

# TP INTRO IA N°2

## Les survivants du Titanic

A rendre à santucci\_j@univ-corse.fr

### 1. Les données

Les données d'apprentissage et de test sont disponibles sur kaggle qui un site de compétition du Machine Learning (chercher dans Google Kaggle Titanic et télécharger les données). Vous devez télécharger le fichiers Train (891 lignes et 12 colonnes) et le fichier Test (418 lignes et 11 colonnes)

#### 1.1 Importation des données

**Question 1.1 :** Importer les données et visualiser les premières observations.

Pour cela , vous pouvez utiliser la librairie pandas :

```
import pandas as pd
```

```
train = pd.read_csv('data/train.csv', sep = ',')
```

#### 1.2. Visualisation

**Question 1.2 :** Visualiser les 10 premières lignes du fichier (à analyser ) avec la commande train.-

	PassengerId	Survived	Pass	Name	Sex	Age	SibSp	Parth	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cummings, Mrs. John Bradley (Florence Briggs Th...	female	35	1	0	PC 17596	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101262	7.9250	NaN	C
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.3600	NaN	C
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4500	NaN	C
6	7	0	1	McCarty, Mr. Timothy J	male	NaN	0	0	17463	51.8625	NaN	S
7	8	0	3	Pelton, Master. Gusto Leonard	male	2	3	1	349909	21.0750	NaN	C
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742	11.1333	NaN	C
9	10	1	2	Nasser, Mrs. Nicholas (Aminé Nasser)	female	14	1	0	257736	30.0710	NaN	C

head(10). Vous allez obtenir le résultat suivant.

#### 1.3. Indexation des données

**Question 1.3 :** Indexer les données (créer une colonne permettant d'indexer le fichier précédent).

Pour cela on va utiliser l'instruction de pandas :

DataFrame.set\_index(keys, drop=True, inplace=True) où :

- keys est la colonne du fichier utilisée comme index,
- drop=True signifie que cette colonne choisie sera supprimée dans le nouveau fichier
- Inplace = True signifie que l'on va modifier le fichier passé en paramètre et pas créer un nouvel objet.

Nous allons indexer le fichier Train avec la colonne PassengerId donc : `train.set_index('PassengerId', inplace=True, drop=True)`

#### 1.4. Analyse des variables

Nous voulons visualiser les variables du problème en affichant le noms des colonnes du fichier l'aide `train.columns`.

**Question 1.4.1 :** Ecrire la commande qui permet de visualiser les noms.

Vous devriez obtenir le résultat suivant :

```
Index([u'PassengerId', u'Survived', u'Pclass', u'Name', u'Sex', u'Age', u'SibSp', u'Parch', u'Ticket', u'Fare', u'Cabin', u'Embarked'], dtype='object')
```

**Question 1.4.2 :** Faites une analyse des variables suivantes : Survived, PClass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Embarked.

L'analyse consistera à faire le bilan des variables à l'aide des informations disponibles sur le web. Vous allez donc pour chaque colonne (à part PassengerId qui sert seulement pour indexer le fichier) analyser :

- . ce que signifie la variable
- . le type de la variable
- . si certaines valeurs de la variable sont manquantes pour certains passagers.

**Question 1.4.3 :** Utiliser respectivement les commandes `dtype` et `count` qui vont vous donner respectivement le type des données et le nombre de valeurs non nulles par variables.

Vous allez obtenir le résultat suivant

Survived – int64	Survived – 891
Pclass – int64	Pclass – 891
Name – object	Name – 891
Sex – object	Sex – 891
Age – float64	Age – 714
SibSp – int64	SibSp – 891
Parch – int64	Parch – 891
Ticket – object	Ticket – 891
Fare – float64	Fare – 891
Cabin – object	Cabin – 204
Embarked – object	Embarked – 889
dtype: object	

Cette analyse permet d'avoir une vue d'ensemble du problème et fournit une aide pour définir les modèles d'apprentissage.

## 2. Premier Modèle

Généralement le premier modèle vise à établir un modèle rapide à concevoir qui sert de base de comparaison aux autres modèles plus complexes qu'il vous faudra concevoir par la suite.

Les trois variables SibSp, Parch et Fare sont faciles à utiliser car elles sont de type intfloat et aucune valeur n'est manquante (remarque : la variable PClass est de type intfloat mais n'est pas continue). Ces trois variables sont complètes et donc pas de réflexion sur d'éventuelles valeurs manquantes à compléter.

Pour le premier modèle vous pouvez utiliser ces variables.

### 2.1. Préparation des données

Avant de définir le modèle, vous allez préparer les données.

A partir du fichier train vous devez récupérer :

- d'une part le vecteur de sortie qu'on appellera par exemple `target` (vecteur à prédire) donc la variable `Survived` et
- d'autre part de filter le fichier pour ne considérer que les trois variables précédentes au sein d'une matrice `X`.

**Question 2.1 :** Pour cela vous écrirez une fonction `parse_model_1` qui prendra en paramètre un fichier de type train et retournera le vecteur de sortie `y` (Colonne `Survived` du fichier passé en paramètre et la matrice `X` qui est un sous-ensemble du fichier passé en paramètre qui a été filtré avec les trois colonnes correspondants aux trois variables.

Vous devez donc obtenir une matrice `X` du type suivant :

	Fare	SibSp	Parch
PassengerId			
1	7.2500	1	0
2	71.2833	1	0
3	7.9250	0	0
4	53.1000	1	0
5	8.0500	0	0

### 2.2. Stratégie de Validation

On va choisir la validation croisée.

**Question 2.2.1 :** Donner selon vous les raisons de choisir une validation croisée.

Indication : Dans notre cas, le jeu de données est très petit et utiliserons simple validation basée sur un split unique risque de ne pas donner de bon résultats. Une validation croisée doit permettre l'obtention de meilleurs résultats.

Pour réussir une validation significative, on se propose d'utiliser la moyenne de plusieurs validations croisées (par exemple 5).

**Question 2.2.2 :** Ecrire une fonction `compute_score` qui prend comme paramètre un classifieur, la matrice `X` et la cible `y` et qui retourne la moyenne de 5 validations croisées (on entraîne sur 4/5 du jeu de données et on teste sur le 1/5 restant. (Indication : utiliser le module `cross_val_score` qui va réaliser 5 validations croisées du classifieur passé en paramètre)

**Question 2.2.3 :** Calculer le modèle de regression logistique (qui constituera votre 1er modèle de prédiction) et utiliser la fonction `compute_score` de la question 2.2.2 pour obtenir le score qui indique la qualité de la prédiction. Ce score sera le score à battre avec les autres modèles que vous allez écrire.

Remarque : vous pourriez être tenté d'essayer d'autres algorithmes plus complexes mais le gain sera pas très important. Il vaut mieux d'abord s'intéresser aux autres variables.

### 3. Etude des variables.

Il existe de nombreuses méthodes qui permettent de qualifier l'importance des variables d'entrée basées sur la réalisation de tests statistiques. Ces méthodes fonctionnent très bien mais l'interprétation des résultats repose sur l'expérience pour interpréter les résultats de ce type de méthodes.

Nous allons explorer une méthode simple pour qualifier l'importance des variables d'un problème donné en Machine Learning.

#### 3.1. Préparation de la méthode

Il va falloir diviser la population en deux sous-populations en fonction de la variable à prédire et tracer l'histogramme de la distribution des sous-populations pour cette variable.

**Question 3.1.1 :** Vous allez tout d'abord séparer les données en deux populations, les survivants et les victimes. avec les instructions suivantes :

```
survived = train[train.Survived == 1]
```

```
dead = train[train.Survived == 0]
```

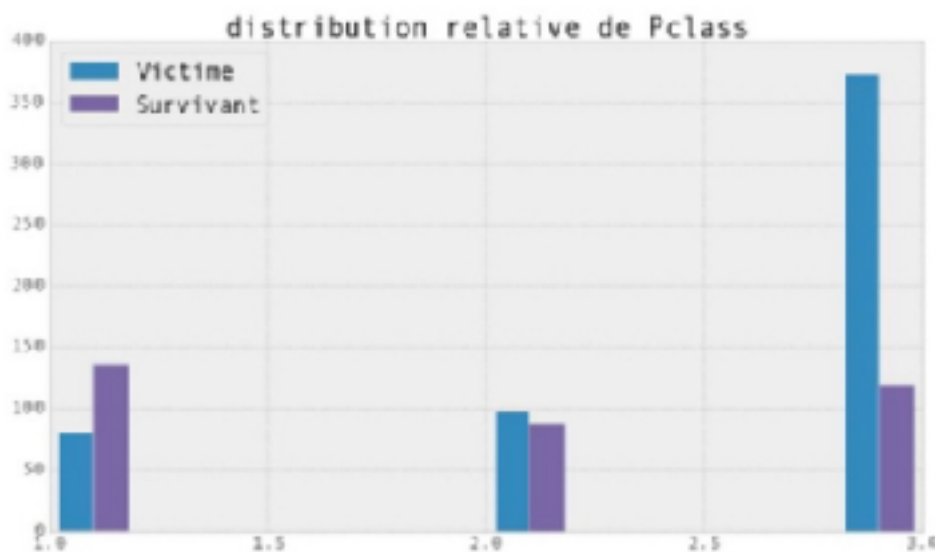
Il faut maintenant observer la distribution comparée des variables de ces deux populations, par exemple, sur la variable Pclass (ou d'autres).

**Question 3.1.2 :** Ecrire la fonction plot\_hist qui prend en arguments le nom d'une des variables des données (Pclass ou autres) et qui trace l'histogramme de la distribution des survivants et des victimes pour la variable passée en paramètre.

#### 3.2. Etude de la variable Pclass.

Nous allons tout d'abord étudier la variable Pclass

**Question 3.2.1 :** Afficher l'histogramme de la distribution des survivants et des victimes pour la variable Pclass en utilisant la fonction plot\_hist. Vous devez obtenir un résultat du type ci-dessous:



**Question 3.2.2 :** La variable Pclass est une variable importante. Pourquoi ?

Pour répondre à cette question, vous devez observer la distribution relative de la classe Pclass donc étudier si on peut discriminer les survivants et les victimes pour les trois classe : 1, 2 et 3.

### 3.3 Traitement de la variable Pclass

Puisque la variable Pclass est une variable importante, il faudra l'intégrer dans le prochain modèle.

Or cette variable est catégorielle (qu'on pourrait aussi considérer comme ordonnée aussi) : il faut effectuer certains traitements avant de l'intégrer dans un modèle. Nous allons la transformer en variables de type booléens. En effet comme la variable possède peu de modalités (les classes 1,2, 3) on peut créer des nouvelles variables (Pclass\_1, Pclass\_2 et Pclass3).

La création de ces variables (qu'on appelle dummies) se fera automatiquement en utilisant la librairie pandas : `pd.get_dummies(X['Pclass'], prefix='split_Pclass')`

Cela permet d'obtenir trois vecteurs booléen pour chacune des modalités de Pclass.

**Question 3.3.1 :** Ecrire la fonction `parse_model_2` en injectant les instructions précédentes dans la fonction `parse_model_1` de la question 2.1.

**Question 3.3.2 :** Afficher la nouvelle matrice X que l'on obtient

Indication : vous devez obtenir un résultat de ce type :

	SibSp	Parch	Fare	split_Pclass 1	split_Pclass 2	split_Pclass 3
Passengerid						
1	1	0	7.2500	0	0	1
2	1	0	71.2833	1	0	0
3	0	0	7.9250	0	0	1
4	1	0	53.1000	1	0	0
5	0	0	8.0500	0	0	1

### 3.4 Validation du 2ème modèle.

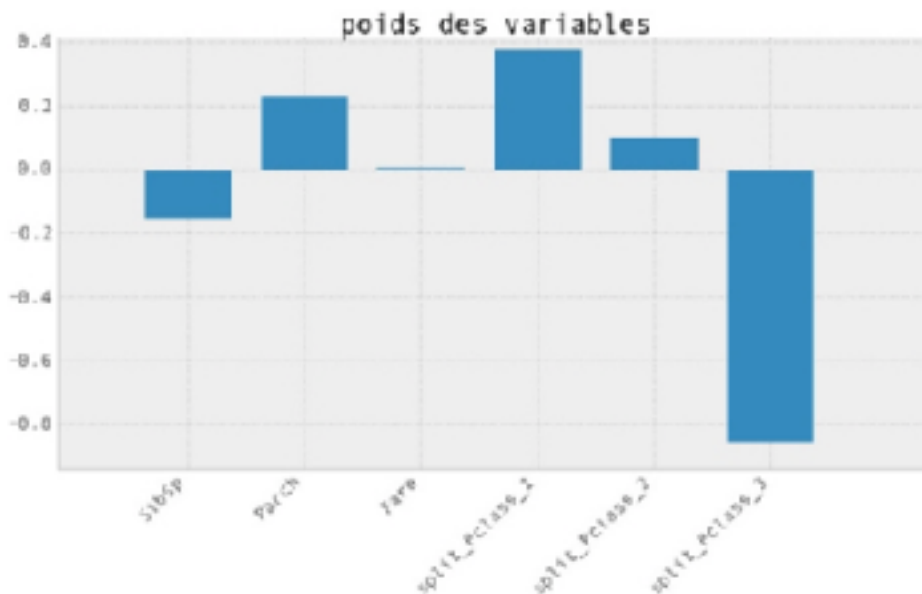
**Question 3.4.1 :** Calculer le modèle de régression logistique avec ce deuxième modèle et calculer le score de validation croisée avec ce nouveau modèle.

**Question 3.4.2 :** Que remarquez vous? (indication : a-t-on amélioré le 1er modèle?).

Nous pouvons aussi nous intéresser aux points des différentes variables dans le modèle de régression utilisé.

Pour cela nous pouvons afficher les coefficients associés à chaque variable de la régression logistique.

**Question 3.4.3 :** Tracer les coefficients de la régression logistique en écrivant une fonction qui affiche les coefficients : `plot_lr_coefs(X, lr)` où `X` est la matrice des données d'entraînement et `lr` un classifieur de régression logistique. vous devez obtenir un graphique de ce type.



Interprétation de l'analyse des poids des variables :

- un poids positif ou un poids négatif important met en évidence l'importance de la variable dans le résultat du modèle.
- un poids proche de zéro signifie que l'influence de la variable est négligeable.

**Question 3.4.4 :** Donner une interprétation concernant les variables du modèle 2 à partir du graphique de la question 3.4.3

#### 4. Etude plus fine des variables.

Des indications mettent en avant que les stratégies d'embarquement des canots de sauvetage ont été différentes à bâbord et à tribord. A tribord « les femmes et les enfants d'abord » a été appliqué en priorité mais pas en exclusivité donc des hommes pouvaient monter sur les canots. Côté bâbord les hommes avaient l'interdiction systématique de monter sur les canots (stratégie pas très maligne puisque il n'y avait que 500 femmes et enfants pour 1200 places dans les canots).

Donc il semble intéressant de s'intéresser aux variables Age et Sex. Il faut cependant traiter ces variables car d'une part Sex est de type string et Age n'est remplie qu'à 80%.



Pour la Class Age, la transformation sera effectuée comme pour la variable PClass : elle sera binarisée en créant deux variables dummy de type booléen.

Pour la variable Age, nous emploierons la stratégie qui va consister à remplir les valeurs manquantes avec une valeur correspond à la médiane de la distribution.

#### **4.1. Création d'un nouveau jeu de données avec les variable Sex et Age**

**Question 4.1 :** Ecrire la fonction parse\_model\_3 en injectant les instructions permettant s'insérer les variables Age et Sex dans la fonction parse\_model\_2 de la question 3.3.1.

Indication : Pour Sex on utilisera les mêmes instructions que pour la variable Pclass (variables dummy) et pour la variable Age, on complétera les valeurs manquantes en utilisant la méthode fillna de pandas.

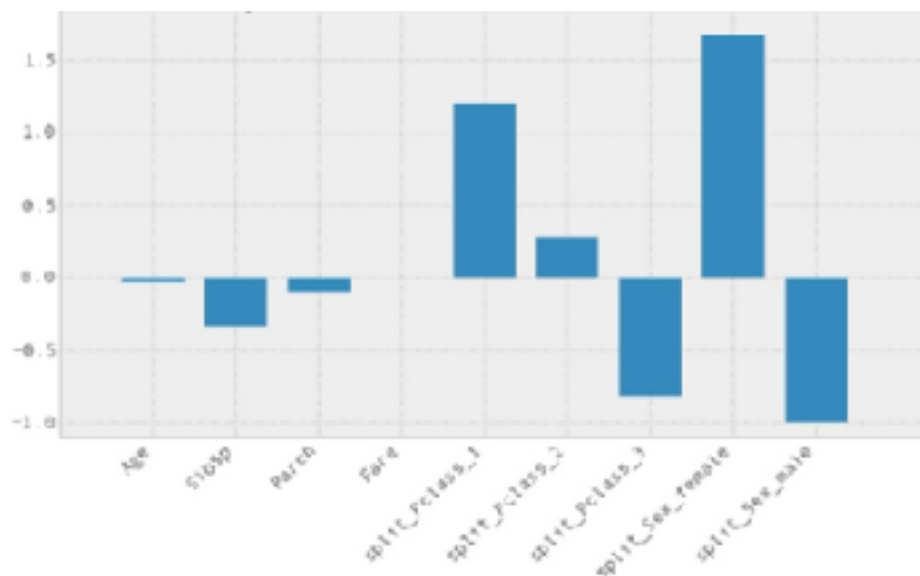
#### **4.2 Calcul du nouveau modèle et évaluation du nouveau score**

**Question 4.2.1 :** Utiliser la régression logistique comme précédemment pour calculer le nouveau modèle et calculer le nouveau score.

Vous devez remarquer une nette amélioration du score. En utilisant la fonction plot\_lr\_coefs(X, lr) de la question 3.4.3 , nous pouvons analyser le poids du modèle 3.

**Question 4.2.2 :** Afficher le graphique permettant d'analyser le poids des variables du modèle 2.

Indication : vous devriez obtenir le graphique suivant :



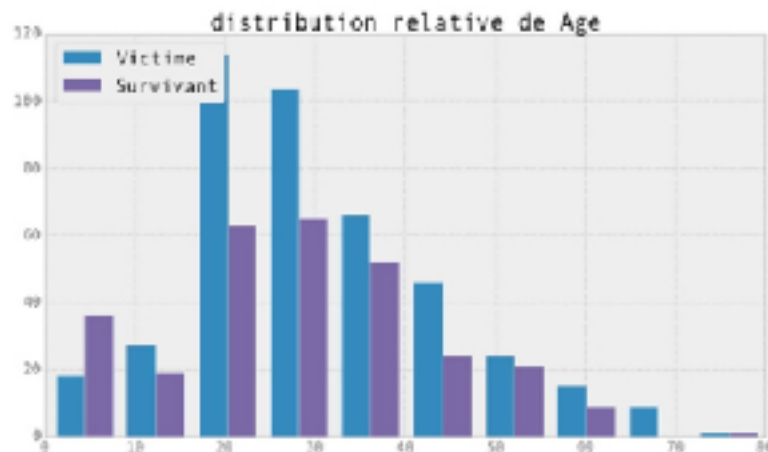
**Question 4.2.3 :** quelles conclusions faites vous pour ce qui concerne l'amélioration du modèle (est ce due à la variable Age ou aux variable de type Sex?)

Nous allons donc étudier un peu plus en détail la variable Age.

### 4.3. Etude de la variable Age et nouveau modèle

Afin d'étudier l'importance de la variable Age, nous allons utiliser la fonction `plot_hist` de la question 3.1.2

**Question 4.3.1 :** Tracer les histogrammes des survivants et des victimes pour la variable Age. Vous devez obtenir une distribution du type ci-dessous :

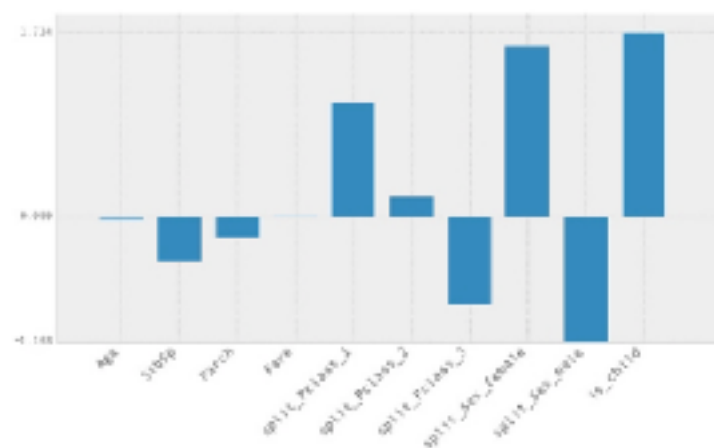


**Question 4.3.2 :** que remarquer vous pour les enfants et les 20 et 30 ans? Après l'analyse de la variable Age, nous voyons l'intérêt de créer des nouvelles variables comme par exemple une variable permettant de mettre en évidence le fait d'être un enfant. On va ajouter la variable `is_child` grâce à l'instruction : `X['is_child'] = X.Age < 8`

**Question 4.3.3 :** Ajouter cette instruction pour obtenir un nouveau modèle permettant de prendre en compte la variable `is_child`.

**Question 4.3.4 :** Comme précédemment (question 2.2.3) calculer le nouveau modèle et le score associé.

**Question 4.3.5 :** Vérifier que le score obtenu correspondant à une amélioration est du à l'introduction de la variable `is_child` (pour cela utiliser la fonction `plot_lr_coefs(X, lr)` de la question 3.4.3). Indication : vous devriez obtenir le graphique ci-dessous



## 5. Modèles non linéaires

Nous avons réussi à prendre en compte l'effet de la non linéarité de variables (exemple le fait dire un enfant) par un modèle linéaire comme la régression logistique et intervenant sur les données (en créant la variable `is_child`).

Nous allons voir maintenant comment utiliser des modèles non linéaires qui sont faits en particulier pour prendre en compte ce genre d'information cachée. Mais dans tous les cas créer les bonnes variables sera la meilleure façon d'obtenir les meilleures performances des modèles que vous créerez.

### 5.1. Modèles de type *Random Forest*

Les modèles non linéaires sont simples à utiliser dans scikit learn (comme vu en cours).

Nous allons prendre comme exemple le module Random Forest sans créer dans un premier temps la variable `is_child` (juste pour observer les résultats obtenus).

**Question 5.1.1 :** Appliquer le classifieur RandomForest de scikit learn sur le modèle sans la variable `child` et calculer le nouveau score obtenu.

**Question 5.1.2 :** Que remarquez vous?

### 5.2. Interprétation du résultats avec *RandomForest*

Puisque le classifieur Random Forest est de type non linéaire, nous ne pouvons plus utiliser le poids des variables comme dans les questions précédentes pour analyser le modèle.

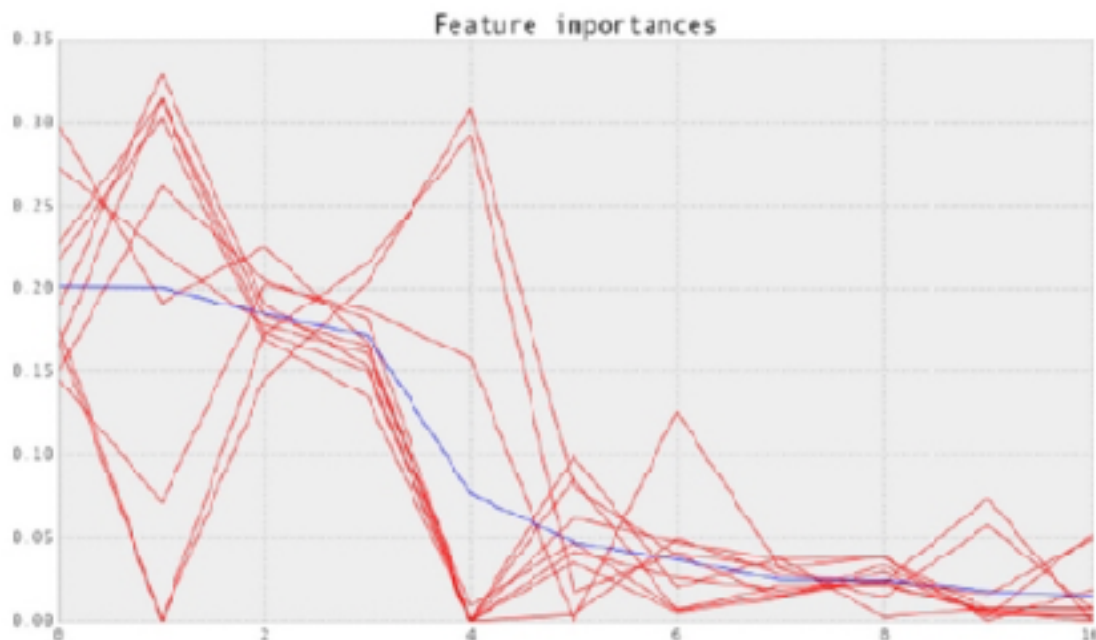
Vous allez pouvoir utiliser une nouvelle fonction permettant d'interpréter les résultats obtenus avec un classifieur de type Random Forest. Cette fonction `clf_importance` permet :

1. de tracer un schéma mettant en évidence l'importance relative des variables pour chaque arbre de la forêt. Chaque tracé correspond à un arbre (par défaut le nombre d'arbres est de 10).  
L'importance relative de chaque variable est indiquée en ordonnée et vous donne des indications sur la concentration plus ou moins forte de l'algorithme Random Forest.
2. de classer chaque variable en donnant sa contribution à l'algorithme.

Voilà la fonction qui permet de tracer l'importance des variables :

```
def classifieur_importance(X, clf):
    import pylab as pl
    importances = clf.feature_importances_
    indices = np.argsort(importances)[::-1]
    pl.title("Feature importances »")
    for tree in clf.estimators_:
        pl.plot(xrange(X.shape[1]), tree.feature_importances_[indices], « r")
    pl.plot(xrange(X.shape[1]), importances[indices], « b")
    pl.show()
    for f in range(X.shape[1]):
        print("%d. feature : %s (%f)" % (f+1, X.columns[indices[f]],
            importances[indices[f]]))
```

**Question 5.2.1 :** Utiliser la fonction `classfier_importance(X, clf)` après avoir créer un classifieur de type Random Forest puis l'entraîner.  
 Vous devriez obtenir le schéma suivant qui met en évidence l'importance des variables dans l'algorithme :



L'importance des variables représentée en abscisse est fournie en ordonnée. Les variables sont classées par ordre d'importance.

Le deuxième résultat de la fonction `classfier_importance(X, clf)` est de fournir la contribution de chaque variable :

```

1. feature : Fare (0.200859)
2. feature : split_Sex_female (0.200355)
3. feature : random (0.184521)
4. feature : Age (0.171788)
5. feature : split_Sex_male (0.076643)
6. feature : split_Pclass_3 (0.047177)
7. feature : split_Pclass_1 (0.036631)
8. feature : Parch (0.024891)
9. feature : SibSp (0.024589)
10. feature : split_Pclass_2 (0.017957)
11. feature : is_child (0.014590)
  
```

**Question 5.2.2 :** Que remarquez vous concernant l'importance des variables.

### 5.3 Utilisation des autres variables

Pour obtenir de bons résultats en Machine Learning, il faut prendre en compte les plus de variables caractéristiques possibles. Or on néglige souvent les variables que l'on a du mal à traiter comme les variables de type string car elles sont déstructurées ou qu'elles présentent des valeurs aberrantes ou manquantes.

Nous allons voir comment prendre en compte de telles variables.  
Prenons comme exemple la variables name.

```
PassengerId      Braund, Mr. Owen Harris
1
2      Cumings, Mrs. John Bradley (Florence Briggs Th...
3      Heikkinen, Miss. Laina
4      Futrelle, Mrs. Jacques Heath (Lily May Peel)
5      Allen, Mr. William Henry
6      Moran, Mr. James
7      McCarthy, Mr. Timothy J
8      Palsson, Master. Gosta Leonard
9      Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
10     Nasser, Mrs. Nicholas (Adele Achen)
11     Sandstrom, Miss. Marguerite Rut
12     Bonnell, Miss. Elizabeth
13     Saunderson, Mr. William Henry
14     Andersson, Mr. Anders Johan
15     Vestrom, Miss. Hulda Amanda Adolfina
```

Nous pouvons remarquer que la valeurs de la variable Name ont souvent la structure suivante :  
nom1, titre, nom2, nom3 (surnom) - surnom est optionnel.

On peut penser que l'information concernant le titre peut jouer un rôle important pour la classification.

**Question 5.3.1** : Créer une nouvelle variable Title avec l'instruction suivante :

```
X['title'] = X.Name.map(lambda x : x.split(',')[1].split('.')[0])
```

Si vous examinez la variable Cabin, vous remarquerez que cette variable est très peu remplie mais cette variable est importante car elle comporte l'information de pont et donc de proximité avec les canots. Quand l'information n'est pas manquante, elle est de la forme : 'pont"numero de cabine'

**Question 5.3.2** : Créer une nouvelle variable cabin avec l'instruction suivante :

```
X['cabin'] = X.Cabin.map(lambda x : x[0] if not pd.isnull(x) else -1)
```

Question 5.3.3 : Ecrire la fonction parse\_model\_5 (X) qui permet de créer un nouveau jeu de données en utilisant les nouvelles variables créées à partir de Name et Cabin et venant compléter le dernier jeu de données de la question 4.3.3

indication : attention vous devez utiliser des variables dummy pour les variables Title et cabin

**Question 5.3.4** : Calculer le modèle de régression logistique correspond et calcul le score obtenu

**Question 5.3.5** : Que remarquez vous?

## **6. Aller plus loin**

### ***6.1 Améliorer le modèle de régression logistique***

**Question 6.1.1** : Vous pouvez aussi étudier les autres variables : surname, embarked, ou ticket éventuellement.

Question 6.1.2 : Améliorer le remplissage des valeurs manquantes

### **6.2 Nouveaux modèles**

**Question 6.2.1** : Utiliser des modèles plus complexes (vous pouvez regarder la documentation concernant le boosting ou le SVM ou autre

**Question 6.2.2** : Vous pouvez aussi choisir de meilleurs paramètres pour chacun des algorithmes.

.