Kaggle Project

Version public

Lilian Besson

25 Apr 2013, 22h:44m:06s

Email: lilian.besson@normale.fr

Table des matières

Sphinx

Cette page est la page d'accueil de la *documentation* pour mon projet, réalisée avec Sphinx (http ://sphinx-doc.org). Cette doc est aussi générée via *PyDoc*, pouvant être consultée ici.

Bienvenue dans la documentation du **projet Kaggle** (Titanic (https ://www.kaggle.com/c/titanic-gettingStarted)) (pour le cours d'apprentissage de L3 à l'ENS de Cachan).

Ce projet est actuellement en 1.1 *version*, public *release*. Et cette documentation a été actualisée le 25 avril 2013, 22h :35m :23s pour la dernière fois.

1 Auteur

L'auteur principal est **Lilian Besson**, et plus de détails sont disponibles là-bas *Auteurs (fichier AUTHOR)* (page 38).

Voir aussi :

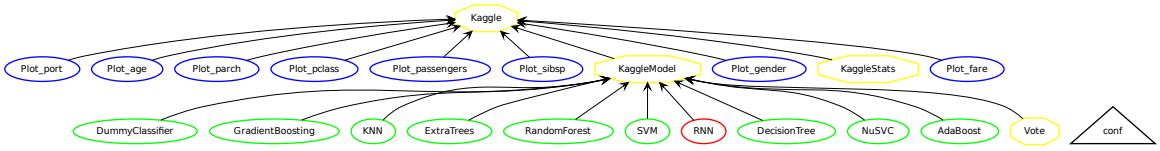
D'autres ressources sont disponibles *en ligne* :

1. sur **Bitbucket.org**, ici lbesson/projet-kaggle (https ://bitbucket.org/lbesson/projet-kaggle).
2. une *archive*, ici kaggle.tar.xz.

2 Plus de détails

Voir le rapport.

Les différents scripts s'organisent comme cela :



3 Installation

3.1 Dependencies

The project is *entirely written in Python*, version *2.7.3*. For more details about the **Python** language, see the official site (http ://www.python.org). Python 2.7.1 or higher is **required**.

The project also **require** the following *unusual module(s)* :

– matplotlib,
– numpy,

– sklearn,
– pylab,

3.2 Plateform(s)

The project have been *developped* on *GNU/Linux* (Ubuntu 11.10).

3.3 License

4 Table des matières

4.1 À propos des données

Source https ://www.kaggle.com/c/titanic-gettingStarted/data

VARIABLE DESCRIPTIONS

survival Survival (0 = No ; 1 = Yes)
pclass Passenger Class (1 = 1st ; 2 = 2nd ; 3 = 3rd)

name Name

sex Sex

age Age

sibsp Number of Siblings/Spouses Aboard

parch Number of Parents/Children Aboard

ticket Ticket Number

fare Passenger Fare

cabin Cabin

embarked Port of Embarkation (C = Cherbourg ; Q = Queenstown ; S = Southampton)

SPECIAL NOTES

– Pclass is a proxy for socio-economic status (SES) 1st ~ Upper ; 2nd ~ Middle ; 3rd ~ Lower
– Age is in Years ; Fractional if Age less than One (1) If the Age is Estimated, it is in the form xx.5
– With respect to the family relation variables (i.e. sibsp and parch) some relations were ignored. The following are the definitions used for sibsp and parch.

Sibling Brother, Sister, Stepbrother, or Stepsister of Passenger Aboard Titanic

Spouse Husband or Wife of Passenger Aboard Titanic (Mistresses and Fiances Ignored)

Parent Mother or Father of Passenger Aboard Titanic

Child Son, Daughter, Stepson, or Stepdaughter of Passenger Aboard Titanic

Other family relatives excluded from this study include cousins, nephews/nieces, aunts/uncles, and in-laws. Some children travelled only with a nanny, therefore parch=0 for them. As well, some travelled with very close friends or neighbors in a village, however, the definitions do not support such relations.

4.2 Graphiques

Le dossier plots/ contient beaucoup de graphiques, utilisés pour montrer certains liens entre des attributs des données et le *taux de survie*.

Plus de détails

Pour chaque script, sa documentation ne détaille pas comment est obtenu le graphique, mais embarque chaque graphique (important), et souligne les points importants montrés par ce graphique.

Scripts

Ces scripts génèrent des graphiques, en se concentrant sur un attribut, ou deux.

Plot_age Module

Some graphics about the training dataset, regarding the age.

Source ../../Plot_age.py

Histogramme Pour tracer les distributions d'âge, on découpe en catégories. Pour ce graphique, par tranche de 10 ans.

Histogramme (taux de survie) Ça montre notamment qu'en 1912 on appliquait à la lettre les principes *"sauvez les enfants et les personnes agées d'abord !"*.

Plot_fare Module

Some graphics about the training dataset, regarding the fare.

Source ../../Plot_fare.py

Histogramme Pour tracer les distributions de prix du ticket, on découpe en catégories.

Histogramme (taux de survie) Cela montre clairement que plus on paie cher plus on “a de change de survivre”.

Diagramme Et ce dernier graphique le montre encore plus.

Plot_gender Module

Some graphics about the training dataset, regarding the gender.

Source ../../Plot_gender.py

Histogramme Cela montre la distribution des survivants et les victimes selon le genre des passagers.

Diagramme Clairement, cela montre encore qu’on sauvait plus les femmes que les hommes en 1912 !

Plot_parch Module

Some graphics about the training dataset, regarding the parch.

Source ../../Plot_parch.py

Histogramme Cela montre la distribution des victimes et survivants selon le nombre de parents et enfants présents parmis les autres passagers du Titanic.

Histogramme (taux de survie) Ce second montre le taux de survie (plutôt une fraction qu’un taux).

On devrait voir que les passagers “un peu” accompagné survivent plus que ceux seuls.

Diagramme Ce diagramme là ne montre pas grand chose par contre.

Plot_passengers Module

Some graphics about the training dataset.

Source ../../Plot_passengers.py

Diagramme Ce diagramme là montre la répartition des survivants et des victimes parmis les passagers.

Diagramme (genre) Ce diagramme là montre la répartition des survivants et des victimes parmis les hommes et les femmes.

Ce diagramme montre aussi que les femmes survivèrent bien plus que les hommes.

Plot_pclass Module

Some graphics about the training dataset, regarding the pclass.

Source ../../Plot_pclass.py

Diagramme 1 Ce diagramme là montre la répartition des trois classes (1ere, 2nd, 3eme) parmis les passagers, puis les survivants et les victimes.

Diagramme 2 Ce diagramme là montre la répartition des survivants et des victimes parmis les trois classes (1ere, 2nd, 3eme).

Ce qui montre encore une fois que la première classe fut "plus sauvée" que les deux autres.

Le taux de survie de la première est plus de deux fois supérieur que la troisième !

Plot_port Module

Some graphics about the training dataset, regarding the embarcation port.

Source ../../Plot_port.py

Histogramme Cet histogramme montre répartition des survivants et des victimes parmis les trois ports d'embarcations (Southampton, Cherbourg, Queenstown).

Clairement, les deux derniers ports d'embarcations ont embarqués bien moins de passagers que Southampton. Donc, il faut nuancer les remarques suivantes, puisqu'il y a peu de passagers venant de Cherbourg et de Queenstown.

Diagramme Ce diagramme là montre la répartition des trois ports d'embarcations (Southampton, Cherbourg, Queenstown) parmis les survivants et des victimes (soit le point de vue inverse).

Ces deux graphiques montrent que les passagers ayant embarqués à Southampton semblent plus "fragiles" alors que ceux ayant embarqués à Cherbourg semblent plus "robustes". Et pour Queenstown (les Irlandais), il ne semble pas y avoir de différence.

Plot_sibsp Module

Some graphics about the training dataset, regarding the sibsp.

Source ../../Plot_sibsp.py

Histogramme Cela montre la distribution des victimes et survivants selon le nombre de frères & soeurs, et époux/épouses présents parmis les autres passagers du Titanic.

Histogramme (taux de survie) Ce second montre le taux de survie (plutôt une fraction qu'un taux).

On devrait voir que les passagers "un peu" accompagné survivent plus que ceux seuls.

Diagramme

Les deux catégories les plus présentes sont 0 et 1 (c'est-à-dire les passagers voyageant seuls ou avec **UN** seul membre de leur famille).

Et ce diagramme montre que les personnes seules survivent moins (68% $\rightarrow$ 72%), alors que les personnes accompagnées survivent plus (23% $\rightarrow$ 33%).

4.3 Representation Module

A representation tool for Titanic dataset.

The doc is here : http ://scikit-learn.org/dev/modules/classes.html

---

Je veux essayer d'utiliser les algorithmiques de clustering, de changement de représentations etc...

Sortie du script

```
$ python Representation.py
Opening the file 'train.csv' and 'test.csv'...
chi2:
        For the attribute pclass        , chi2=30.8736994366        , and pval=2
        For the attribute sex        , chi2=92.7024469789        , and pval=6.07
        For the attribute age        , chi2=0.308599072344        , and pval=0.5
        For the attribute sibsp        , chi2=2.58186537899        , and pval=0.
        For the attribute parch        , chi2=10.0974991118        , and pval=0.
        For the attribute fare        , chi2=8.81917152221        , and pval=0.0
        For the attribute embarked        , chi2=4.16460364538        , and pval
f_regression:
        For the attribute pclass        , F=284.129532979        , and pval=1.57
        For the attribute sex        , F=56.9535623581        , and pval=1.10392
        For the attribute age        , F=376.857543848        , and pval=2.92484
        For the attribute sibsp        , F=54.6074317492        , and pval=3.393
        For the attribute parch        , F=103.639800789        , and pval=4.234
```

```
        For the attribute fare        , F=306.822356392        , and pval=3.0574
        For the attribute embarked        , F=278.555265732        , and pval=1.
f_classif:
        For the attribute pclass        , F=115.031272188        , and pval=2.53
        For the attribute sex        , F=372.405723602        , and pval=1.40606
        For the attribute age        , F=4.35351608908        , and pval=0.03721
        For the attribute sibsp        , F=1.11057220411        , and pval=0.292
        For the attribute parch        , F=5.9634638366        , and pval=0.0147
        For the attribute fare        , F=63.030764228        , and pval=6.12018
        For the attribute embarked        , F=14.3305250028        , and pval=0.
```

4.4 Modèles

Le dossier csv/ contient beaucoup de prévisions pour l'ensemble de tests.

Plus de détails

Pour chaque script, sa documentation ne détaille pas comment sont obtenus les prédictions, mais souligne les faiblesses et la méthodologie de chaque modèle.

Scripts

Ces scripts génèrent des prévisions, avec une méthode d'apprentissage particulière.

Kaggle Module

Kaggle ML Project

This script introduces the first use of csv and numpy to manipulate the dataset of the Kaggle Titanic Project.
– I want first to read the datas,
– then to use them to compute some statistics about the datas,
– and also use them to plot some graphics : on the Plot_*.py files,

---

About This script is **still in development**. The reference page here on Kaggle (http ://www.kaggle.com/c/titanic-gettingStarted).

Documentation

The doc is available on-line, on one of my own pages :
- on the cr@ns (cr@ns) network besson/publis/kaggle/ (http ://perso.crans.org/besson/publis/kaggle/),
- on the CS department at ENS de Cachan ~lbesson/publis/kaggle/ (http ://www.dptinfo.ens-cachan.fr/ lbesson/publis/kaggle/),

Copyrigths

`Kaggle.`**`csv_file_object`** **= <_csv.reader object at 0x38b1910>**
Load in the csv file

`Kaggle.`**`data`** **= array([[‘0’, ‘3’, ‘Braund, Mr. Owen Harris’, ..., ‘7.25’, ‘’, ‘S’], [‘1’, ‘1’, ‘Cumings, Mrs. John Br**
Then convert from a list to an array

`Kaggle.`**`names_columns_train`** **= [’survived’, ‘pclass’, ‘name’, ‘sex’, ‘age’, ‘sibsp’, ‘parch’, ‘ticket’, ‘fare’, ‘**
Nom des attributs.

`Kaggle.`**`survived`** **= 0**
ce qu’on veut réussir à prédire !

`Kaggle.`**`pclass`** **= 1**
moyenne + influence, cf Plot_pclass.py

`Kaggle.`**`name`** **= 2**
aucune influence ?

`Kaggle.`**`sex`** **= 3**
grande influence, cf Plot_gender.py

`Kaggle.`**`age`** **= 4**
moyenne influence, cf Plot_age.py

`Kaggle.`**`sibsp`** **= 5**
moyenne + influence, cf Plot_sibsp.py

`Kaggle.`**`parch`** **= 6**
moyenne + influence, cf Plot_parch.py

`Kaggle.`**`ticket`** **= 7**
complexe : numéro du ticket

`Kaggle.`**`fare`** **= 8**
grande influence, cf Plot_fare.py

`Kaggle.`**`cabin`** **= 9**
complexe : localisation dans le bateau

`Kaggle.`**`embarked`** **= 10**
moyenne - influence, cf Plot_port.py

`Kaggle.`**`number_passengers`** **= 891**
Nombre de données d’apprentissage

`Kaggle.`**`number_survived`** **= 342.0**
Nombre de survivants

`Kaggle.`**`number_dead`** **= 549.0**
Nombre de victimes

`Kaggle.`**`proportion_survivors`** **= 0.38383838383838381**
Proportion de survivants

`Kaggle.`**`proportion_women`** **= 0.35241301907968575**
Proportion de femmes

`Kaggle.`**`proportion_women_survived`** **= 0.7420382165605095**
Proportion de femmes ayant survécues

`Kaggle.`**`proportion_men`** **= 0.6475869809203143**
Proportion d'hommes

`Kaggle.`**`proportion_men_survived`** **= 0.18890814558058924**
Proportion d'hommes ayant survécus

`Kaggle.`**`proportion_s_survived`** **= 0.33695652173913043**
Proportion de survivants dans les passagers venant de Southampton

`Kaggle.`**`proportion_c_survived`** **= 0.5535714285714286**
Proportion de survivants dans les passagers venant de Cherbourg

`Kaggle.`**`proportion_q_survived`** **= 0.38961038961038963**
Proportion de survivants dans les passagers venant de Queenstown

`Kaggle.`**`proportion_known_ages`** **= 0.8013468013468014**
On ne connais pas l'âge de tous les passagers

`Kaggle.`**`age_min`** **= 0.41999999999999998**
Age minimum

`Kaggle.`**`age_max`** **= 80.0**
Age max

`Kaggle.`**`age_mean`** **= 29.69911764705882**
Age moyen, utilisé pour compléter les données

`Kaggle.`**`fare_min`** **= 0.0**
Prix min

`Kaggle.`**`fare_max`** **= 512.32920000000001**
Prix max

`Kaggle.`**`fare_mean`** **= 32.2042079685746**
Prix moyen, utilisé pour compléter les données

`Kaggle.`**`header`** **= ['pclass', 'name', 'sex', 'age', 'sibsp', 'parch', 'ticket', 'fare', 'cabin', 'embarked']**
Skip the fist line as it is a header

KaggleModel Module

Transformation des données.

Ce script permet de lire les données des deux fichiers train.csv, test.csv et de les transformer en matrics de flottants.

Conversion Les données lues depuis ces deux fichiers sont sous formes de numpy.array de **strings**. On doit les convertir en **float** pour apprendre.

Les deux difficultés sont pour le port d'embarcation et le genre. On convertit : 'male' et 'female' en nombre (1 ou 0).

Seulement, pour ces attributs, le problème sous-jacent est de la classification (un passager est un homme **OU** une femme), pas de régression (un passager n'est pas 71% une femme...).

Malgré ça, le fait qu'il y a de nombreux autres attributs fait qu'on pourra quand même passer nos données à notre algorithme de classification, sans que la présence de deux attributs "catégoricaux" et non numériques ne le perturbe.

Source doc de scikit-learn (http ://scikit-learn.org/)

Complétion Le problème principal est le manque de certaines données.

Par exemple, 20% des passengers n'ont pas d'âge dans l'ensemble d'apprentissage.

Et quelques passengers n'ont pas de "prix de tickets" dans l'ensemble de test.

On completera donc simplement par la *valeur moyenne* de cet attribut. Cette valeur est calculée dans le script Kaggle.

`KaggleModel.header` **= ['pclass', 'name', 'sex', 'age', 'sibsp', 'parch', 'ticket', 'fare', 'cabin', 'embarked']**
Skip the fist line as it is a header

`KaggleModel.table_sex` **= {'male' : 1, 'female' : 0}**
Table of translation : strings -> float, for the gender

`KaggleModel.table_embarked` **= {'Q' : 2, 'C' : 0, 'S' : 1}**
Table of translation : strings -> float, for the port

`KaggleModel.train_data` **= array([[0. , 3. , 1. , ..., 2. , 0.0915427 , 1.], [1. , 2. , 1. , ..., 1. , 0.05074862, 1.], [0**
Then convert from a list to an array of float numbers !

`KaggleModel.test_data` **= array([[3. , 1. , 0.43125 , ..., 0. , 0.01528158, 2.], [3. , 0. , 0.5875 , ..., 0. , 0.013663**
Then convert from a list to an array of float numbers !

KaggleStats Module

Some statistics (printed on the screen) about the datas.

Sortie du script

```
$ python KaggleStats.py
OK, I read this file, I have now a huge array (of size : (891, 11)) !
This array have 891 passengers, and the following attributes :
 the 0th attribute is survived.
 the values of this attribute are known for 100.00% of the passengers
 the 1th attribute is pclass.
 the values of this attribute are known for 100.00% of the passengers
 the 2th attribute is name.
 the values of this attribute are known for 100.00% of the passengers
 the 3th attribute is sex.
 the values of this attribute are known for 100.00% of the passengers
 the 4th attribute is age.
 the values of this attribute are known for 80.13% of the passengers
 the 5th attribute is sibsp.
 the values of this attribute are known for 100.00% of the passengers
 the 6th attribute is parch.
 the values of this attribute are known for 100.00% of the passengers
```

```
 the 7th attribute is ticket.
 the values of this attribute are known for 100.00% of the passengers
 the 8th attribute is fare.
 the values of this attribute are known for 100.00% of the passengers
 the 9th attribute is cabin.
 the values of this attribute are known for 22.90% of the passengers
 the 10th attribute is embarked.
 the values of this attribute are known for 99.78% of the passengers
This array contains 891 passengers...
and 342 passengers survived the Titanic accident in 1912 :(
(so 549 passengers died)
which gives a proportion of 0.3838 survivors.

Considering the women and men ...
The training data set contains 314 women,
The training data set contains 577 men.
Proportion of women who survived is 0.7420,
Proportion of men who survived is 0.1889.

Considering the three different embarcations points:
 * S = Southampton (EN) -> 1;
 * C = Cherbourg (FR) -> 0;
 * Q = Queenstown (EN) -> 2.

644 passengers took the Titanic at Southampton;
168 passengers took the Titanic at Cherbourg;
77 passengers took the Titanic at Queenstown.

Proportion of survivors, for the Southampton part, is 0.3370;
Proportion of survivors, for the Cherbourg part, is 0.5536;
Proportion of survivors, for the Queenstown part, is 0.3896.

Considering the age of the passengers...
The age is known for 80.13% of the passengers (so 714 passengers).
For them, the min age is 0.42.
For them, the max age is 80.00.
For them, the median age is 29.70.
The median age of the victims is 30.626.
The median age of the survivors is 28.344.
... so : no obvious link between age and survivance rate ?
Yeah, but maybe median age is not the best way to show it.

Considering the fare of the passengers...
The fare is known for 100.00% of the passengers (so 891 passengers).
For them, the min fare is 0.00.
For them, the max fare is 512.33.
For them, the median fare is 32.20.
The median fare of the victims is 22.118.
The median fare of the survivors is 48.395.
Opening the file 'test.csv'...
This array have 418 passengers, and the following attributes :
```

```
 the 0th attribute is pclass.
   the values of this attribute are known for 100.00% of the passengers
 the 1th attribute is name.
   the values of this attribute are known for 100.00% of the passengers
 the 2th attribute is sex.
   the values of this attribute are known for 100.00% of the passengers
 the 3th attribute is age.
   the values of this attribute are known for 79.43% of the passengers
 the 4th attribute is sibsp.
   the values of this attribute are known for 100.00% of the passengers
 the 5th attribute is parch.
   the values of this attribute are known for 100.00% of the passengers
 the 6th attribute is ticket.
   the values of this attribute are known for 100.00% of the passengers
 the 7th attribute is fare.
   the values of this attribute are known for 99.76% of the passengers
 the 8th attribute is cabin.
   the values of this attribute are known for 21.77% of the passengers
 the 9th attribute is embarked.
   the values of this attribute are known for 100.00% of the passengers
```

RandomForest Module

A random forest model.

The doc is here : http ://scikit-learn.org/dev/modules/generated/sklearn.ensemble.RandomForestClassifier.html

C'est le premier modèle que j'ai utilisé. Pour 5 arbres, Kaggle me donne 75.59% de réussites.

Sortie du script

```
$ python RandomForest.py
Opening the file 'train.csv' and 'test.csv'...
Find the best value for the meta parameter n_estimators, with 10 run for each...
Searching in the range : [1, 2, 4, 5, 8, 10, 20]...
Using the first part (67.00%, 596 passengers) of the training dataset as trainin
and the second part (33.00%, 295 passengers) as testing !
For 1 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 84.09%...
For 2 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 84.03%...
For 4 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 87.40%...
For 5 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 87.87%...
For 8 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 89.24%...
For 10 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 88.71%...
For 20 random tree(s), learning from the first part of the dataset...
```

```
... this value of n_estimators seems to have a (mean) quality = 90.25%...
With trying each of the following n_estimators ([1, 2, 4, 5, 8, 10, 20]), each 1
Creating the random forest (of 20 estimators)...
Learning...
 Proportion of perfect fitting for the training dataset = 97.31%
Predicting for the testing dataset
Prediction: wrote in the file csv/RandomForest_best.csv.
```

Résultats La soumission du résultat à Kaggle donne 77.38%.

`RandomForest.list_n_estimators` = **[1, 2, 4, 5, 8, 10, 20]**
: Espace de recherche

`RandomForest.Number_try` = **10**
: Nombre de tests utilisés pour méta-apprendre

`RandomForest.proportion_train` = **0.67**
: Proportion d'individus utilisés pour méta-apprendre.

`RandomForest.best_n_estimators` = **20**
: La valeur optimale trouvée pour le paramètre n_estimators

`RandomForest.score` = **97.194163860830528**
: The score for this classifier.

KNN Module

A K Nearest Neighbors model.

Comme on doit utiliser une distance dans R^d avec $d = 8$, il faut que les données fournies soient des nombres flottants, et soient complètes.

The doc is here : http ://scikit-learn.org/dev/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.nei

---

Sortie du script

```
$ python KNN.py
Opening the file 'train.csv' and 'test.csv'...
Find the best value for the meta parameter n_neighbors, with 10 run for each...
Searching in the range : xrange(1, 30)...
Using the first part (65.00%, 579 passengers) of the training dataset as trainin
and the second part (35.00%, 312 passengers) as testing !
For 1 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 86.18%...
For 2 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.05%...
For 3 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 88.43%...
For 4 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.74%...
```

```
For 5 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.74%...
For 6 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.56%...
For 7 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.05%...
For 8 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.22%...
For 9 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 86.70%...
For 10 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 86.87%...
For 11 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 86.70%...
For 12 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 86.70%...
For 13 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.05%...
For 14 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.39%...
For 15 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.22%...
For 16 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.74%...
For 17 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.56%...
For 18 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.74%...
For 19 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.74%...
For 20 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.56%...
For 21 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.91%...
For 22 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.74%...
For 23 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.74%...
For 24 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.91%...
For 25 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.74%...
For 26 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.74%...
For 27 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.74%...
For 28 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.56%...
For 29 Nearest Neighbors, learning from the first part of the dataset...
... this value of n_neighbors seems to have a (mean) quality = 87.22%...
With trying each of the following n_neighbors (xrange(1, 30)), each 10 times, th
```

```
Creating the classifier with the optimal value of n_neighbors.
Learning...
 Proportion of perfect fitting for the training dataset = 97.42%
Predicting for the testing dataset
Prediction: wrote in the file csv/KNN_best.csv.
For the attribut survived        , chi2=30.8736994366        , and pval=2.753785
For the attribut pclass        , chi2=92.7024469789        , and pval=6.07783826
For the attribut sex        , chi2=0.308599072344        , and pval=0.5785411254
For the attribut age        , chi2=2.58186537899        , and pval=0.10809421012
For the attribut sibsp        , chi2=10.0974991118        , and pval=0.001484706
For the attribut parch        , chi2=8.81917152221        , and pval=0.002980819
For the attribut fare        , chi2=4.16460364538        , and pval=0.0412770695
```

Résultats La soumission du résultat à Kaggle donne 71.70%.

`KNN.`**`list_n_neighbors`** **= xrange(1, 30)**
: Espace de recherche

`KNN.`**`Number_try`** **= 10**
: Nombre de tests utilisés pour méta-apprendre

`KNN.`**`proportion_train`** **= 0.65**
: Proportion d'individus utilisés pour méta-apprendre.

`KNN.`**`best_n_neighbors`** **= 28**
: La valeur optimale trouvée pour le paramètre n_estimators

`KNN.`**`score`** **= 97.418630751964088**
: The score for this classifier.

RNN Module

A Radius Nearest Neighbors model.

The doc is here : http ://scikit-learn.org/dev/modules/generated/sklearn.neighbors.RadiusNeighborsClassifier.html#sklear

Limitation Le rayon doit être grand pour que chaque point ait au moins un voisin, sinon l'algorithme râle !

À propos Dans ce script, j'utilise sklearn.grid_search pour explorer automatiquement un ensemble de méta-paramètre.

Sortie du script

```
$ python RNN.py
Opening the file 'train.csv' and 'test.csv'...
Searching for parameters in {'radius': [100.0, 1000.0, 10000.0, 500.0, 5000.0, 5
Learning...
 Proportion of perfect fitting for the training dataset = 61.62%
```

```
The best parameters are : {'radius': 100.0, 'leaf_size': 1}.
Predicting for the testing dataset
Prediction: wrote in the file csv/RNN_best.csv.
```

`RNN.score` = **61.616161616161612**
The score for this classifier.

DecisionTree Module

A Decision Tree model.

The doc is here : http ://scikit-learn.org/dev/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.Dec

---

Sortie du script

```
$ python DecisionTree.py
Opening the file 'train.csv' and 'test.csv'...
Find the best value for the meta parameter max_depth, with 10 run for each...
Searching in the range : xrange(1, 30)...
Using the first part (67.00%, 596 passengers) of the training dataset as trainin
and the second part (33.00%, 295 passengers) as testing !
For max_depth=1, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 77.35%...
For max_depth=2, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 77.35%...
For max_depth=3, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 81.04%...
For max_depth=4, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 81.71%...
For max_depth=5, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 82.89%...
For max_depth=6, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 84.23%...
For max_depth=7, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 84.90%...
For max_depth=8, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 84.40%...
For max_depth=9, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 85.57%...
For max_depth=10, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 85.91%...
For max_depth=11, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 86.24%...
For max_depth=12, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 86.07%...
For max_depth=13, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 87.08%...
For max_depth=14, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 87.42%...
```

```
For max_depth=15, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 87.92%...
For max_depth=16, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 87.92%...
For max_depth=17, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 88.26%...
For max_depth=18, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 88.26%...
For max_depth=19, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 88.26%...
For max_depth=20, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 88.26%...
For max_depth=21, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 88.26%...
For max_depth=22, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 88.26%...
For max_depth=23, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 88.26%...
For max_depth=24, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 88.26%...
For max_depth=25, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 88.26%...
For max_depth=26, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 88.26%...
For max_depth=27, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 88.26%...
For max_depth=28, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 88.26%...
For max_depth=29, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 88.26%...
With trying each of the following max_depth (xrange(1, 30)), each 10 times, the
Find the best value for the meta parameter min_samples_split, with 10 run for ea
Searching in the range : xrange(1, 10)...
Using the first part (67.00%, 596 passengers) of the training dataset as trainin
and the second part (33.00%, 295 passengers) as testing !
For min_samples_split=1, learning from the first part of the dataset...
... this value of min_samples_split seems to have a (mean) quality = 88.26%...
For min_samples_split=2, learning from the first part of the dataset...
... this value of min_samples_split seems to have a (mean) quality = 88.26%...
For min_samples_split=3, learning from the first part of the dataset...
... this value of min_samples_split seems to have a (mean) quality = 87.25%...
For min_samples_split=4, learning from the first part of the dataset...
... this value of min_samples_split seems to have a (mean) quality = 87.08%...
For min_samples_split=5, learning from the first part of the dataset...
... this value of min_samples_split seems to have a (mean) quality = 86.74%...
For min_samples_split=6, learning from the first part of the dataset...
... this value of min_samples_split seems to have a (mean) quality = 86.58%...
For min_samples_split=7, learning from the first part of the dataset...
... this value of min_samples_split seems to have a (mean) quality = 86.07%...
For min_samples_split=8, learning from the first part of the dataset...
... this value of min_samples_split seems to have a (mean) quality = 85.91%...
```

```
For min_samples_split=9, learning from the first part of the dataset...
... this value of min_samples_split seems to have a (mean) quality = 85.23%...
With trying each of the following min_samples_split (xrange(1, 10)), each 10 tim
Find the best value for the meta parameter min_samples_leaf, with 10 run for eac
Searching in the range : xrange(1, 10)...
Using the first part (67.00%, 596 passengers) of the training dataset as trainin
and the second part (33.00%, 295 passengers) as testing !
For min_samples_leaf=1, learning from the first part of the dataset...
... this value of min_samples_leaf seems to have a (mean) quality = 88.26%...
For min_samples_leaf=2, learning from the first part of the dataset...
... this value of min_samples_leaf seems to have a (mean) quality = 83.89%...
For min_samples_leaf=3, learning from the first part of the dataset...
... this value of min_samples_leaf seems to have a (mean) quality = 83.89%...
For min_samples_leaf=4, learning from the first part of the dataset...
... this value of min_samples_leaf seems to have a (mean) quality = 84.23%...
For min_samples_leaf=5, learning from the first part of the dataset...
... this value of min_samples_leaf seems to have a (mean) quality = 82.21%...
For min_samples_leaf=6, learning from the first part of the dataset...
... this value of min_samples_leaf seems to have a (mean) quality = 82.38%...
For min_samples_leaf=7, learning from the first part of the dataset...
... this value of min_samples_leaf seems to have a (mean) quality = 80.70%...
For min_samples_leaf=8, learning from the first part of the dataset...
... this value of min_samples_leaf seems to have a (mean) quality = 81.88%...
For min_samples_leaf=9, learning from the first part of the dataset...
... this value of min_samples_leaf seems to have a (mean) quality = 81.38%...
With trying each of the following min_samples_leaf (xrange(1, 10)), each 10 time
Creating the classifier, with optimal parameters.
Learning...
 Proportion of perfect fitting for the training dataset = 97.87%
Predicting for the testing dataset
Prediction: wrote in the file csv/DecisionTree_best.csv.
```

Résultats La soumission du résultat à Kaggle donne 76.07%.

`DecisionTree.`**`list_max_depth = xrange(1, 30)`**
: Espace de recherche

`DecisionTree.`**`best_max_depth = 18`**
: La valeur optimale trouvée pour le paramètre max_depth

`DecisionTree.`**`best_min_samples_split = 1`**
: La valeur optimale trouvée pour le paramètre min_samples_split

`DecisionTree.`**`Number_try = 10`**
: Nombre de tests utilisés pour méta-apprendre

`DecisionTree.`**`proportion_train = 0.67`**
: Proportion d'individus utilisés pour méta-apprendre.

`DecisionTree.`**`best_min_samples_leaf = 1`**
: La valeur optimale trouvée pour le paramètre min_samples_leaf

`DecisionTree.`**`score`** **= 98.092031425364752**
The score for this classifier.

ExtraTrees Module

An extra-trees model.

The doc is here : http ://scikit-learn.org/dev/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html

---

Sortie du script

```
$ python ExtraTrees.py
Opening the file 'train.csv' and 'test.csv'...
Find the best value for the meta parameter n_estimators, with 5 run for each...
Searching in the range : [1, 2, 3, 4, 5, 8, 10, 20, 30, 100]...
Using the first part (66.00%, 588 passengers) of the training dataset as trainin
and the second part (34.00%, 303 passengers) as testing !
For 1 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 87.41%...
For 2 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 88.44%...
For 3 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 87.89%...
For 4 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 88.06%...
For 5 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 88.64%...
For 8 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 87.82%...
For 10 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 88.54%...
For 20 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 88.40%...
For 30 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 88.54%...
For 100 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 88.23%...
With trying each of the following n_estimators ([1, 2, 3, 4, 5, 8, 10, 20, 30, 1
Creating the classifier (of 100 estimators)...
Learning...
 Proportion of perfect fitting for the training dataset = 98.20%
Predicting for the testing dataset
Prediction: wrote in the file csv/ExtraTrees_best.csv.
```

Résultats La soumission du résultat à Kaggle donne 75.59%.

`ExtraTrees.`**`list_n_estimators`** **= [1, 2, 3, 4, 5, 8, 10, 20, 30, 100]**
Espace de recherche

`ExtraTrees`**`.Number_try = 5`**
Nombre de tests utilisés pour méta-apprendre

`ExtraTrees`**`.proportion_train = 0.66`**
Proportion d'individus utilisés pour méta-apprendre.

`ExtraTrees`**`.best_n_estimators = 100`**
La valeur optimale trouvée pour le paramètre n_estimators

`ExtraTrees`**`.score = 98.204264870931539`**
The score for this classifier.

SVM Module

A SVM model.

The doc is here : http ://scikit-learn.org/dev/modules/svm.html#svm and there : http ://scikit-learn.org/dev/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

Méta-apprentissage Pour régler les méta-paramètres des méthodes de classification, dans chaque script, je sépare les données de *train.csv* en apprentissage et test (selon une proportion de 55% à 85%).

À faire

Utiliser sklearn.cross_validation.train_test_split (http ://scikit-learn.org/dev/modules/generated/sklearn.cross_validation.
pour séparer les données de 'train.csv' en train et test.

Pour les SVM, je règle la constante C de régularisation.

Il y a d'autres méta-paramètres, que je n'ai pas encore cherché à modifier.

Sortie du script

```
$ python SVM.py
Opening the file 'train.csv' and 'test.csv'...
Find the best value for the meta parameter C, with 5 run for each...
Searching in the range : [1e-05, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 100
Using the first part (67.00%, 596 passengers) of the training dataset as trainin
and the second part (33.00%, 295 passengers) as testing !
For C=1e-05, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 61.91%...
For C=0.0001, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 61.91%...
For C=0.001, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 61.91%...
For C=0.01, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 61.91%...
For C=0.1, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 80.03%...
For C=1, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 80.03%...
```

```
For C=10, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 81.38%...
For C=100, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 81.21%...
For C=1000, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 81.38%...
For C=10000, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 82.21%...
For C=100000, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 81.04%...
For C=5e-06, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 61.91%...
For C=5e-05, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 61.91%...
For C=0.0005, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 61.91%...
For C=0.005, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 61.91%...
For C=0.05, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 78.19%...
For C=0.5, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 79.70%...
For C=5.0, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 81.04%...
For C=50.0, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 80.87%...
For C=500.0, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 81.38%...
For C=5000.0, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 81.71%...
For C=50000.0, learning from the first part of the dataset...
... this value of C seems to have a (mean) quality = 80.87%...
With trying each of the following C ([1e-05, 0.0001, 0.001, 0.01, 0.1, 1, 10, 10
Creating the classifier with the optimal value of C.
Learning...
 Proportion of perfect fitting for the training dataset = 87.21%
Predicting for the testing dataset
Prediction: wrote in the file csv/SVM_best.csv.
```

Résultats La soumission du résultat à Kaggle donne 65.07%.

SVM.**list_C = [1e-05, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000, 5e-06, 5e-05, 0.0005, 0.005, 0.05, 0**
: Espace de recherche

SVM.**Number_try = 5**
: Nombre de tests utilisés pour méta-apprendre

SVM.**proportion_train = 0.67**
: Proportion d'individus utilisés pour méta-apprendre.

SVM.**best_C = 500.0**
: La valeur optimale trouvée pour le paramètre n_estimators

`SVM.score` = **85.746352413019082**
The score for this classifier.

AdaBoost Module

A AdaBoost model.

The doc is here : http ://scikit-learn.org/dev/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

Overfitting Je fais une vérification de l'overfitting via la fonction *Score*, qui permet de savoir combien des passengers dans l'ensemble d'apprentissage sont classés dans la bonne catégorie par le classifieur.

Ce score devrait être < 95%, sinon c'est "louche".

Ce paragraphe s'applique à tous les autres classifieurs présentés après, mais il n'est pas répété.

Sortie du script

```
$ python AdaBoost.py
Opening the file 'train.csv' and 'test.csv'...
Find the best value for the meta parameter n_estimators, with 5 run for each...
Searching in the range : [1, 5, 10, 20, 30, 40, 50, 60, 75, 82, 100]...
Using the first part (67.00%, 596 passengers) of the training dataset as trainin
and the second part (33.00%, 295 passengers) as testing !
For 1 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 78.86%...
For 5 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 81.54%...
For 10 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 79.53%...
For 20 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 81.04%...
For 30 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 82.55%...
For 40 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 81.71%...
For 50 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 81.54%...
For 60 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 82.05%...
For 75 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 81.71%...
For 82 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 82.21%...
For 100 random tree(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 81.88%...
With trying each of the following n_estimators ([1, 5, 10, 20, 30, 40, 50, 60, 7
Creating the adaboost classifier (of 100 estimators)...
Learning...
 Proportion of perfect fitting for the training dataset = 82.60%
```

```
Predicting for the testing dataset
Prediction: wrote in the file csv/AdaBoost_best.csv.
```

Résultats La soumission du résultat à Kaggle donne 75.12%.

`AdaBoost.list_n_estimators` = **[1, 5, 10, 20, 30, 40, 50, 60, 75, 82, 100]**
Espace de recherche

`AdaBoost.Number_try` = **5**
Nombre de tests utilisés pour méta-apprendre

`AdaBoost.proportion_train` = **0.67**
Proportion d'individus utilisés pour méta-apprendre.

`AdaBoost.best_n_estimators` = **100**
La valeur optimale trouvée pour le paramètre n_estimators

`AdaBoost.score` = **84.062850729517393**
The score for this classifier.

GradientBoosting Module

A Gradient Boosting model.

Semble être bien performant, mais **très** lent !

The doc is here : http ://scikit-learn.org/dev/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#sklear

---

Sortie du script

```
$ python GradientBoosting.py
Opening the file 'train.csv' and 'test.csv'...
Find the best value for the meta parameter n_estimators, with 10 run for each...
Searching in the range : [1, 2, 5, 7, 10, 15, 20, 50, 100]...
Using the first part (68.00%, 605 passengers) of the training dataset as trainin
and the second part (32.00%, 286 passengers) as testing !
For 1 estimator(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 61.79%...
For 2 estimator(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 67.98%...
For 5 estimator(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 81.06%...
For 7 estimator(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 82.02%...
For 10 estimator(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 82.18%...
For 15 estimator(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 83.31%...
For 20 estimator(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 83.47%...
```

```
For 50 estimator(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 85.67%...
For 100 estimator(s), learning from the first part of the dataset...
... this value of n_estimators seems to have a (mean) quality = 86.64%...
With trying each of the following n_estimators ([1, 2, 5, 7, 10, 15, 20, 50, 100
Find the best value for the meta parameter max_depth, with 10 run for each...
Searching in the range : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 1000]...
Using the first part (67.00%, 596 passengers) of the training dataset as trainin
and the second part (33.00%, 295 passengers) as testing !
For random trees with depth <= 1, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 81.43%...
For random trees with depth <= 2, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 83.74%...
For random trees with depth <= 3, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 86.81%...
For random trees with depth <= 4, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 88.67%...
For random trees with depth <= 5, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 89.60%...
For random trees with depth <= 6, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 89.45%...
For random trees with depth <= 7, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 89.14%...
For random trees with depth <= 8, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 88.93%...
For random trees with depth <= 9, learning from the first part of the dataset...
... this value of max_depth seems to have a (mean) quality = 89.24%...
For random trees with depth <= 10, learning from the first part of the dataset..
... this value of max_depth seems to have a (mean) quality = 89.23%...
For random trees with depth <= 15, learning from the first part of the dataset..
... this value of max_depth seems to have a (mean) quality = 88.20%...
For random trees with depth <= 20, learning from the first part of the dataset..
... this value of max_depth seems to have a (mean) quality = 88.37%...
For random trees with depth <= 1000, learning from the first part of the dataset
... this value of max_depth seems to have a (mean) quality = 88.66%...
With trying each of the following max_depth ([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15,
Creating the Gradient Boosting classifier with best meta parameters.
Learning...
 Proportion of perfect fitting for the training dataset = 95.85%
Predicting for the testing dataset
Prediction: wrote in the file csv/GradientBoosting_best.csv.
```

Résultats La soumission du résultat à Kaggle donne 75.59%.

`GradientBoosting.`**`list_n_estimators`** **= [1, 2, 5, 7, 10, 15, 20, 50, 100]**
Espace de recherche

`GradientBoosting.`**`best_n_estimators`** **= 100**
La valeur optimale trouvée pour le paramètre n_estimators

`GradientBoosting.`**`list_max_depth`** **= [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 1000]**
Espace de recherche

`GradientBoosting.`**`Number_try`** **= 10**
Nombre de tests utilisés pour méta-apprendre (faible car l'algo est lent)

`GradientBoosting.`**`proportion_train`** **= 0.67**
Proportion d'individus utilisés pour méta-apprendre.

`GradientBoosting.`**`best_max_depth`** **= 10**
La valeur optimale trouvée pour le paramètre n_estimators

`GradientBoosting.`**`score`** **= 98.204264870931539**
The score for this classifier.

DummyClassifier Module

A dummy model, with 3 strategies :
– stratified ;
– most_frequent ;
– uniform.
The doc is here : http ://scikit-learn.org/dev/modules/generated/sklearn.dummy.DummyClassifier.html#sklearn.dummy.D

Sortie du script

```
$ python DummyClassifier.py
Opening the file 'train.csv' and 'test.csv'...
Find the best value for the meta parameter strategy, with 100 run for each...
Searching in the range : ['stratified', 'most_frequent', 'uniform']...
Using the first part (75.00%, 668 passengers) of the training dataset as trainin
and the second part (25.00%, 223 passengers) as testing !
For the strategy stratified, learning from the first part of the dataset...
... this value of strategy seems to have a (mean) quality = 52.34%...
For the strategy most_frequent, learning from the first part of the dataset...
... this value of strategy seems to have a (mean) quality = 60.03%...
For the strategy uniform, learning from the first part of the dataset...
... this value of strategy seems to have a (mean) quality = 50.15%...
With trying each of the following strategy (['stratified', 'most_frequent', 'uni
Creating the Dummy Classifier classifier with best meta parameters.
Learning...
 Proportion of perfect fitting for the training dataset = 61.62%
Predicting for the testing dataset
Prediction: wrote in the file csv/Dummy_best.csv.
```

Résultats La soumission du résultat à Kaggle donne ? ?. ? ?% (pas encore fait).

`DummyClassifier.`**`list_strategy`** **= ['stratified', 'most_frequent', 'uniform']**
Espace de recherche

`DummyClassifier.`**`Number_try`** **= 100**
Nombre de tests utilisés pour méta-apprendre

`DummyClassifier.`**`proportion_train`** **= 0.75**
Proportion d'individus utilisés pour méta-apprendre.

`DummyClassifier.`**`best_strategy`** **= 'most_frequent'**
La valeur optimale trouvée pour le paramètre strategy

`DummyClassifier.`**`score`** **= 61.616161616161612**
The score for this classifier.

NuSVC Module

A Nu-Support Vector Classification. model.

The doc is here : http ://scikit-learn.org/dev/modules/generated/sklearn.svm.NuSVC.html#sklearn.svm.NuSVC

Sortie du script

```
$ python NuSVC.py
Opening the file 'train.csv' and 'test.csv'...
Find the best value for the meta parameter nu, with 5 run for each...
Searching in the range : [0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09,
Using the first part (67.00%, 596 passengers) of the training dataset as trainin
and the second part (33.00%, 295 passengers) as testing !
For nu=0.01, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 51.01%...
For nu=0.02, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 42.45%...
For nu=0.03, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 40.77%...
For nu=0.04, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 68.79%...
For nu=0.05, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 38.09%...
For nu=0.06, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 55.20%...
For nu=0.07, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 36.41%...
For nu=0.08, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 54.53%...
For nu=0.09, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 38.42%...
For nu=0.1, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 51.85%...
For nu=0.11, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 73.99%...
For nu=0.12, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 69.63%...
For nu=0.13, learning from the first part of the dataset...
```

```
... this value of nu seems to have a (mean) quality = 66.61%...
For nu=0.14, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 65.10%...
For nu=0.15, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 77.01%...
For nu=0.16, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 76.01%...
For nu=0.17, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 67.11%...
For nu=0.18, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 82.05%...
For nu=0.19, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 78.02%...
For nu=0.2, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 76.85%...
For nu=0.21, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 81.88%...
For nu=0.22, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 83.22%...
For nu=0.23, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 82.21%...
For nu=0.24, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 82.55%...
For nu=0.25, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 83.72%...
For nu=0.26, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 80.37%...
For nu=0.27, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 82.55%...
For nu=0.28, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 83.72%...
For nu=0.29, learning from the first part of the dataset...
... this value of nu seems to have a (mean) quality = 83.22%...
With trying each of the following nu ([0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07,
Creating the classifier with the optimal value of nu.
Learning...
 Proportion of perfect fitting for the training dataset = 84.40%
Predicting for the testing dataset
Prediction: wrote in the file csv/NuSVC_best.csv.
```

Résultats La soumission du résultat à Kaggle donne ? ?. ? ?%.

`NuSVC.`**`list_nu = [0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.1`**
Espace de recherche

`NuSVC.`**`Number_try = 5`**
Nombre de tests utilisés pour méta-apprendre

`NuSVC.`**`proportion_train = 0.67`**
Proportion d'individus utilisés pour méta-apprendre.

`NuSVC.best_nu` = **0.29**
La valeur optimale trouvée pour le paramètre n_estimators

`NuSVC.score` = **85.970819304152641**
The score for this classifier.

QDA Module

A Quadratic Discriminant Analysis (QDA)

The doc is here : http ://scikit-learn.org/dev/modules/generated/sklearn.qda.QDA.html#sklearn.qda.QDA

Sortie du script

```
$ python QDA.py
Opening the file 'train.csv' and 'test.csv'...
Learning...
 Proportion of perfect fitting for the training dataset = 80.58%
Class priors (sum to 1) = [ 0.61616162  0.38383838].
        For the attribute pclass        , mean=[ 2.53187614  1.9502924 ].
        For the attribute sex        , mean=[ 0.85245902  0.31871345].
        For the attribute age        , mean=[ 0.38018875  0.35687223].
        For the attribute sibsp        , mean=[ 0.55373406  0.47368421].
        For the attribute parch        , mean=[ 0.32969035  0.46491228].
        For the attribute fare        , mean=[ 0.04317124  0.09446154].
        For the attribute embarked        , mean=[ 0.94899818  0.81578947].
 Score for test part = 81.63%
 Score for test part = 81.97%
 Score for test part = 74.15%
 Score for test part = 79.59%
 Score for test part = 83.67%
Predicting for the testing dataset
Prediction: wrote in the file csv/QDA_best.csv.
```

`QDA.score` = **77.210884353741491**
The score for this classifier.

4.5 Vote Module

Un voteur.

Ce script lit toutes les prédictions déjà réalisées, et fait un simple vote pour tenter de les améliorer !

Sortie du script

```
$ python Vote.py
Opening the file 'train.csv' and 'test.csv'...
Voting, using the files ['SVM_best.csv', 'RandomForest_best.csv', 'ExtraTrees_be
For the file SVM_best.csv, re-constructing the prediction.
        the prediction said 0.32% of victims...
For the file RandomForest_best.csv, re-constructing the prediction.
        the prediction said 0.35% of victims...
For the file ExtraTrees_best.csv, re-constructing the prediction.
        the prediction said 0.38% of victims...
For the file GradientBoosting_best.csv, re-constructing the prediction.
        the prediction said 0.35% of victims...
For the file KNN_best.csv, re-constructing the prediction.
        the prediction said 0.36% of victims...
For the file DecisionTree_best.csv, re-constructing the prediction.
        the prediction said 0.39% of victims...
For the file AdaBoost_best.csv, re-constructing the prediction.
        the prediction said 0.38% of victims...
For the file QDA_best.csv, re-constructing the prediction.
        the prediction said 0.40% of victims...
For the file RNN_best.csv, re-constructing the prediction.
        the prediction said 0.00% of victims...
For the file Dummy_best.csv, re-constructing the prediction.
        the prediction said 0.00% of victims...
For the file NuSVC_best.csv, re-constructing the prediction.
        the prediction said 0.31% of victims...
Predicting for the testing dataset
  for the 0th passenger, the vote of all predictions is 0.0.
  for the 1th passenger, the vote of all predictions is 0.0909090909091.
  for the 2th passenger, the vote of all predictions is 0.181818181818.
  for the 3th passenger, the vote of all predictions is 0.272727272727.
  for the 4th passenger, the vote of all predictions is 0.454545454545.
  for the 5th passenger, the vote of all predictions is 0.0.
  for the 6th passenger, the vote of all predictions is 0.363636363636.
  for the 7th passenger, the vote of all predictions is 0.0.
  for the 8th passenger, the vote of all predictions is 0.727272727273.
  for the 9th passenger, the vote of all predictions is 0.0.
  for the 10th passenger, the vote of all predictions is 0.0.
  for the 11th passenger, the vote of all predictions is 0.0909090909091.
  for the 12th passenger, the vote of all predictions is 0.818181818182.
  for the 13th passenger, the vote of all predictions is 0.0.
  for the 14th passenger, the vote of all predictions is 0.818181818182.
  for the 15th passenger, the vote of all predictions is 0.818181818182.
  for the 16th passenger, the vote of all predictions is 0.0.
  for the 17th passenger, the vote of all predictions is 0.454545454545.
  for the 18th passenger, the vote of all predictions is 0.454545454545.
  for the 19th passenger, the vote of all predictions is 0.363636363636.
  for the 20th passenger, the vote of all predictions is 0.545454545455.
  for the 21th passenger, the vote of all predictions is 0.545454545455.
  for the 22th passenger, the vote of all predictions is 0.818181818182.
```

```
for the 23th passenger, the vote of all predictions is 0.636363636364.
for the 24th passenger, the vote of all predictions is 0.636363636364.
for the 25th passenger, the vote of all predictions is 0.0.
for the 26th passenger, the vote of all predictions is 0.818181818182.
for the 27th passenger, the vote of all predictions is 0.272727272727.
for the 28th passenger, the vote of all predictions is 0.454545454545.
for the 29th passenger, the vote of all predictions is 0.0.
for the 30th passenger, the vote of all predictions is 0.0.
for the 31th passenger, the vote of all predictions is 0.0.
for the 32th passenger, the vote of all predictions is 0.363636363636.
for the 33th passenger, the vote of all predictions is 0.0909090909091.
for the 34th passenger, the vote of all predictions is 0.636363636364.
for the 35th passenger, the vote of all predictions is 0.0909090909091.
for the 36th passenger, the vote of all predictions is 0.181818181818.
for the 37th passenger, the vote of all predictions is 0.272727272727.
for the 38th passenger, the vote of all predictions is 0.0.
for the 39th passenger, the vote of all predictions is 0.454545454545.
for the 40th passenger, the vote of all predictions is 0.181818181818.
for the 41th passenger, the vote of all predictions is 0.0909090909091.
for the 42th passenger, the vote of all predictions is 0.0.
for the 43th passenger, the vote of all predictions is 0.818181818182.
for the 44th passenger, the vote of all predictions is 0.818181818182.
for the 45th passenger, the vote of all predictions is 0.0.
for the 46th passenger, the vote of all predictions is 0.0909090909091.
for the 47th passenger, the vote of all predictions is 0.0.
for the 48th passenger, the vote of all predictions is 0.636363636364.
for the 49th passenger, the vote of all predictions is 0.545454545455.
for the 50th passenger, the vote of all predictions is 0.0909090909091.
for the 51th passenger, the vote of all predictions is 0.0.
for the 52th passenger, the vote of all predictions is 0.818181818182.
for the 53th passenger, the vote of all predictions is 0.727272727273.
for the 54th passenger, the vote of all predictions is 0.0909090909091.
for the 55th passenger, the vote of all predictions is 0.0.
for the 56th passenger, the vote of all predictions is 0.0.
for the 57th passenger, the vote of all predictions is 0.0.
for the 58th passenger, the vote of all predictions is 0.0.
for the 59th passenger, the vote of all predictions is 0.818181818182.
for the 60th passenger, the vote of all predictions is 0.0.
for the 61th passenger, the vote of all predictions is 0.0909090909091.
for the 62th passenger, the vote of all predictions is 0.0.
for the 63th passenger, the vote of all predictions is 0.727272727273.
for the 64th passenger, the vote of all predictions is 0.545454545455.
for the 65th passenger, the vote of all predictions is 0.818181818182.
for the 66th passenger, the vote of all predictions is 0.636363636364.
for the 67th passenger, the vote of all predictions is 0.0.
for the 68th passenger, the vote of all predictions is 0.181818181818.
for the 69th passenger, the vote of all predictions is 0.636363636364.
for the 70th passenger, the vote of all predictions is 0.727272727273.
for the 71th passenger, the vote of all predictions is 0.0.
for the 72th passenger, the vote of all predictions is 0.272727272727.
for the 73th passenger, the vote of all predictions is 0.0.
```

```
for the 74th passenger, the vote of all predictions is 0.818181818182.
for the 75th passenger, the vote of all predictions is 0.181818181818.
for the 76th passenger, the vote of all predictions is 0.0.
for the 77th passenger, the vote of all predictions is 0.818181818182.
for the 78th passenger, the vote of all predictions is 0.0.
for the 79th passenger, the vote of all predictions is 0.727272727273.
for the 80th passenger, the vote of all predictions is 0.545454545455.
for the 81th passenger, the vote of all predictions is 0.181818181818.
for the 82th passenger, the vote of all predictions is 0.181818181818.
for the 83th passenger, the vote of all predictions is 0.0.
for the 84th passenger, the vote of all predictions is 0.0909090909091.
for the 85th passenger, the vote of all predictions is 0.0.
for the 86th passenger, the vote of all predictions is 0.727272727273.
for the 87th passenger, the vote of all predictions is 0.363636363636.
for the 88th passenger, the vote of all predictions is 0.636363636364.
for the 89th passenger, the vote of all predictions is 0.727272727273.
for the 90th passenger, the vote of all predictions is 0.363636363636.
for the 91th passenger, the vote of all predictions is 0.0.
for the 92th passenger, the vote of all predictions is 0.727272727273.
for the 93th passenger, the vote of all predictions is 0.0.
for the 94th passenger, the vote of all predictions is 0.272727272727.
for the 95th passenger, the vote of all predictions is 0.0.
for the 96th passenger, the vote of all predictions is 0.818181818182.
for the 97th passenger, the vote of all predictions is 0.0909090909091.
for the 98th passenger, the vote of all predictions is 0.454545454545.
for the 99th passenger, the vote of all predictions is 0.0909090909091.
for the 100th passenger, the vote of all predictions is 0.818181818182.
for the 101th passenger, the vote of all predictions is 0.0.
for the 102th passenger, the vote of all predictions is 0.0.
for the 103th passenger, the vote of all predictions is 0.0.
for the 104th passenger, the vote of all predictions is 0.636363636364.
for the 105th passenger, the vote of all predictions is 0.0.
for the 106th passenger, the vote of all predictions is 0.0.
for the 107th passenger, the vote of all predictions is 0.0.
for the 108th passenger, the vote of all predictions is 0.0.
for the 109th passenger, the vote of all predictions is 0.0.
for the 110th passenger, the vote of all predictions is 0.0.
for the 111th passenger, the vote of all predictions is 0.636363636364.
for the 112th passenger, the vote of all predictions is 0.727272727273.
for the 113th passenger, the vote of all predictions is 0.545454545455.
for the 114th passenger, the vote of all predictions is 0.818181818182.
for the 115th passenger, the vote of all predictions is 0.0.
for the 116th passenger, the vote of all predictions is 0.0.
for the 117th passenger, the vote of all predictions is 0.818181818182.
for the 118th passenger, the vote of all predictions is 0.454545454545.
for the 119th passenger, the vote of all predictions is 0.818181818182.
for the 120th passenger, the vote of all predictions is 0.818181818182.
for the 121th passenger, the vote of all predictions is 0.0.
for the 122th passenger, the vote of all predictions is 0.818181818182.
for the 123th passenger, the vote of all predictions is 0.0.
for the 124th passenger, the vote of all predictions is 0.0.
```

```
for the 125th passenger, the vote of all predictions is 0.818181818182.
for the 126th passenger, the vote of all predictions is 0.0.
for the 127th passenger, the vote of all predictions is 0.818181818182.
for the 128th passenger, the vote of all predictions is 0.0.
for the 129th passenger, the vote of all predictions is 0.0.
for the 130th passenger, the vote of all predictions is 0.363636363636.
for the 131th passenger, the vote of all predictions is 0.0909090909091.
for the 132th passenger, the vote of all predictions is 0.0909090909091.
for the 133th passenger, the vote of all predictions is 0.0.
for the 134th passenger, the vote of all predictions is 0.0.
for the 135th passenger, the vote of all predictions is 0.0.
for the 136th passenger, the vote of all predictions is 0.0.
for the 137th passenger, the vote of all predictions is 0.0.
for the 138th passenger, the vote of all predictions is 0.181818181818.
for the 139th passenger, the vote of all predictions is 0.181818181818.
for the 140th passenger, the vote of all predictions is 0.0.
for the 141th passenger, the vote of all predictions is 0.818181818182.
for the 142th passenger, the vote of all predictions is 0.272727272727.
for the 143th passenger, the vote of all predictions is 0.0.
for the 144th passenger, the vote of all predictions is 0.363636363636.
for the 145th passenger, the vote of all predictions is 0.0.
for the 146th passenger, the vote of all predictions is 0.0.
for the 147th passenger, the vote of all predictions is 0.0.
for the 148th passenger, the vote of all predictions is 0.0909090909091.
for the 149th passenger, the vote of all predictions is 0.0.
for the 150th passenger, the vote of all predictions is 0.818181818182.
for the 151th passenger, the vote of all predictions is 0.0.
for the 152th passenger, the vote of all predictions is 0.0.
for the 153th passenger, the vote of all predictions is 0.636363636364.
for the 154th passenger, the vote of all predictions is 0.0.
for the 155th passenger, the vote of all predictions is 0.0.
for the 156th passenger, the vote of all predictions is 0.818181818182.
for the 157th passenger, the vote of all predictions is 0.363636363636.
for the 158th passenger, the vote of all predictions is 0.363636363636.
for the 159th passenger, the vote of all predictions is 0.727272727273.
for the 160th passenger, the vote of all predictions is 0.727272727273.
for the 161th passenger, the vote of all predictions is 0.454545454545.
for the 162th passenger, the vote of all predictions is 0.818181818182.
for the 163th passenger, the vote of all predictions is 0.0.
for the 164th passenger, the vote of all predictions is 0.0.
for the 165th passenger, the vote of all predictions is 0.363636363636.
for the 166th passenger, the vote of all predictions is 0.272727272727.
for the 167th passenger, the vote of all predictions is 0.0.
for the 168th passenger, the vote of all predictions is 0.818181818182.
for the 169th passenger, the vote of all predictions is 0.272727272727.
for the 170th passenger, the vote of all predictions is 0.0.
for the 171th passenger, the vote of all predictions is 0.0.
for the 172th passenger, the vote of all predictions is 0.0.
for the 173th passenger, the vote of all predictions is 0.0.
for the 174th passenger, the vote of all predictions is 0.181818181818.
for the 175th passenger, the vote of all predictions is 0.818181818182.
```

```
for the 176th passenger, the vote of all predictions is 0.818181818182.
for the 177th passenger, the vote of all predictions is 0.545454545455.
for the 178th passenger, the vote of all predictions is 0.818181818182.
for the 179th passenger, the vote of all predictions is 0.727272727273.
for the 180th passenger, the vote of all predictions is 0.0.
for the 181th passenger, the vote of all predictions is 0.181818181818.
for the 182th passenger, the vote of all predictions is 0.818181818182.
for the 183th passenger, the vote of all predictions is 0.0.
for the 184th passenger, the vote of all predictions is 0.818181818182.
for the 185th passenger, the vote of all predictions is 0.0.
for the 186th passenger, the vote of all predictions is 0.818181818182.
for the 187th passenger, the vote of all predictions is 0.181818181818.
for the 188th passenger, the vote of all predictions is 0.0.
for the 189th passenger, the vote of all predictions is 0.0.
for the 190th passenger, the vote of all predictions is 0.0.
for the 191th passenger, the vote of all predictions is 0.0.
for the 192th passenger, the vote of all predictions is 0.454545454545.
for the 193th passenger, the vote of all predictions is 0.0909090909091.
for the 194th passenger, the vote of all predictions is 0.727272727273.
for the 195th passenger, the vote of all predictions is 0.0.
for the 196th passenger, the vote of all predictions is 0.818181818182.
for the 197th passenger, the vote of all predictions is 0.272727272727.
for the 198th passenger, the vote of all predictions is 0.0.
for the 199th passenger, the vote of all predictions is 0.181818181818.
for the 200th passenger, the vote of all predictions is 0.727272727273.
for the 201th passenger, the vote of all predictions is 0.818181818182.
for the 202th passenger, the vote of all predictions is 0.272727272727.
for the 203th passenger, the vote of all predictions is 0.818181818182.
for the 204th passenger, the vote of all predictions is 0.0.
for the 205th passenger, the vote of all predictions is 0.0.
for the 206th passenger, the vote of all predictions is 0.272727272727.
for the 207th passenger, the vote of all predictions is 0.0.
for the 208th passenger, the vote of all predictions is 0.818181818182.
for the 209th passenger, the vote of all predictions is 0.0.
for the 210th passenger, the vote of all predictions is 0.0.
for the 211th passenger, the vote of all predictions is 0.0.
for the 212th passenger, the vote of all predictions is 0.0.
for the 213th passenger, the vote of all predictions is 0.454545454545.
for the 214th passenger, the vote of all predictions is 0.272727272727.
for the 215th passenger, the vote of all predictions is 0.0.
for the 216th passenger, the vote of all predictions is 0.727272727273.
for the 217th passenger, the vote of all predictions is 0.0909090909091.
for the 218th passenger, the vote of all predictions is 0.818181818182.
for the 219th passenger, the vote of all predictions is 0.0.
for the 220th passenger, the vote of all predictions is 0.818181818182.
for the 221th passenger, the vote of all predictions is 0.0.
for the 222th passenger, the vote of all predictions is 0.818181818182.
for the 223th passenger, the vote of all predictions is 0.363636363636.
for the 224th passenger, the vote of all predictions is 0.545454545455.
for the 225th passenger, the vote of all predictions is 0.272727272727.
for the 226th passenger, the vote of all predictions is 0.0.
```

```
for the 227th passenger, the vote of all predictions is 0.636363636364.
for the 228th passenger, the vote of all predictions is 0.0.
for the 229th passenger, the vote of all predictions is 0.0.
for the 230th passenger, the vote of all predictions is 0.0.
for the 231th passenger, the vote of all predictions is 0.818181818182.
for the 232th passenger, the vote of all predictions is 0.0909090909091.
for the 233th passenger, the vote of all predictions is 0.0.
for the 234th passenger, the vote of all predictions is 0.545454545455.
for the 235th passenger, the vote of all predictions is 0.0.
for the 236th passenger, the vote of all predictions is 0.545454545455.
for the 237th passenger, the vote of all predictions is 0.363636363636.
for the 238th passenger, the vote of all predictions is 0.818181818182.
for the 239th passenger, the vote of all predictions is 0.818181818182.
for the 240th passenger, the vote of all predictions is 0.545454545455.
for the 241th passenger, the vote of all predictions is 0.727272727273.
for the 242th passenger, the vote of all predictions is 0.272727272727.
for the 243th passenger, the vote of all predictions is 0.0.
for the 244th passenger, the vote of all predictions is 0.0.
for the 245th passenger, the vote of all predictions is 0.0.
for the 246th passenger, the vote of all predictions is 0.818181818182.
for the 247th passenger, the vote of all predictions is 0.0909090909091.
for the 248th passenger, the vote of all predictions is 0.818181818182.
for the 249th passenger, the vote of all predictions is 0.181818181818.
for the 250th passenger, the vote of all predictions is 0.818181818182.
for the 251th passenger, the vote of all predictions is 0.0.
for the 252th passenger, the vote of all predictions is 0.181818181818.
for the 253th passenger, the vote of all predictions is 0.0.
for the 254th passenger, the vote of all predictions is 0.0.
for the 255th passenger, the vote of all predictions is 0.0.
for the 256th passenger, the vote of all predictions is 0.0.
for the 257th passenger, the vote of all predictions is 0.0.
for the 258th passenger, the vote of all predictions is 0.818181818182.
for the 259th passenger, the vote of all predictions is 0.0.
for the 260th passenger, the vote of all predictions is 0.0.
for the 261th passenger, the vote of all predictions is 0.0.
for the 262th passenger, the vote of all predictions is 0.818181818182.
for the 263th passenger, the vote of all predictions is 0.818181818182.
for the 264th passenger, the vote of all predictions is 0.181818181818.
for the 265th passenger, the vote of all predictions is 0.0.
for the 266th passenger, the vote of all predictions is 0.0909090909091.
for the 267th passenger, the vote of all predictions is 0.0.
for the 268th passenger, the vote of all predictions is 0.181818181818.
for the 269th passenger, the vote of all predictions is 0.0.
for the 270th passenger, the vote of all predictions is 0.181818181818.
for the 271th passenger, the vote of all predictions is 0.0.
for the 272th passenger, the vote of all predictions is 0.818181818182.
for the 273th passenger, the vote of all predictions is 0.818181818182.
for the 274th passenger, the vote of all predictions is 0.0.
for the 275th passenger, the vote of all predictions is 0.727272727273.
for the 276th passenger, the vote of all predictions is 0.0.
for the 277th passenger, the vote of all predictions is 0.0.
```

```
for the 278th passenger, the vote of all predictions is 0.0.
for the 279th passenger, the vote of all predictions is 0.0.
for the 280th passenger, the vote of all predictions is 0.181818181818.
for the 281th passenger, the vote of all predictions is 0.727272727273.
for the 282th passenger, the vote of all predictions is 0.636363636364.
for the 283th passenger, the vote of all predictions is 0.272727272727.
for the 284th passenger, the vote of all predictions is 0.818181818182.
for the 285th passenger, the vote of all predictions is 0.0.
for the 286th passenger, the vote of all predictions is 0.0.
for the 287th passenger, the vote of all predictions is 0.0909090909091.
for the 288th passenger, the vote of all predictions is 0.0.
for the 289th passenger, the vote of all predictions is 0.0.
for the 290th passenger, the vote of all predictions is 0.0.
for the 291th passenger, the vote of all predictions is 0.272727272727.
for the 292th passenger, the vote of all predictions is 0.0.
for the 293th passenger, the vote of all predictions is 0.0.
for the 294th passenger, the vote of all predictions is 0.0.
for the 295th passenger, the vote of all predictions is 0.0.
for the 296th passenger, the vote of all predictions is 0.727272727273.
for the 297th passenger, the vote of all predictions is 0.0.
for the 298th passenger, the vote of all predictions is 0.0.
for the 299th passenger, the vote of all predictions is 0.0.
for the 300th passenger, the vote of all predictions is 0.272727272727.
for the 301th passenger, the vote of all predictions is 0.0.
for the 302th passenger, the vote of all predictions is 0.0.
for the 303th passenger, the vote of all predictions is 0.0.
for the 304th passenger, the vote of all predictions is 0.636363636364.
for the 305th passenger, the vote of all predictions is 0.727272727273.
for the 306th passenger, the vote of all predictions is 0.363636363636.
for the 307th passenger, the vote of all predictions is 0.818181818182.
for the 308th passenger, the vote of all predictions is 0.0.
for the 309th passenger, the vote of all predictions is 0.181818181818.
for the 310th passenger, the vote of all predictions is 0.0.
for the 311th passenger, the vote of all predictions is 0.363636363636.
for the 312th passenger, the vote of all predictions is 0.0.
for the 313th passenger, the vote of all predictions is 0.363636363636.
for the 314th passenger, the vote of all predictions is 0.818181818182.
for the 315th passenger, the vote of all predictions is 0.727272727273.
for the 316th passenger, the vote of all predictions is 0.363636363636.
for the 317th passenger, the vote of all predictions is 0.0.
for the 318th passenger, the vote of all predictions is 0.0.
for the 319th passenger, the vote of all predictions is 0.0.
for the 320th passenger, the vote of all predictions is 0.0.
for the 321th passenger, the vote of all predictions is 0.0.
for the 322th passenger, the vote of all predictions is 0.0.
for the 323th passenger, the vote of all predictions is 0.454545454545.
for the 324th passenger, the vote of all predictions is 0.818181818182.
for the 325th passenger, the vote of all predictions is 0.0.
for the 326th passenger, the vote of all predictions is 0.818181818182.
for the 327th passenger, the vote of all predictions is 0.0909090909091.
for the 328th passenger, the vote of all predictions is 0.0909090909091.
```

```
for the 329th passenger, the vote of all predictions is 0.0.
for the 330th passenger, the vote of all predictions is 0.727272727273.
for the 331th passenger, the vote of all predictions is 0.0909090909091.
for the 332th passenger, the vote of all predictions is 0.0.
for the 333th passenger, the vote of all predictions is 0.545454545455.
for the 334th passenger, the vote of all predictions is 0.0.
for the 335th passenger, the vote of all predictions is 0.0.
for the 336th passenger, the vote of all predictions is 0.0.
for the 337th passenger, the vote of all predictions is 0.181818181818.
for the 338th passenger, the vote of all predictions is 0.0.
for the 339th passenger, the vote of all predictions is 0.0.
for the 340th passenger, the vote of all predictions is 0.0.
for the 341th passenger, the vote of all predictions is 0.0909090909091.
for the 342th passenger, the vote of all predictions is 0.272727272727.
for the 343th passenger, the vote of all predictions is 0.818181818182.
for the 344th passenger, the vote of all predictions is 0.0.
for the 345th passenger, the vote of all predictions is 0.545454545455.
for the 346th passenger, the vote of all predictions is 0.0.
for the 347th passenger, the vote of all predictions is 0.636363636364.
for the 348th passenger, the vote of all predictions is 0.0.
for the 349th passenger, the vote of all predictions is 0.818181818182.
for the 350th passenger, the vote of all predictions is 0.818181818182.
for the 351th passenger, the vote of all predictions is 0.0.
for the 352th passenger, the vote of all predictions is 0.0.
for the 353th passenger, the vote of all predictions is 0.0.
for the 354th passenger, the vote of all predictions is 0.727272727273.
for the 355th passenger, the vote of all predictions is 0.181818181818.
for the 356th passenger, the vote of all predictions is 0.818181818182.
for the 357th passenger, the vote of all predictions is 0.0.
for the 358th passenger, the vote of all predictions is 0.0.
for the 359th passenger, the vote of all predictions is 0.727272727273.
for the 360th passenger, the vote of all predictions is 0.0.
for the 361th passenger, the vote of all predictions is 0.818181818182.
for the 362th passenger, the vote of all predictions is 0.818181818182.
for the 363th passenger, the vote of all predictions is 0.272727272727.
for the 364th passenger, the vote of all predictions is 0.818181818182.
for the 365th passenger, the vote of all predictions is 0.272727272727.
for the 366th passenger, the vote of all predictions is 0.0.
for the 367th passenger, the vote of all predictions is 0.545454545455.
for the 368th passenger, the vote of all predictions is 0.818181818182.
for the 369th passenger, the vote of all predictions is 0.363636363636.
for the 370th passenger, the vote of all predictions is 0.0.
for the 371th passenger, the vote of all predictions is 0.818181818182.
for the 372th passenger, the vote of all predictions is 0.181818181818.
for the 373th passenger, the vote of all predictions is 0.0.
for the 374th passenger, the vote of all predictions is 0.818181818182.
for the 375th passenger, the vote of all predictions is 0.818181818182.
for the 376th passenger, the vote of all predictions is 0.181818181818.
for the 377th passenger, the vote of all predictions is 0.0.
for the 378th passenger, the vote of all predictions is 0.0.
for the 379th passenger, the vote of all predictions is 0.0909090909091.
```

```
  for the 380th passenger, the vote of all predictions is 0.0.
  for the 381th passenger, the vote of all predictions is 0.0.
  for the 382th passenger, the vote of all predictions is 0.363636363636.
  for the 383th passenger, the vote of all predictions is 0.363636363636.
  for the 384th passenger, the vote of all predictions is 0.272727272727.
  for the 385th passenger, the vote of all predictions is 0.818181818182.
  for the 386th passenger, the vote of all predictions is 0.0.
  for the 387th passenger, the vote of all predictions is 0.0.
  for the 388th passenger, the vote of all predictions is 0.0909090909091.
  for the 389th passenger, the vote of all predictions is 0.0.
  for the 390th passenger, the vote of all predictions is 0.181818181818.
  for the 391th passenger, the vote of all predictions is 0.818181818182.
  for the 392th passenger, the vote of all predictions is 0.727272727273.
  for the 393th passenger, the vote of all predictions is 0.0.
  for the 394th passenger, the vote of all predictions is 0.0.
  for the 395th passenger, the vote of all predictions is 0.818181818182.
  for the 396th passenger, the vote of all predictions is 0.0.
  for the 397th passenger, the vote of all predictions is 0.818181818182.
  for the 398th passenger, the vote of all predictions is 0.0.
  for the 399th passenger, the vote of all predictions is 0.0.
  for the 400th passenger, the vote of all predictions is 0.818181818182.
  for the 401th passenger, the vote of all predictions is 0.0.
  for the 402th passenger, the vote of all predictions is 0.818181818182.
  for the 403th passenger, the vote of all predictions is 0.0909090909091.
  for the 404th passenger, the vote of all predictions is 0.363636363636.
  for the 405th passenger, the vote of all predictions is 0.181818181818.
  for the 406th passenger, the vote of all predictions is 0.0.
  for the 407th passenger, the vote of all predictions is 0.0909090909091.
  for the 408th passenger, the vote of all predictions is 0.727272727273.
  for the 409th passenger, the vote of all predictions is 0.818181818182.
  for the 410th passenger, the vote of all predictions is 0.636363636364.
  for the 411th passenger, the vote of all predictions is 0.818181818182.
  for the 412th passenger, the vote of all predictions is 0.181818181818.
  for the 413th passenger, the vote of all predictions is 0.0.
  for the 414th passenger, the vote of all predictions is 0.818181818182.
  for the 415th passenger, the vote of all predictions is 0.0.
  for the 416th passenger, the vote of all predictions is 0.0.
  for the 417th passenger, the vote of all predictions is 0.363636363636.
Prediction: wrote in the file csv/Vote_best.csv.
```

Résultats

La soumission du résultat à Kaggle donne ? ?. ? ?%.

`Vote.`**`csv_files`** **= ['SVM_best.csv', 'RandomForest_best.csv', 'ExtraTrees_best.csv', 'GradientBoosting_best**
Fichiers utilisés pour le vote.

`Vote.`**`vote`**(*Outputs*, *z=0*) $\rightarrow$ {0|1}
Make the vote, by regarding the most probable output, by considering all the predictions, stored in Outputs :

–Outputs must be a dictionnary {filename :Output}.
–And Output is a prediction int-array (like [1,1,1,1,0,0,1,0,1,1,0,...]).
–z is the index of the passenger.

4.6 Auteurs (fichier AUTHOR)

List of contributors

Main author : Lilian BESSON (mailto :lilian.besson@normale.fr).

Thanks to

– Bitbucket for git hosting ;
– Michèle Sebag and Benjamin Monmege for their support.

4.7 Extrait de la license

Index des modules Python