In [1]:
```
import pandas as pd
X_train = pd.DataFrame(pd.read_csv('billets (1).csv',sep=';'))
X_test = pd.DataFrame(pd.read_csv('billets_production (1).csv'))
```

In [2]:
```
X_test.head()
```

Out[2]:

	diagonal	height_left	height_right	margin_low	margin_up	length	id
0	171.76	104.01	103.54	5.21	3.30	111.42	A_1
1	171.87	104.17	104.13	6.00	3.31	112.09	A_2
2	172.00	104.58	104.29	4.99	3.39	111.57	A_3
3	172.49	104.55	104.34	4.44	3.03	113.20	A_4
4	171.65	103.63	103.56	3.77	3.16	113.33	A_5

In [3]:
```
# Compléter les valeurs manquantes de billets train

#On régresse margin_low en fonction des autres variables de la df, de manière à retirer celles qui sont non significatives
#On constate ici que certains paramètres ne sont pas forcément pertinent car leur p-valeur n'est pas inférieure à 5 %
#De ce fait nous avons deux variables : is_genuine et margin_up
import statsmodels.formula.api as smf

#On effectue un régression linéaire simple
r_multi = smf.ols('margin_low~is_genuine+margin_up', data=X_train).fit()

#On cherche ensuite à faire une regression linéaire, en utilisant le modèle ci-dessus,
#On va donc pouvoir estimer les valeurs manquantes de margin_low

#On a ici l'index des valeurs manquantes
v_manq = X_train[X_train.isnull().any(axis=1)].index.tolist()
i=0
#On créé une boucle qui va déterminer le margin_low pour chaque ligne ayant une valeur manquante
for i in v_manq:
    prev = pd.DataFrame({'is_genuine': X_train.loc[[i], 'is_genuine'].item(),
                         'margin_up': X_train.loc[[i], 'margin_up'].item()}, index=[0])
    v_margin_low = r_multi.predict(prev)
    X_train.loc[[i], 'margin_low'] = round(v_margin_low[0], 2)
```

In [7]:

```
#On prédit la nature des billets testés

#On transforme la valeur booléenne en 0 (False) ou 1 (True)
X_train["is_genuine"] = X_train["is_genuine"].astype(int)

#On permet l'apprentissage des caractéristique pour True et False
#On garde uniquement les variables ayant des p-valeurs inférieurs à 5% :
import statsmodels.formula.api as smf
import statsmodels.api as sm
#On précise ici le .Binomial() pour faire comprendre que l'on cherche à faire une régression logistique
reg_log = smf.glm('is_genuine~height_right+margin_low+margin_up+length',
                  data=X_train, family=sm.families.Binomial()).fit()

#On arrange le dataframe test selon les résultats de prédiction du billets
y_pred = []
X_index_list = X_test.index.tolist()

#Pour chaque billet
for i in X_index_list :
#On récupère les valeurs qui nous intéressent :
    b_prevoir = pd.DataFrame({'height_right': X_test.loc[[i], 'height_right'].item(),
                          'margin_low': X_test.loc[[i], 'margin_low'].item(),
                          'margin_up': X_test.loc[[i], 'margin_up'].item(),
                          'length': X_test.loc[[i], 'length'].item()}, index=[0])
#On prédit la valeur estimé
    prediction_billets = reg_log.predict(b_prevoir).item()
#On créé une boucle qui attribue vrai ou faux dans une liste
    if round(prediction_billets) == 1:
        y_pred.append('Vrai')
    else :
        y_pred.append('Faux')

#On ajoute cette liste dans une nouvelle colonne de manière à pouvoir lire la nature de chaque billet
X_test['Nature du billets'] = y_pred
X_test
```

Out[7]:

	diagonal	height_left	height_right	margin_low	margin_up	length	id	Nature du billets
0	171.76	104.01	103.54	5.21	3.30	111.42	A_1	Faux
1	171.87	104.17	104.13	6.00	3.31	112.09	A_2	Faux
2	172.00	104.58	104.29	4.99	3.39	111.57	A_3	Faux
3	172.49	104.55	104.34	4.44	3.03	113.20	A_4	Vrai
4	171.65	103.63	103.56	3.77	3.16	113.33	A_5	Vrai