Machine Learning V
Classification supervisée

Nicolas Bourgeois

SVM : objectif

On dispose d'un ensemble d'observations $(\tilde{X}, \tilde{Y})$.

SVM : objectif

On dispose d'un ensemble d'observations $(\tilde{X}, \tilde{Y})$.

On veut produire une partition de l'espace des X potentiels selon Y, par exemple avec des hyperplans.

SVM : objectif

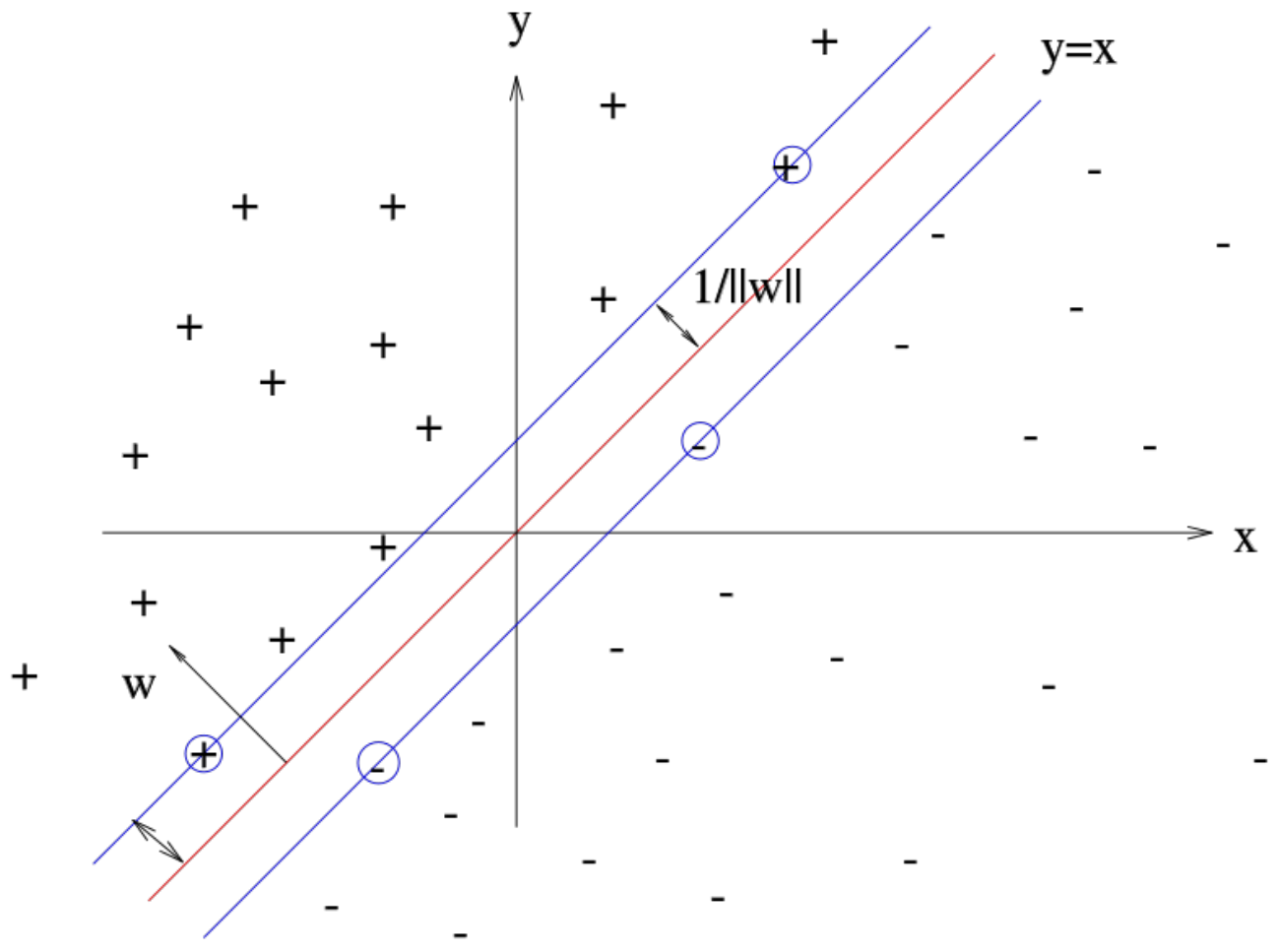
On dispose d'un ensemble d'observations $(\tilde{X}, \tilde{Y})$.

On veut produire une partition de l'espace des X potentiels selon Y, par exemple avec des hyperplans.

Pour optimiser la robustesse on cherche à séparer au maximum les données d'entraînement.

$$\max_{\mathcal{H}} \sum_{x \in X} \min_{y \in \mathcal{H}} ||x - y||^2$$

Exemple



Marge maximale

Exercice

Exercice

Importez les données digits avec `dataset.load_digits`*. Divisez votre échantillon en deux parties. Effectuez un svm pour identifier les chiffre et testez sur l'échantillon de test.*

Exercice

Comparez avec le résultat obtenu si on limite fortement le nombre d'itérations (10)

Résultat attendu

1.0 0.3212045169385194
1 3 0 5 3 6 3 6 1 3 3 4 3 7 3 3 2 3 5 3 3 3 3 3 3 3 9 8 0
1 4 0 5 3 6 9 6 1 7 5 4 4 7 2 8 2 2 5 7 9 5 4 4 9 0 8 9 8 0

0.998 0.8983688833124216
1 4 0 5 3 6 5 6 1 7 5 4 4 7 2 8 2 2 5 7 9 5 5 4 9 0 8 9 8 0
1 4 0 5 3 6 9 6 1 7 5 4 4 7 2 8 2 2 5 7 9 5 4 4 9 0 8 9 8 0

Solution

```
import numpy as np
import pandas as pd
from sklearn.svm import SVC
from sklearn.datasets import load_digits
from random import shuffle
chiffres = load_digits()
X,Y = chiffres.data,chiffres.target
X_train,X_test,Y_train,Y_test=X[:1000],X[1000:],Y[:1000],Y[1000:]
s = SVC()
s.fit(X_train,Y_train)
print(s.score(X_train,Y_train),s.score(X_test,Y_test))
print(s.predict(X_test)[:30])
print(Y_test[:30])
s = SVC(max_iter=10)
s.fit(X_train,Y_train)
print(s.score(X_train,Y_train),s.score(X_test,Y_test))
print(s.predict(X_test)[:30])
print(Y_test[:30])
```

Exercice

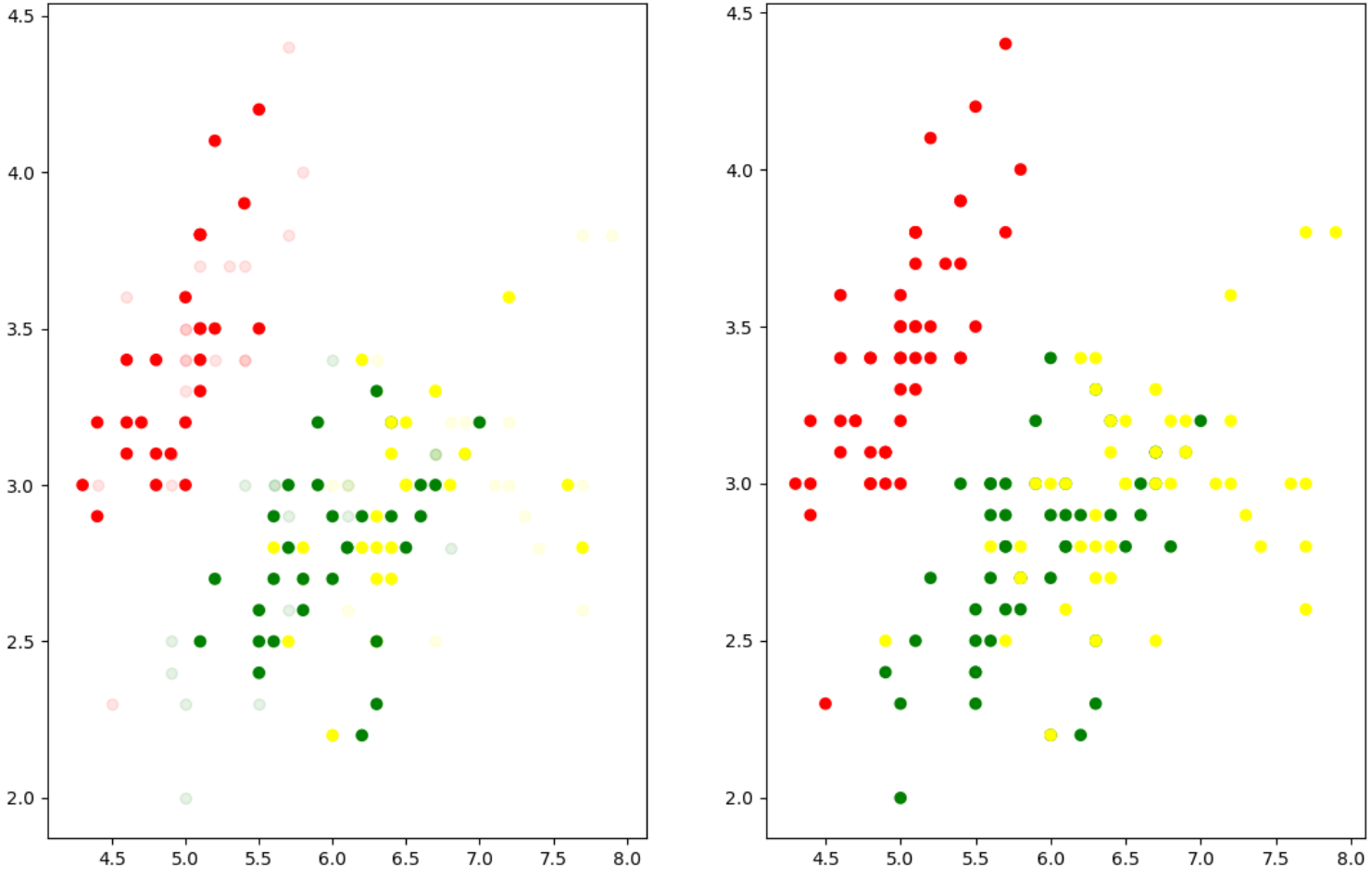
Exercice

Importez les données iris avec `dataset.load_iris`*. Divisez votre échantillon en deux parties. Effectuez un svm linéaire pour séparer les trois types de fleurs et testez le.*

Exercice

Représentez côte à côte les valeurs observées $(\tilde{X}, \tilde{Y})$ *et le résultat de la prédiction (dans ce dernier on utilisera l'opacité pour distinguer les échantillons d'apprentissage et de test)*

Résultat attendu



Solution

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.datasets import load_iris
from random import shuffle
def color(x):
    return({0:'red',1:'green',2:'yellow'}[x])
iris = load_iris()
X,Y = iris.data,pd.Series(iris.target)
i_train = list(range(len(X)))
shuffle(i_train)
X_train,X_test,Y_train=X[i_train[:80]],X[i_train[80:]],Y[i_train[:80]]
s = SVC(kernel='linear')
s.fit(X_train,Y_train)
plt.subplot(121)
for p in X_test:
    plt.scatter(p[0],p[1],alpha=0.1,
        c=color(s.predict([[p[0],p[1],p[2],p[3]]])[0]))
plt.scatter(X_train[:, 0],X_train[:, 1],c=Y_train.apply(color))
plt.subplot(122)
plt.scatter(X[:, 0], X[:, 1],c=Y.apply(color))
plt.show()
```

Exercice

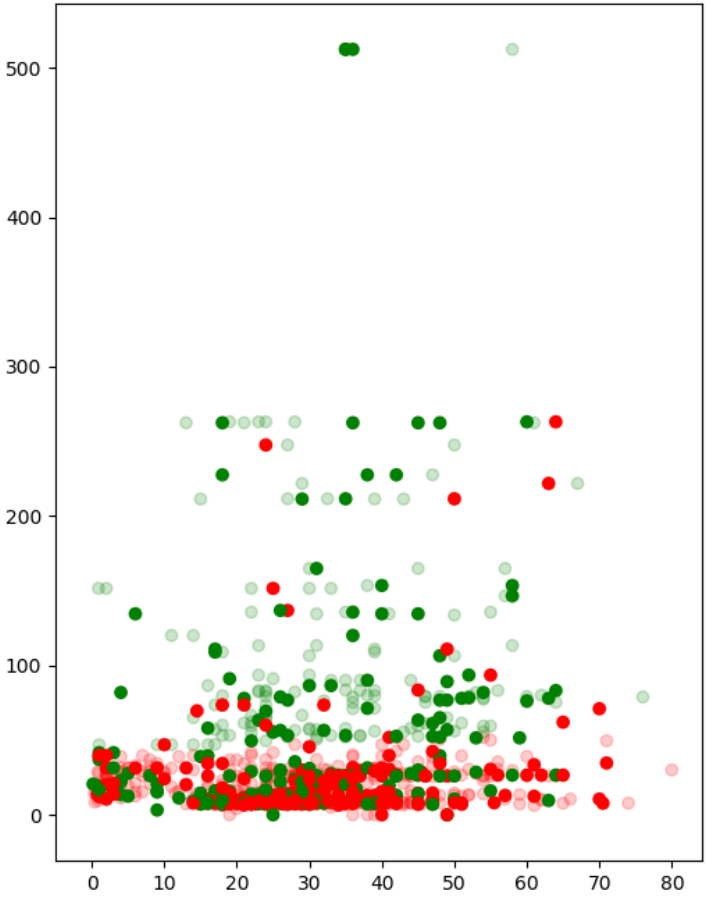
Exercice

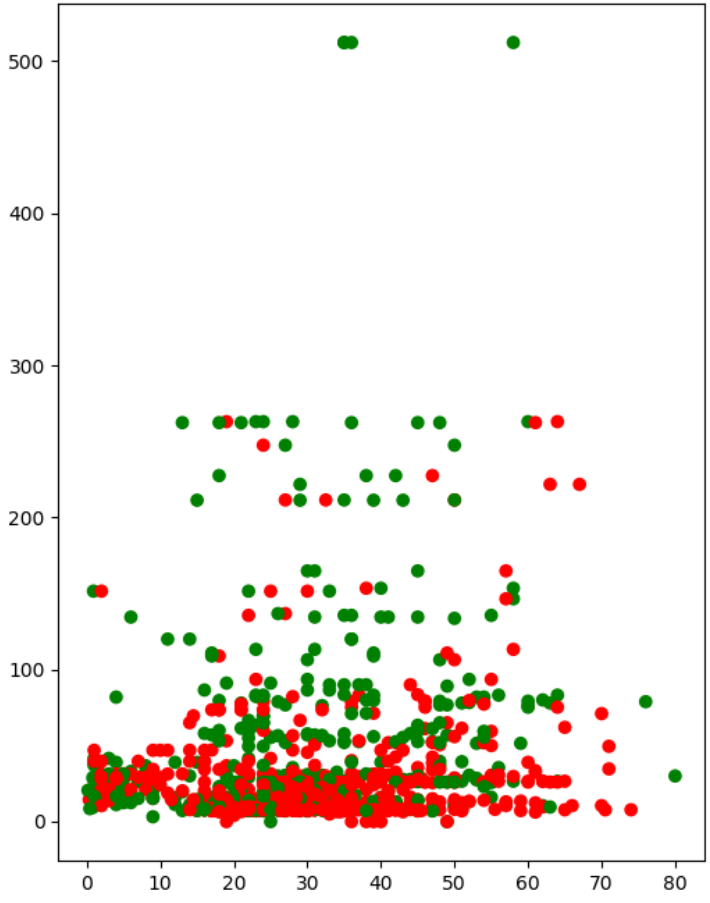
Effectuez la même manipulation avec les données age et fare du titanic, en vérifiant si vous prédisez bien survived.

Exercice

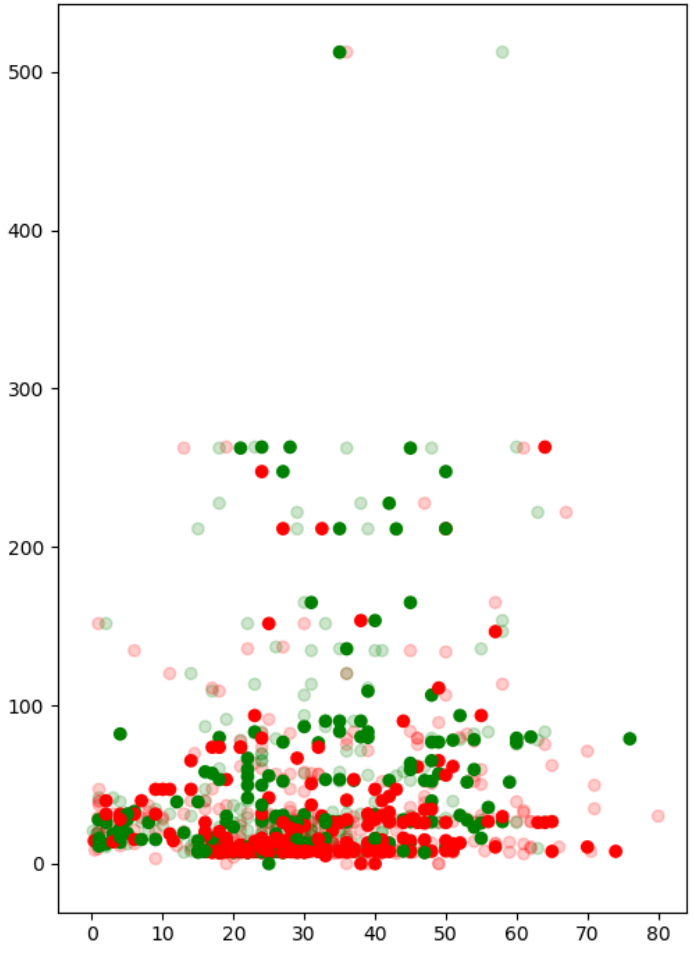
Même chose, mais en intégrant la variable sex (numérisée) dans l'apprentissage.

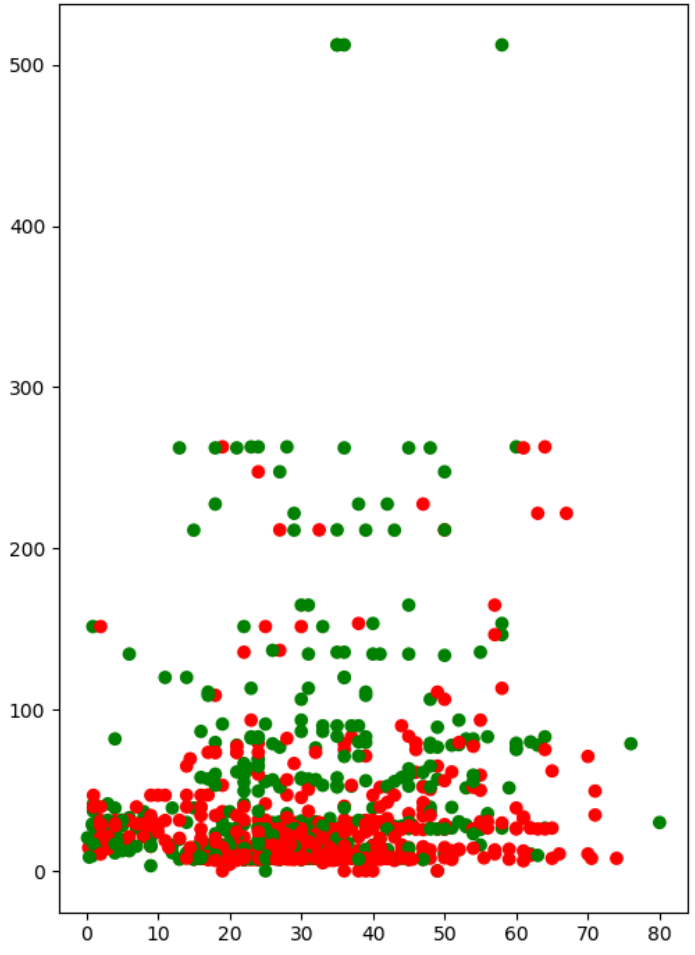
Résultat attendu (1)





Résultat attendu (2)





Solution (1)

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from random import shuffle
def color(x):
    return({0:'red',1:'green'}[x])
titanic = pd.read_csv('./C2/data1.csv')[['age','fare','survived']]
titanic = titanic.dropna().reset_index()
X,Y = titanic[['age','fare']], titanic.survived
i_train = list(range(len(X)))
shuffle(i_train)
X_train=X.loc[X.index.isin(i_train[:400])]
X_test=X.loc[X.index.isin(i_train[400:])]
Y_train=Y.loc[Y.index.isin(i_train[:400])]
s = SVC(kernel='linear')
s.fit(X_train,Y_train)
Y_pred = pd.Series(s.predict(X_test))
plt.subplot(121)
plt.scatter(X_test.age,X_test.fare,c=Y_pred.apply(color),alpha=0.2)
plt.scatter(X_train.age,X_train.fare,c=Y_train.apply(color))
plt.subplot(122)
plt.scatter(X.age, X.fare,c=Y.apply(color))
plt.show()
```

Solution (2)

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from random import shuffle
def color(x):
    return({0:'red',1:'green'}[x])
titanic = pd.read_csv('./C2/data1.csv')[['age','fare','sex','survived'
titanic = titanic.dropna().reset_index()
titanic.sex = titanic.sex.apply(lambda x:{'male':0,'female':1}[x])
X,Y = titanic[['age','fare','sex']], titanic.survived
i_train = list(range(len(X)))
shuffle(i_train)
X_train=X.loc[X.index.isin(i_train[:400])]
X_test=X.loc[X.index.isin(i_train[400:])]
Y_train=Y.loc[Y.index.isin(i_train[:400])]
s = SVC(kernel='linear')
s.fit(X_train,Y_train)
Y_pred = pd.Series(s.predict(X_test))
plt.subplot(121)
plt.scatter(X_test.age,X_test.fare,c=Y_pred.apply(color),alpha=0.2)
plt.scatter(X_train.age,X_train.fare,c=Y_train.apply(color))
plt.subplot(122)
plt.scatter(X.age, X.fare,c=Y.apply(color))
```

Régression logistique : objectif

On dispose d'un ensemble d'observations $(\tilde{X}, \tilde{Y})$ où $\tilde{X}$ est numérique et $\tilde{Y}$ binaire.

Régression logistique : objectif

On dispose d'un ensemble d'observations $(\tilde{X}, \tilde{Y})$ où $\tilde{X}$ est numérique et $\tilde{Y}$ binaire.

On veut produire une fonction explicite Y=f(X).

Régression logistique : objectif

On dispose d'un ensemble d'observations $(\tilde{X}, \tilde{Y})$ où $\tilde{X}$ est numérique et $\tilde{Y}$ binaire.

On veut produire une fonction explicite Y=f(X).

Dont le logit est linéaire :

$$\ln \frac{p(Y=1|X)}{p(Y=0|X)} = a_0 + a_1 X_1 + ... + a_k X_k$$

Exercice

Exercice

Importez les données du titanic. Entraînez une régression logistique sur la variable de survie à partir de l'age, du prix du billet et du genre.

Résultat attendu

0.7875 0.7782945736434108

1 0 0 1 0 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 0 0 1 1 1 1 0 0 0 1
1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 0 1 1 1

Solution

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
titanic = pd.read_csv('./C2/data1.csv')[['age','fare','sex','survived'
titanic = titanic.dropna().reset_index()
titanic.sex = titanic.sex.apply(lambda x:{'male':0,'female':1}[x])
X,Y = titanic[['fare','age','sex']], titanic.survived
i_train = list(range(len(X)))
shuffle(i_train)
X_train=X.loc[X.index.isin(i_train[:400])]
X_test=X.loc[X.index.isin(i_train[400:])]
Y_train=Y.loc[Y.index.isin(i_train[:400])]
Y_test=Y.loc[Y.index.isin(i_train[400:])]
logr = LogisticRegression()
logr.fit(X_train,Y_train)
print(logr.score(X_train,Y_train),logr.score(X_test,Y_test))
print(logr.predict(X_test)[:30])
print(np.array(Y_test[:30]))
```