre contenant l'explication, une légende a été ajoutée afin de faciliter sa compréhension. Le code couleur de l'explication a été également changé, à présent les poids sont classés du plus important (vert) au moins important (gris).

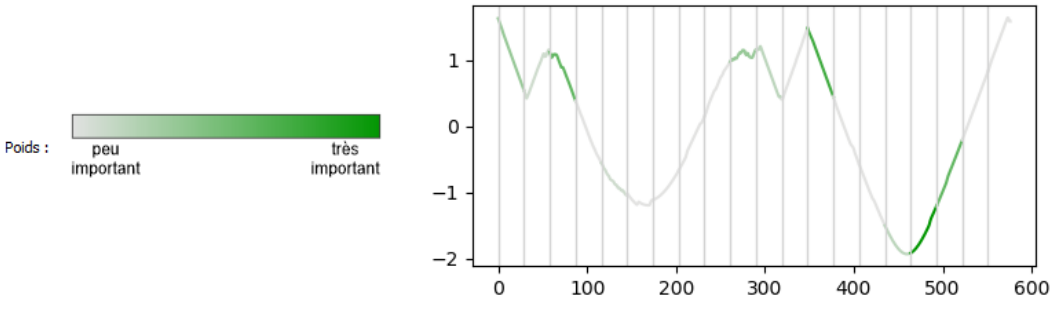


FIGURE 11 – Nouveau code couleur de l'explication

3.4 Paramètres des lignes de commande

Le comportement des lignes de commandes a été légèrement modifié par rapport à ce qui avait été annoncé dans le rapport de spécifications.

3.4.1 Commande : apprentissage et sauvegarde d'un classifieur

L'emplacement de sauvegarde du classifieur ne doit plus être renseigné directement dans la ligne de commande. Tous les classifieurs sont enregistrés automatiquement dans un dossier spécifique.

3.4.2 Commande : affichage de l'explication

Le détail des classes à expliquer n'est plus à renseigner, puisqu'on n'explique qu'une seule série temporelle à la fois. Un paramètre correspondant à la technique de mesure de distance n'est plus variable et a donc été retiré des paramètres de la ligne de commande. Certains paramètres ont cependant été rajoutés comme le nombre de voisins à générer ou la finesse de segmentation de la série temporelle.

4 Bilan de la planification

4.1 Organisation et gestion du projet

Pour une bonne organisation du projet, il était nécessaire de répartir le travail en modules de façon optimale pour gagner en temps et en efficacité. Dans notre groupe, cinq personnes étaient toujours présentes lors du second semestre.

Il est évident que le fait d'être en charge d'un module n'empêchait en rien les personnes de travailler sur les autres modules. Une personne responsable d'un module ne devait pas travailler exclusivement sur ce module mais simplement être consciente de son avancée, des besoins et des problèmes rencontrés et faire remonter ces remarques pendant les réunions hebdomadaires.

Pour éviter au maximum les conflits au cours du développement, nous avons utiliser un gestionnaire de version. Pour cela, nous avons utilisé *GitHub* comme plateforme d'hébergement, cette dernière intégrant la fonctionnalité de Merge Request. Ceci dans le but de permettre à tous les membres du groupe d'être au courant de l'avancée de chacun, permettant ainsi de travailler de manière collaborative.

Afin de rendre ce fonctionnement le plus efficace possible, chaque personne a travaillé sur une partie la plus fine possible du code, c'est-à-dire chaque personne a apporté des modifications à un endroit précis ou un comportement précis du code et n'a pas touché à trop de fichiers différents afin d'éviter au mieux les conflits.

Nous avons ainsi réparti le projet entre l'apprentissage et la sauvegarde des classifieurs, la partie concernant LIME et enfin l'interface graphique. Toutes ces parties ont pu être reliées grâce aux moyens mis en place tout au long de l'année. Ces derniers ont permis aux membres du groupe de rester au courant de l'avancée de chaque partie.

4.2 Respect des délais et du planning

Concernant la partie de développement, différents programmes sont rendus :
— le code inspiré de LIME adapté aux séries temporelles,
— le code pour entraîner et sauvegarder des classifieurs de séries temporelles,
— une interface utilisateur.

Nous rendons aussi un exécutable correspondant à l'application et comprenant toutes ces parties. Un manuel utilisateur indiquant toutes les commandes à effectuer permet à l'utilisateur d'avoir les instructions nécessaires.

Pour estimer le temps nécessaire à la réalisation de ce projet, ce dernier a d'abord été découpé en tâches. Ainsi, il a été plus facile d'attribuer à chaque tâche une estimation de temps. Il a été décidé de diviser le travail à faire en plusieurs tâches différentes. Ensuite, chaque tâche a été découpée en sous-tâches afin de donner une estimation encore plus précise du temps nécessaire pour mener chaque tâche à bien. Le découpage du projet en tâches était le suivant :

— architecture de l'application,
— générateur de classifieurs de série temporelle,
— adaptation de LIME aux séries temporelles,
— interface Graphique,
— tests de validation.

Nous avions ensuite fait l'étude des tâches parallélisables et réalisé la hiérarchie des tâches lors du rapport de planification. Grâce au diagramme de Gantt établi, et aux jalons posés,

nous avons réussi à réaliser les différentes tâches dans le temps imparti afin de respecter les délais imposés.

4.3 Estimation des heures

Nous avons effectué un suivi du temps de travail tout au long du projet lors de nos réunions hebdomadaires : cela nous a permis d'analyser la charge de travail sur les trois parties du projet ainsi que sur les tests. La Figure 12 montre la répartition du temps de travail pour chaque tâche.

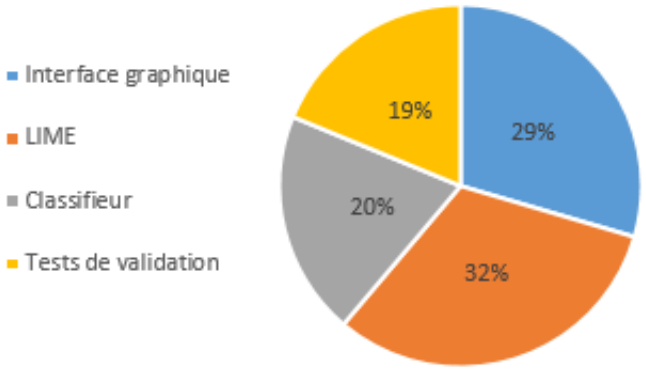


FIGURE 12 – Répartition du temps de travail par tâches

Le total final des heures passées sur ce projet est de 1112h à la rédaction de ce rapport, et en particulier 366h sur le second semestre. La Figure 13 montre la répartition du temps de travail par semaine pour l'ensemble du groupe, entre le 14 janvier et le 6 mai. Les relevés ont été fait à chaque réunion de projet, à savoir tous les sept jours. Lors de la soutenance de planification, nous avions estimé la charge de travail sur le projet au second semestre à 390 heures, soit 5h par étudiant par semaine. Cette estimation est proche de ce que nous avons atteint à la fin du projet, une fois la soutenance et le showroom passés.

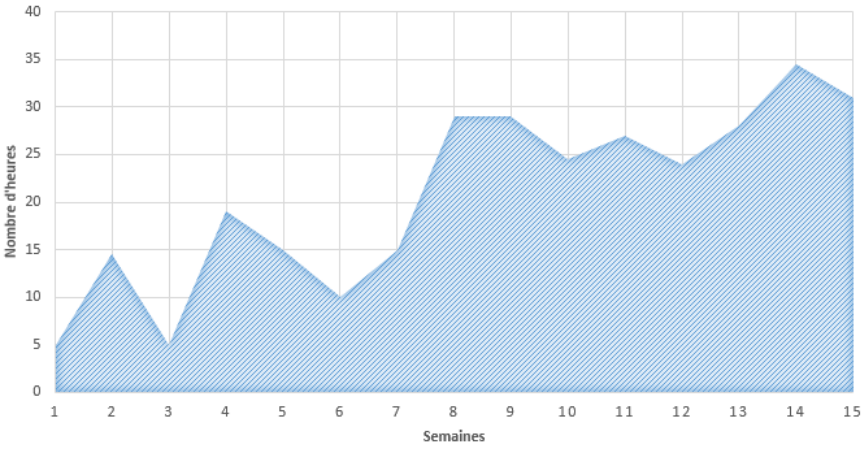


FIGURE 13 – Répartition du temps de travail sur le semestre

Concernant les estimations des heures qui étaient prévues pour chaque partie, il est possible de prendre du recul maintenant que toutes les parties ont été finalisées.

Il a été nécessaire de passer plus de temps sur le classifieur 1NN car il y a eu un changement, comme expliquée dans la partie *3.2 Changement de l'algorithme 1NN-DTW à 1NN distance euclidienne*. En effet, l'estimation du temps passé sur le classifieur 1NN-DTW était correcte mais suite aux tests sur ce classifieur, nous avons dû implémenter le classifieur 1NN couplé à la distance euclidienne. C'est un imprévu qui a nécessité de passer plus de temps sur la partie classifieur. Cependant, lors de la planification nous avions prévu du temps pour les imprévus, donc ce changement n'a pas posé de problème dans notre planning.

Au sein de la partie LIME, 30 heures avaient été allouées pour chacune des trois classes. Cependant, la classe *TSExplainer* a requis plus de temps que les deux autres, celle-ci étant dépendante de ces deux premières classes. De plus, il a fallu ensuite lier avec l'interface graphique pour réaliser des tests supplémentaires et se rendre compte des possibles erreurs à corriger. Avec du recul, nous réalisons que nous aurions pu estimer 20 heures pour chacune des classes *TSDomainMapper* et *IndexedTS* et 50 heures pour la classe *TSExplainer*.

Pour l'interface graphique, les durées estimées ont été globalement respectées. Cependant, à cause de l'évolution de la partie LIME, le dernier onglet de l'interface a nécessité plusieurs changements tout au long du projet.

Conclusion

Au terme de ce projet de 4ème année INSA, les objectifs initiaux ont été atteints. L'outil TSExplanation permet aux utilisateurs d'entraîner un classifieur de séries temporelles 1NN couplé avec une distance euclidienne, un classifieur 1NN-DTW, ou un classifieur Learning Shapelet. Il est ensuite possible de sauvegarder un tel classifieur, de l'appliquer sur de nouvelles séries temporelles et de générer une explication du résultat de cette classification.

Plusieurs difficultés ont été rencontrées au cours de ce projet, notamment au niveau de la complexité trop élevée du classifieur 1NN-DTW, et au niveau du traitement des séries temporelles et de l'adaptation de l'algorithme LIME. Face à ces difficultés liées aux notions abordées par le sujet, les membres du groupe ont su reformuler certains objectifs et ajouter des fonctionnalités pour délivrer un outil opérationnel.

Le sujet de ce projet se rapprochant d'un sujet de recherche, les phases initiales de pré-étude et de spécification ont été primordiales pour comprendre les enjeux, les objectifs et les solutions existantes et réutilisables. Il a fallu découvrir de nouvelles notions comme celles de séries temporelles et de Shapelets et découvrir l'algorithme complexe qu'est LIME afin de déterminer les éléments de son code qui pouvaient être récupérés et ceux qu'il allait falloir implémenter entièrement. Ce projet a donc permis aux membres du groupe de se familiariser avec ce travail de recherche en amont du développement, qui peut être lourd mais néanmoins capital pour mener à bien un projet comme TSExplanation.

Les objectifs fixés en début de projet sont atteints, les délais ont été respectés, et une interface graphique a été implémentée pour faciliter l'utilisation de l'outil TSExplanation. Il est également possible d'utiliser TSExplanation via des lignes de commande, ce qui vient s'ajouter aux objectifs initiaux et à l'interface graphique.

La documentation utilisateur de cet outil est présente dans les annexes de ce rapport.

Annexes

Annexe A : Documentation utilisateur

1 Interface graphique

1.1 Entraînement et sauvegarde d'un classifieur

A travers l'onglet Classifieur de l'application TSExplanation, l'utilisateur va pouvoir construire et sauvegarder un classifieur, qu'il pourra ensuite utiliser dans l'onglet LIME s'il le désire. Comme le montre la Figure 14, il y a quelques paramètres simples pour choisir le type de classifieur à construire.

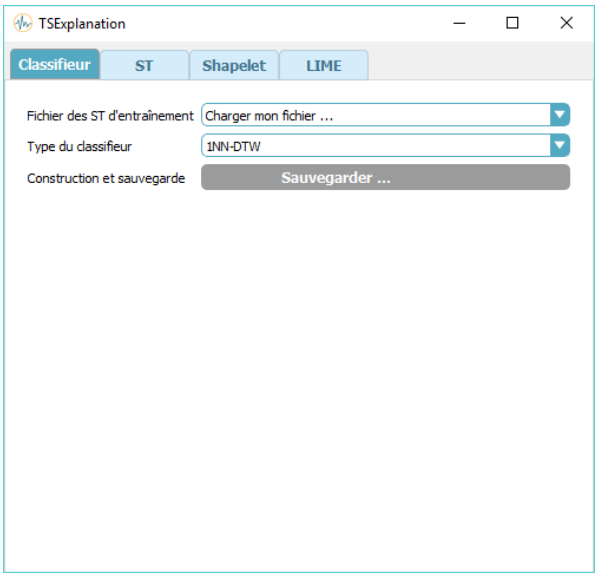


FIGURE 14 – Onglet classifieur

La première étape consiste à choisir le fichier contenant les exemples de séries temporelles qui permettront l'apprentissage du classifieur. L'utilisateur a le choix de sélectionner dans la liste déroulante un fichier de la base de séries temporelles proposée (provenant du site de l'UCR) ou d'utiliser son propre fichier. Dans ce cas cela ouvrira un explorateur de fichiers comme sur la Figure 15 où il suffira de naviguer et de sélectionner le fichier voulu, qui devra être sous le format TXT.

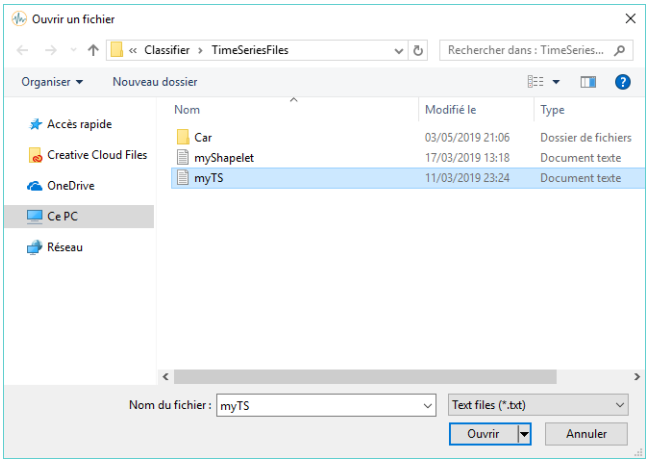


FIGURE 15 – Choix du fichier des exemples

L'étape suivante est la sélection du type de classifieur : 1NN-DTW, 1NN ou Learning Shapelet. Pour finir, il reste l'étape de l'entraînement et la sauvegarde de ce classifieur. Le bouton "Sauvegarder..." permet d'ouvrir un explorateur de fichiers comme la Figure 16, l'utilisateur peut alors choisir le chemin et le nom de sauvegarde du classifieur, le fichier aura l'extension *.sav*.

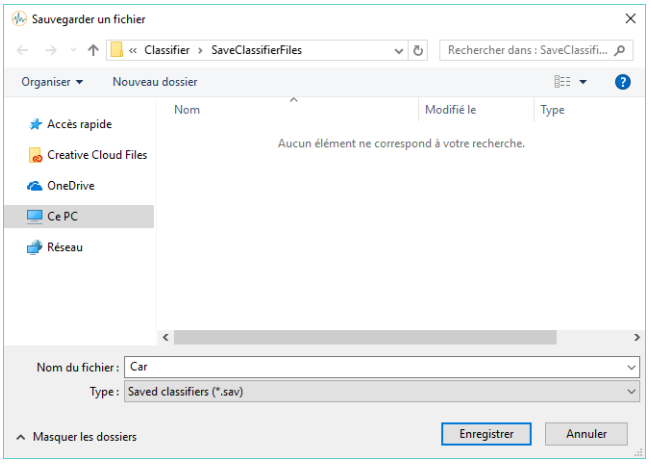


FIGURE 16 – Choix de sauvegarde du classifieur

Après un certain délai, une fenêtre comme sur la Figure 17 apparaît pour confirmer l'entraînement et la sauvegarde du classifieur.

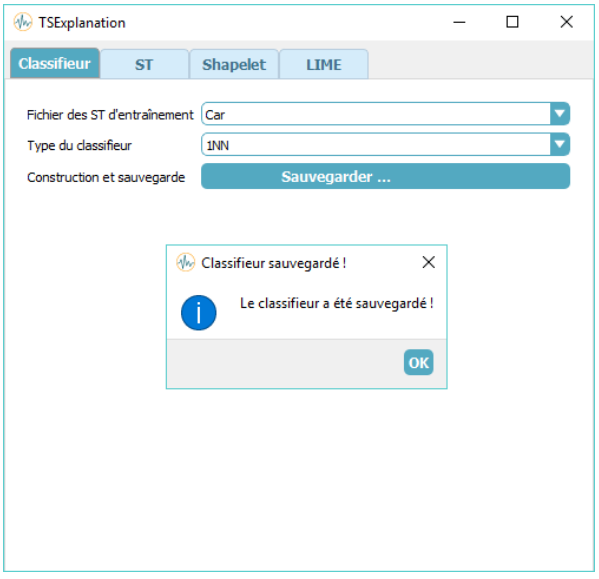


FIGURE 17 – Fin de sauvegarde du classifieur

1.2 Affichage d'une série temporelle

L'onglet ST de l'application a pour fonction l'affichage d'une série temporelle. Comme le montre la Figure 18, il y a peu de paramètres à choisir.

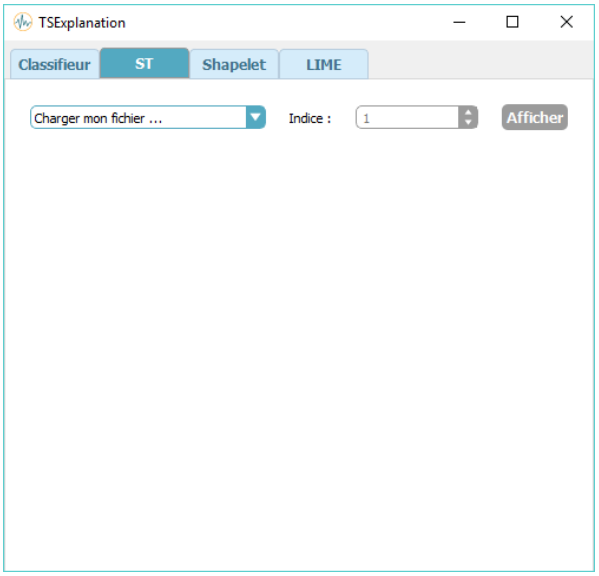


FIGURE 18 – Onglet ST

La première étape consiste à choisir le fichier contenant la série temporelle à afficher. L'utilisateur a le choix de sélectionner dans la liste déroulante un fichier de la base de séries temporelles proposée (provenant du site de l'UCR) ou d'utiliser son propre fichier. Dans ce cas cela ouvrira un explorateur de fichiers comme sur la Figure 19 où il suffira de naviguer et de sélectionner le fichier voulu, qui devra être sous le format TXT.

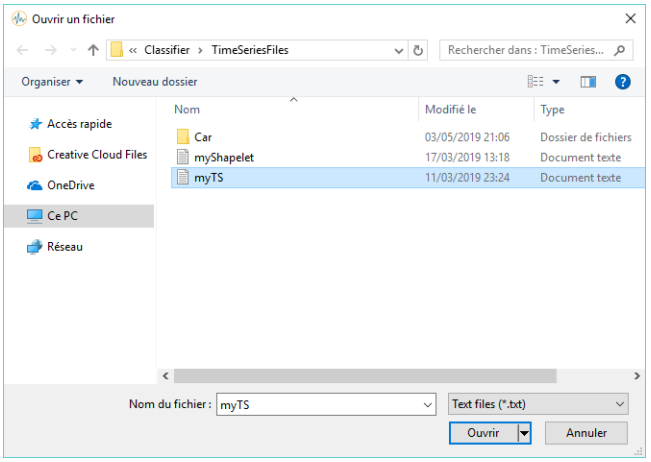


FIGURE 19 – Choix du fichier des séries temporelles

Un fois le fichier de l'ensemble des séries temporelles sélectionné, il faut renseigner l'indice de la série temporelle voulue puis le bouton "Afficher" fait apparaître le graphique associé, comme le montre la Figure 20.

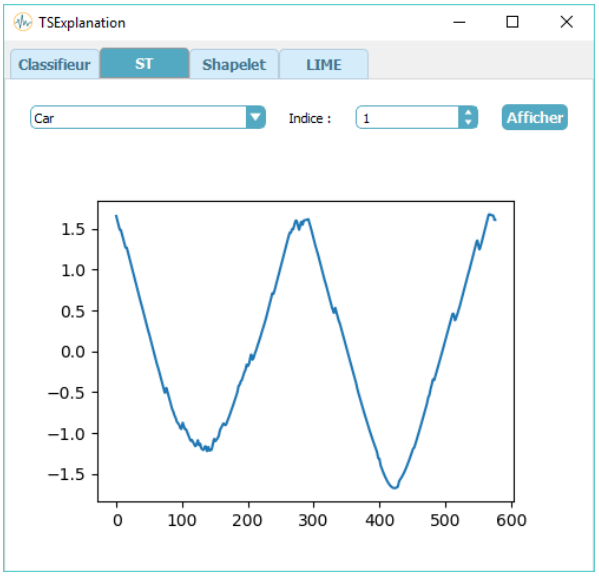


FIGURE 20 – Affichage de la série temporelle

1.3 Affichage de la distance entre une série temporelle et une Shapelet

L'onglet Shapelet propose à l'utilisateur de choisir une shapelet et une série temporelle afin d'afficher graphiquement la distance entre les deux. Dans cet onglet il faut renseigner le même type de paramètres que dans le précédent, comme le montre la Figure 21.

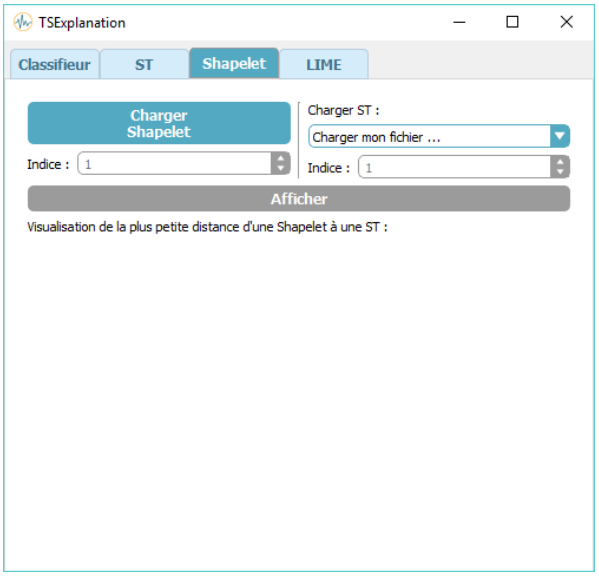


FIGURE 21 – Onglet Shapelet

La première étape consiste à choisir le fichier contenant la Shapelet à utiliser. Le bouton "Charger Shapelet" ouvre un explorateur de fichiers comme sur la Figure 22 où il suffira de naviguer et de sélectionner le fichier voulu, qui devra être sous le format TXT.

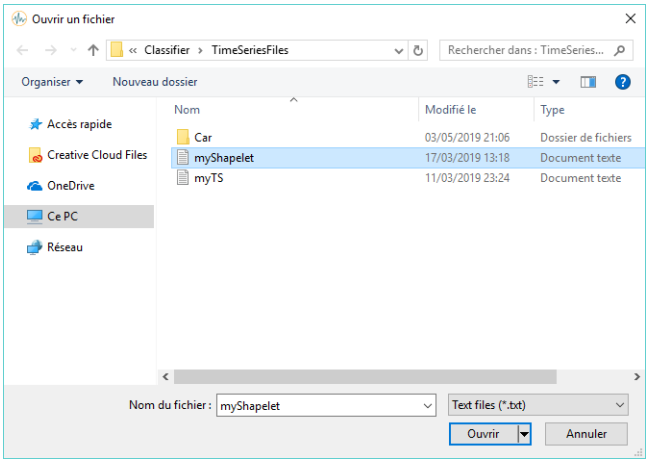


FIGURE 22 – Choix du fichier des Shapelets

Une fois le fichier de l'ensemble des Shapelet sélectionné, il faut renseigner l'indice de la Shapelet voulue. L'étape suivante concerne la série temporelle à afficher. L'utilisateur a le choix de sélectionner dans la liste déroulante un fichier de la base de séries temporelles proposée (provenant du site de l'UCR) ou d'utiliser son propre fichier. Dans ce cas cela ouvrira un explorateur de fichiers comme sur la Figure 23 où il suffira de naviguer et de sélectionner le fichier voulu, qui devra être sous le format TXT.

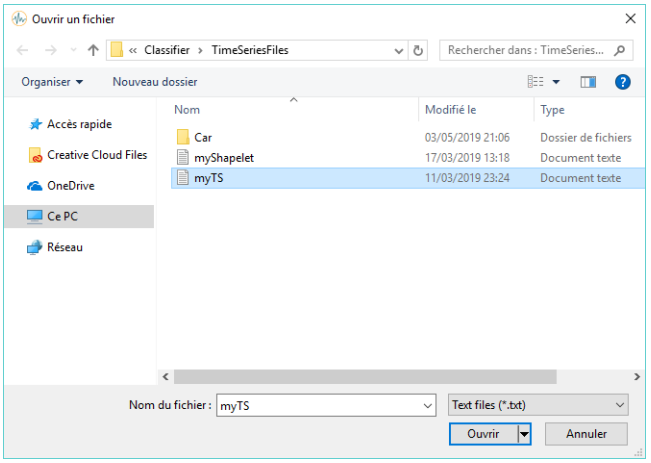


FIGURE 23 – Choix du fichier des séries temporelles

Une fois le fichier de l'ensemble des séries temporelles sélectionné, il faut renseigner l'indice de la série temporelle voulue. Enfin, le bouton "Afficher" fait apparaître le graphique associé, comme le montre la Figure 24. On retrouve en rouge la Shapelet, placée à l'endroit ou la distance à la série temporelle est la plus faible. La zone grisée permet de mieux visualiser l'écart entre les deux courbes. Si la Shapelet sélectionnée est plus courte que la série temporelle, aucun graphique ne s'affiche.

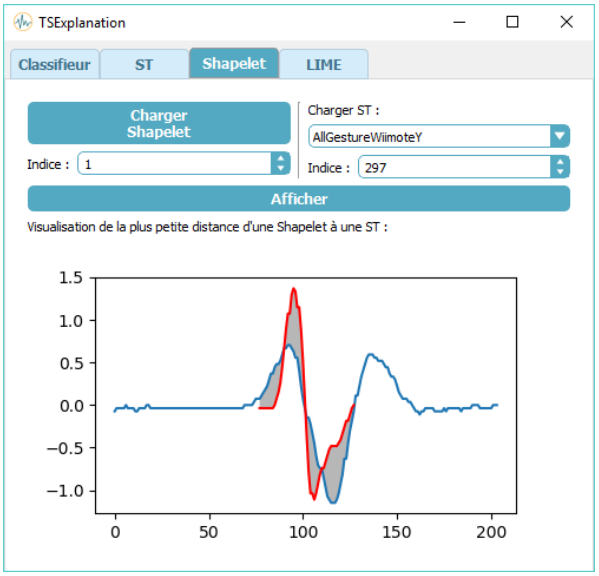


FIGURE 24 – Affichage de la distance

1.4 Explication de la classification d'une série temporelle

L'onglet LIME représente la dernière partie de l'application TSExplanation, il contient la fonction principale de notre outil : l'interprétabilité de séries temporelles. A travers de nombreux paramètres, l'utilisateur peut obtenir l'explication de la classification d'une série temporelle donnée, comme le montre la Figure 25.

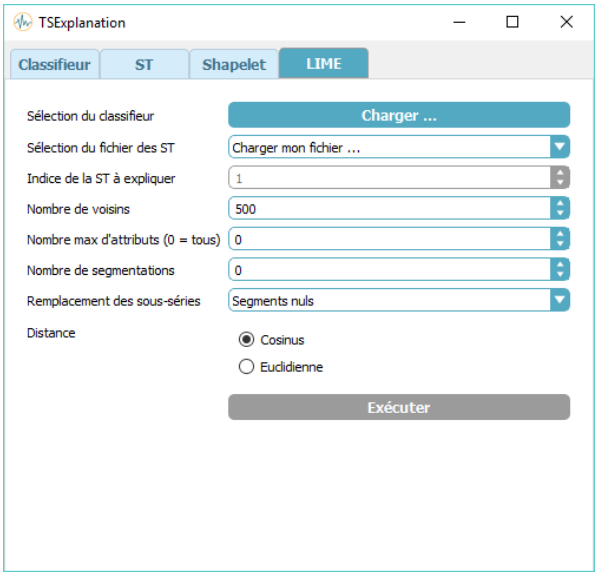


FIGURE 25 – Onglet LIME

Dans un premier temps, l'utilisateur doit sélectionner le classifieur à utiliser. Le bouton associé ouvre un explorateur de fichiers, il faut choisir le fichier avec l'extension *.sav* contenant le classifieur voulu, comme sur la Figure 26.

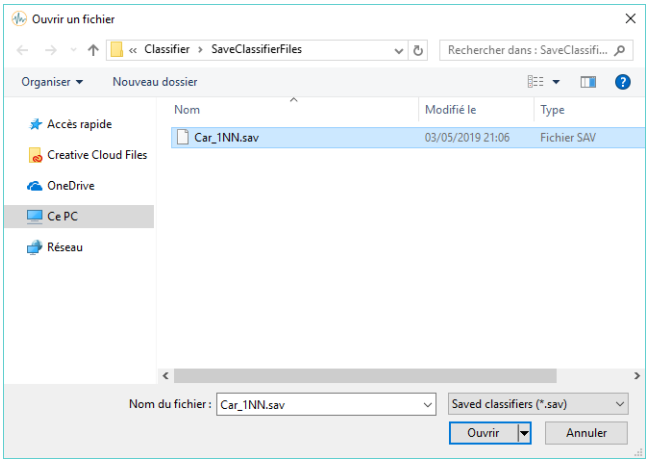


FIGURE 26 – Sélection du classifieur

L'étape suivante est la sélection du fichier contenant les séries temporelles. L'utilisateur a le choix de sélectionner dans la liste déroulante un fichier de la base de séries temporelles proposée (provenant du site de l'UCR) ou d'utiliser son propre fichier. Dans ce cas cela ouvrira un explorateur de fichiers comme sur la Figure 27 où il suffira de naviguer et de sélectionner le fichier voulu, qui devra être sous le format TXT.

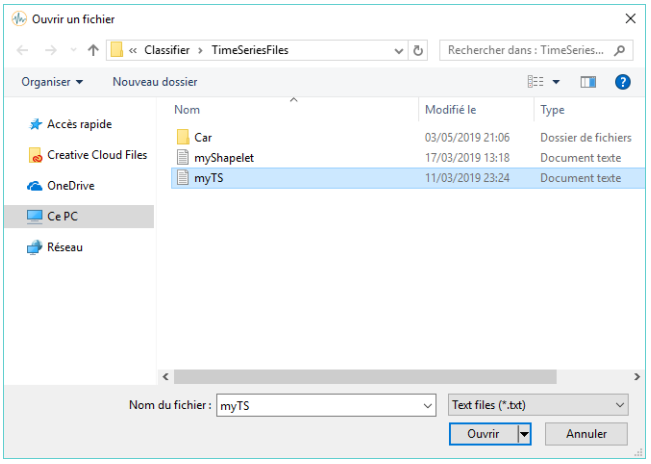


FIGURE 27 – Choix du fichier des séries temporelles

L'utilisateur doit ensuite spécifier l'indice de la série temporelle qu'il veut traiter dans le fichier TXT, ainsi que le nombre de voisins à prendre en compte. Il faut aussi indiquer le nombre maximum d'attributs, le nombre de segmentations, le type de remplacement des sous-séries temporelles et la distance à utiliser. La Figure 28 illustre un exemple de paramètres qui peuvent être utilisés.

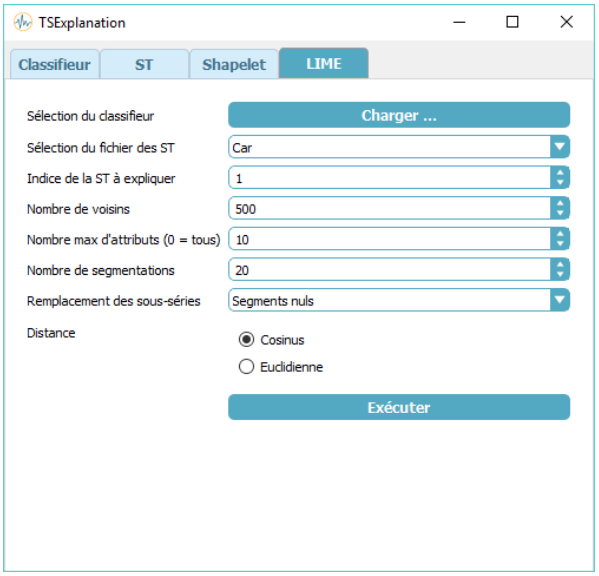


FIGURE 28 – Paramètres de LIMEShape

Enfin, le bouton "Exécuter" permet d'afficher l'explication correspondante calculée avec l'algorithme LIMEShape, une fenêtre Explanation comme la Figure 29 s'ouvre alors.

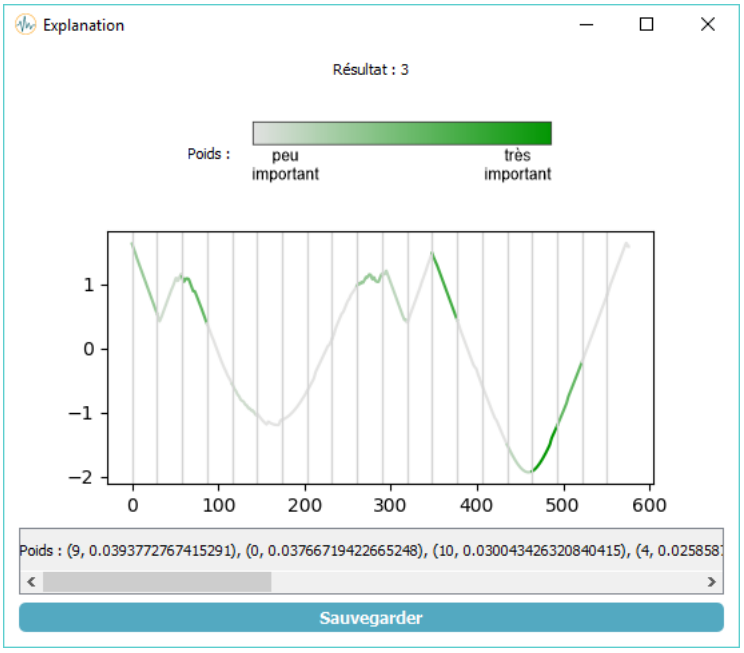


FIGURE 29 – Affichage de l'explication

Le résultat de la classification apparaît en haut de la fenêtre. Les différentes sous-séries temporelles qui composent la série temporelle sont colorées d'après leur poids. Comme l'indique la légende, les sous séries ayant les poids les plus importants sont colorées en vert. Plus les poids baissent, plus la couleur des sous-séries associées tend vers le gris. Sous le graphique on retrouve les valeurs des poids des sous-séries temporelles qui sont triés par ordre croissant. Le bouton "Sauvegarder" permet à l'utilisateur d'enregistrer l'explication sous format HTML, il ouvre un explorateur de fichiers comme sur la Figure 30 où il suffira de naviguer et de sélectionner l'emplacement voulu.

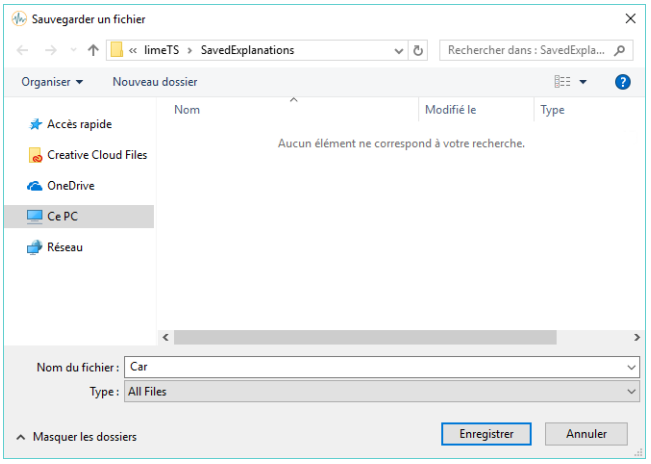


FIGURE 30 – Choix de l'emplacement de sauvegarde de l'explication

Une fois l'emplacement choisi, un dossier contenant le fichier HTML et ses images est créé. Le fichier HTML ressemble à la Figure 31, il contient les mêmes informations que la fenêtre Explanation.

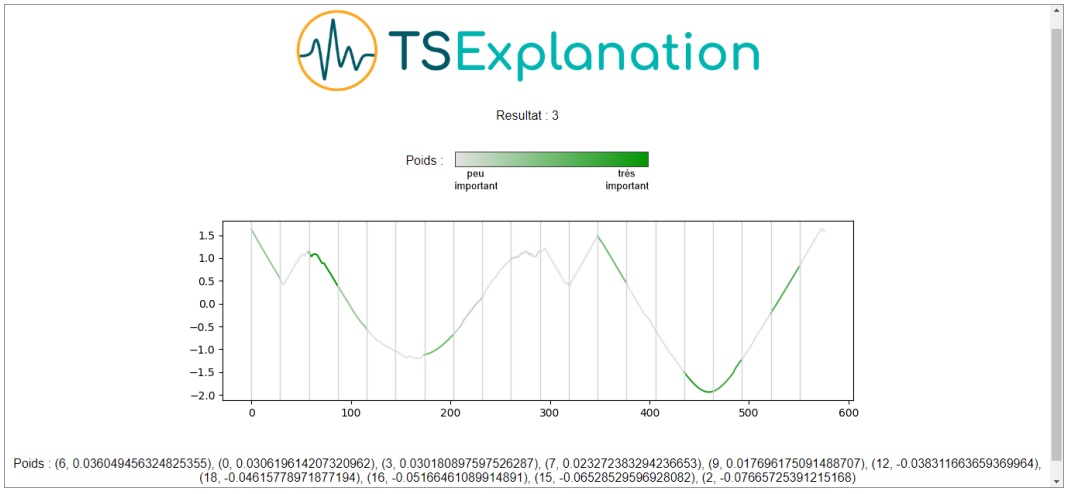


FIGURE 31 – Fichier HTML de l'explication

2 Lignes de commande

2.1 Entraînement et sauvegarde d'un classifieur

La Figure 32 ci-dessous explique les différents paramètres de cette ligne de commande.

```
usage: saveClassifier.py [-h] [-o OUTPUT] [-l LOCAL]
                         {Car,GunPoint,TwoPatterns,local} {1NN,1NN-DTW,LS}

positional arguments:
  {Car,GunPoint,TwoPatterns,local}
                        TS used to learn the classifier
  {1NN,1NN-DTW,LS}      classifier's type

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT, --output OUTPUT
                        name of the output file, it will be followed by the
                        <classifier_type>.
  -l LOCAL, --local LOCAL
                        indicate that the TS file <input_file> is on the
                        computer
```

FIGURE 32 – Aide utilisateur de la sauvegarde d'un classifieur par ligne de commande

2.2 Explication de la classification d'une série temporelle

La Figure 33 ci-dessous explique les différents paramètres de cette ligne de commande.

```
usage: TSExplanation.py [-h] [-i INDEX] [-f FEATURES] [-c CUTS] [-s SAMPLES]
                        [-o OUTPUT] [-l LOCAL]
                        {Car,GunPoint,TwoPatterns,local} classifier

positional arguments:
  {Car,GunPoint,TwoPatterns,local}
                        TS file
  classifier            classifier

optional arguments:
  -h, --help            show this help message and exit
  -i INDEX, --index INDEX
                        index of the ts explained (default : 0).
  -f FEATURES, --features FEATURES
                        max of features present in explanation (default : 10).
  -c CUTS, --cuts CUTS  number of sub series, i.e. segmentation (default :
                        24).
  -s SAMPLES, --samples SAMPLES
                        size of the neighborhood to learn the linear model
                        (default : 1000).
  -o OUTPUT, --output OUTPUT
                        save the explanation with the name <output>.
  -l LOCAL, --local LOCAL
                        indicate that the TS file <input_file> is on the
                        computer
```

FIGURE 33 – Aide utilisateur de l'affichage d'une explication par ligne de commande