Les survivants du Titanic (3)

Apprentissage automatique avancé
Master 2 WeDSci
2024 / 2025

Ce TP est une adaptation du chapitre 17 "Prédire les survivants du Titanic" du livre de Eric Biernat et Michel Lutz "Data Science : fondamentaux et études de cas. Machine Learning avec Python et R", Eyrolles, 2015.

1 Les données

Les données d'apprentissage et de test sont disponibles sur kaggle qui un site de compétition des data scientists.

1. Importer les données et visualiser les premières observations :

```
import pandas as pd
train = pd.read_csv('data01/titanic_train.csv', sep = ',')

train.head(10)
```

2. Indexer les données :

```
train.set_index('PassengerId', inplace=True, drop=True)
```

3. Observer les noms des variables à l'aide `train.columns` et faire le bilan de celles-ci à l'aide des informations disponibles sur le web.

2 Premier modèle de niveau 0

Ce premier modèle vise à établir un modèle rapide à concevoir qui sert de base de comparaison aux autres modèles plus sophistiqués qui suivront.

Les trois variables `SibSp`, `Parch` et `Fare` semblent facile à utiliser car elles sont de type intfloat et aucune valeur n'est manquante.

1. Filtrer et construire les données du modèle 0 :

```
def parse_model_0(X):
    target = X.Survived
    X = X[['Fare', 'SibSp', 'Parch']]
```

```
    return X, target

X, y = parse_model_0(train.copy())
```

2. Pourquoi choisit-on comme stratégie de validation la validation croisée (cross-validation) pour ce jeu de données ?

```
import numpy as np
from sklearn.cross_validation import cross_val_score

def compute_score(clf, X, y):
    xval = cross_val_score(clf, X, y, cv = 5)
    return np.mean(xval)
```

3. Calculer le modèle de régression logistique.

3 Deuxième modèle

Pour construire ce deuxième modèle, on observe l'importance des variables. Il existe de nombreuses méthodes pour mesurer l'importance des variables. L'une des plus simples est de tracer la valeur finale à prédire en fonction de chaque variable.

1. Diviser la population en deux sous-populations en fonction de la variable à prédire :

```
survived = train[train.Survived == 1]
dead = train[train.Survived == 0]
```

2. Tracer l'histogramme de la distribution des survivants et des victimes pour la variable Pclass :

```
def plot_hist(feature, bins = 20):
    x1 = array(dead[feature].dropna())
    x2 = array(survived[feature].dropna())
    plt.hist([x1, x2], label=['Victime', 'Survivant'], bins = bins)#, color = ['', 'b']
    plt.legend(loc = 'upper left')
    plt.title('distribution relative de %s' % feature)
    plt.show()

plot_hist('Pclass')
```

3. La variable Pclass est-elle importante ? Pourquoi ?
4. La variable catégorielle Pclass a peu de modalité, on peut donc créer une variable 'dummy' pour chaque modalité. Transformer la variable catégorielle Pclass en variable dummy et créer le nouveau jeu de donnée pour le modèle 1 :

```
def parse_model_1(X):
    target = X.Survived
    class_dummies = pd.get_dummies(X['Pclass'], prefix='split_Pclass')
    X = X.join(class_dummies)
    to_del = ['Name', 'Age', 'Cabin', 'Embarked', 'Survived', 'Ticket', 'Sex', 'Pclass']
    for col in to_del :
        del X[col]
    return X, target
```

5. Calculer le modèle de régression logistique. Commenter.
6. Tracer les coefficients de la régression logistique et donner une interprétation ceux-ci :

```
def plot_lr_coefs(X, lr):
    fig, ax = plt.subplots()

    xlabels = X.columns.values.tolist()
    yvalues = lr.coef_[0,]
    index = np.arange(len(yvalues))

    bar_width = 0.35
    opacity = 0.4
    rects = plt.bar(index, yvalues,
                    bar_width, alpha=opacity,
                    color='b', label ='')
    plt.ylabel('Valeur')
    plt.title('Poids des variables')
    plt.xticks(index, xlabels, rotation=40)

    plt.legend()
    plt.tight_layout()
    plt.show()

plot_lr_coefs(X, lr)
```

4 Troisième modèle

Dans la tragédie du Titanic, les femmes et les enfants ont été considéré différemment des hommes (enfin selon l'officier).

La variable Sex peut être traitée de la même façon que la variable Pclass en créant 2 variables dummy.

Par contre, certaine valeur de la variable Age sont manquantes. Il existe de nombreuses façon de traiter les valeurs manquantes. Ici, nous choisissons une façon simple qui remplace la valeur par la valeur médiane de la distribution.

1. Créer le nouveau jeu de donnée en utilisant la méthode `fillna` pour les valeurs manquantes.
2. Calculer la régression logistique.
3. Les nouvelles variables créées sont-elles importantes dans la régression obtenue ?
4. Tracer les histogrammes des survivants et des victimes pour la variable Age. Commenter.
5. Créer la variable non-linéaire `is_child` :

   ```
   X['is_child'] = X.Age < 8
   ```

6. Calculer le modèle de régression logistique. Commenter.

5 Selection de variable

1. En considérant les variables vu jusqu'à présent, quel est le meilleur modèle logistique ayant 2, 3 et 4 variables ?

2. Transformer les variables existantes en variable binaire (si ce n'est déjà le cas).
3. Créer les variables produits $x_i x_j$ des variables binaires.
4. Calculer et comparer la régression logistique sur les nouvelles variables ainsi créées.
5. Transformer les variables existantes en variable binaire dont les valeurs σ_i sont -1 ou $+1$.
6. Créer les variables produits $\sigma_i \sigma_j$ des variables binaires.
7. Calculer et comparer la régression logistique sur les nouvelles variables ainsi créées.

6 Random forest

1. Calculer le modèle de régression random forest sans utiliser la variable `is_child`.
2. Comment est mesuré l'importance d'une variable dans le modèle random forest ?
3. Tracer l'importance des variables :

```
def classifier_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]]))
```

à utiliser par ces lignes :

```
rf = RandomForestClassifier()
rf.fit(X, y)
classifier_importance(X, rf)
```

4. Calculer l'importance des variables par la méthode "Permutation".
5. Interpréter l'importance des variables.

7 Gradient boosting

1. Comparer les performance de la méthode gradient boosting avec random forest.