POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

DUPLAT CLÉMENCE
2407959

Devoir 3

MTH6312 - MÉTHODES STATISTIQUES D'APPRENTISSAGE

October 11, 2024

1 Question 1

Pour cette question, comme au devoir 2, vous devez d'abord obtenir vos données personnalisées (`mondata1`) en fonction de votre matricule (voir les instructions à cet effet sur le site). Il s'agit d'un échantillon aléatoire de 500 observations de la base de données `Wage` du package `ISLR2`.

Vous disposez ainsi de 500 observations sous la forme $\{(x_i, y_i), i = 1, \ldots, 500\}$, où y_i représente l'état de santé du travailleur (`health`) codée 1 (très bon), 0 (bon ou moins); $x_i = (x_{i1}, x_{i2})^\top$, où x_{i1} est l'âge (`age`) du travailleur et x_{i2} son salaire (`wage`).

Pour les données décrites ci-dessus, après avoir déterminé le degré de flexibilité approprié de chaque méthode, on veut sélectionner la meilleure méthode de classification parmi les suivantes : le KNN, la régression logistique, l'analyse discriminante linéaire et l'analyse discriminante quadratique.

1a.

a) KNN.

1. En utilisant les techniques de validation croisée LOOCV et 5-Fold CV sur les 500 observations, estimer le taux d'erreur test pour différentes valeurs du nombre de voisins, K, avec $K = 1, 2, \ldots, 180$. Tracer la courbe du taux d'erreur en fonction de $1/K$.

2. Compte tenu des résultats ci-dessus, quelle valeur du nombre de voisins K devrait-on utiliser pour la classification des données du contexte par le KNN? Justifier brièvement.

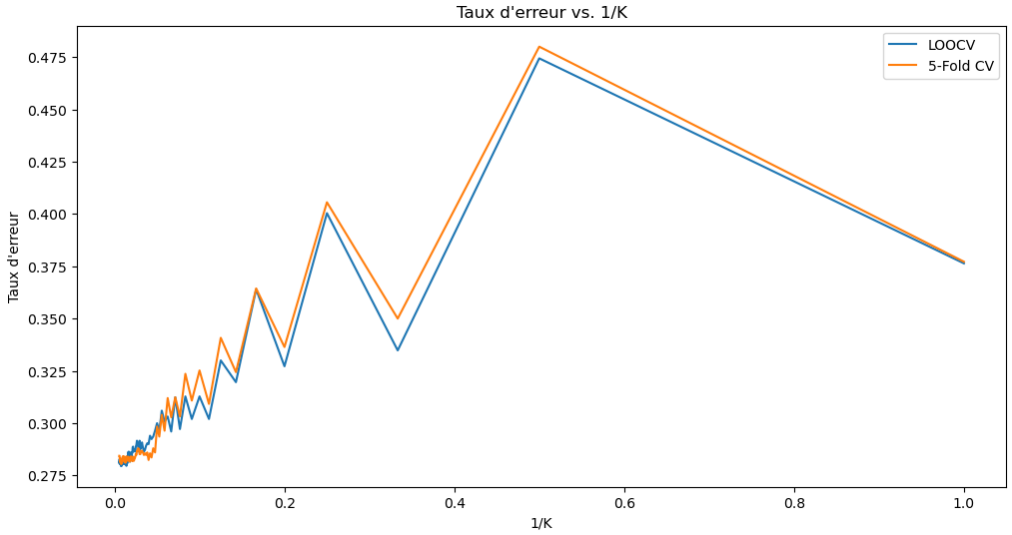

Figure 1: courbe du taux d'erreur en fonction de $1/K$

```
K optimal pour LOOCV: 72
K optimal pour 5-Fold CV: 128
```

On voit que le K optimal pour les deux méthodes est différent. La technique de validation croisée LOOCV prend une seule observation pour tester le modèle (et donc toutes les autres sont pour pour l'entraînement) et va répeter ceci pour les 500 observations. La technique de 5-Fold CV va diviser les observations en 5 ensembles. Un ensemble est utilisé pour tester le modèle (et donc les 4 autres sont pour pour l'entraînement), et va répeter ceci 5 fois. On voit donc que loocv sera bien plus précis que 5-Fold CV mais prendra beaucoup plus de temps de compilation (car nécessite 100 fois plus d'entraînement). On voit sur la Figure 1 que les 2 courbes d'erreurs sont très proches l'une de l'autre, montrant que l'on peut prendre un $K = 72$ pour les 2, tout en ayant une erreur relativement basse. De plus cela montre qu'il est plus justicieux d'utiliser 5-Fold CV pour avoir une bonne précision mais un temps de calcul plus rapide.

1b.

b) Régression logistique.

Dans les équations suivantes $p(x)$ représente $p(x;\beta) = P(Y = 1 \mid X = x)$, la probabilité que le travailleur soit en très bonne santé étant donné les mesures $x = (x_1, x_2)$ observées.

On envisage deux modèles de régression logistique dont les équations sont :

- Modèle 1 : $\ln\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$
- Modèle 2 : $\ln\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2$

1. En utilisant les techniques de validation croisée LOOCV et 5-Fold CV sur les 500 observations, estimer le taux d'erreur test pour chacun des 2 modèles.
2. Compte tenu des résultats ci-dessus, lequel des 2 modèles de régression logistique devrait-on utiliser pour la classification des données du contexte ? Justifier brièvement.

```
Taux d'erreur:
M1 LOOCV 0.28
M1 5-Fold 0.27880000000000005
M2 LOOCV 0.2804
M2 5-Fold 0.2804
```

Le modèle 1 est linéaire tandis que le modèle 2 a des termes quadratiques qui vont permettre de capturer des relations non linéaires. On voit que les taux d'erreurs sont proches pour les deux modèles, mais que le modèle 1 à quand même une valeure légèrement plus basse. Le modèle 2 risque aussi un surapprentissage. Ceci a lieu lorsque le modèle s'ajuste trop bien aux données d'entraînement et va donc capturer le signal et le bruit. On va donc préferer choisir le modèle 1, qui est plus simple.

1c.

c) Analyse discriminante.

1. En utilisant les techniques de validation croisée LOOCV et 5-Fold CV sur les 500 observations, estimer le taux d'erreur test de l'analyse discriminante linéaire (LDA) et celui de l'analyse discriminante quadratique (QDA).
2. Compte tenu des résultats ci-dessus, laquelle des deux analyses discriminantes devrait-on utiliser pour la classification des données du contexte ? Justifier brièvement.

```
Taux d'erreur:
LDA LOOCV  0.2792
LDA 5-Fold  0.2772
QDA LOOCV  0.286
QDA 5-Fold  0.28640000000000004
```

LDA et QDA sont toutes les deux des méthodes de classification qui vont reposer sur des hypothèses de distribution des données. L'hypothèse de LDA est que les données de chaque classe suivent une distribution gaussienne (avec une moyenne et variance spécifique) mais toutes les classes ont la même matrice de covariance. Au contraire QDA va émettre l'hypothèse que chaque classe peut posséder sa propre matrice de covariance. Cette différence d'hypothèse va avoir une influence sur les frontières de décisions entre les différentes classes. La frontière va être linéaire dans le cas de LDA et quadratique pour QDA. En fonction de la manière dont les classes sont séparées, il est judicieux d'utiliser l'une ou l'autre méthode. Notons aussi que LDA est une méthode plus simple et robuste mais moins flexible. On voit que le taux d'erreur est inférieur pour LDA dans notre cas. Les données sont donc probablement mieux séparées par une frontière linéaire. On va donc préférer utiliser LDA.

1d.

d) Résumé graphique et comparaison des méthodes.

Tracer le nuage des 500 points (2 couleurs de votre choix) et ajouter au graphique les courbes (trois en tout, similaires à celles de la figure 5.7 page 207 dans `ISLr`) séparant les deux classes dans chacun des cas suivants :

- le KNN (avec la valeur optimale retenue du nombre de voisins K),
- le modèle de régression logistique retenu (l'un des deux modèles considérés),
- l'analyse discriminante retenue (LDA ou QDA).

Comparer les trois méthodes en utilisant leurs matrices de confusion et indices de performance.

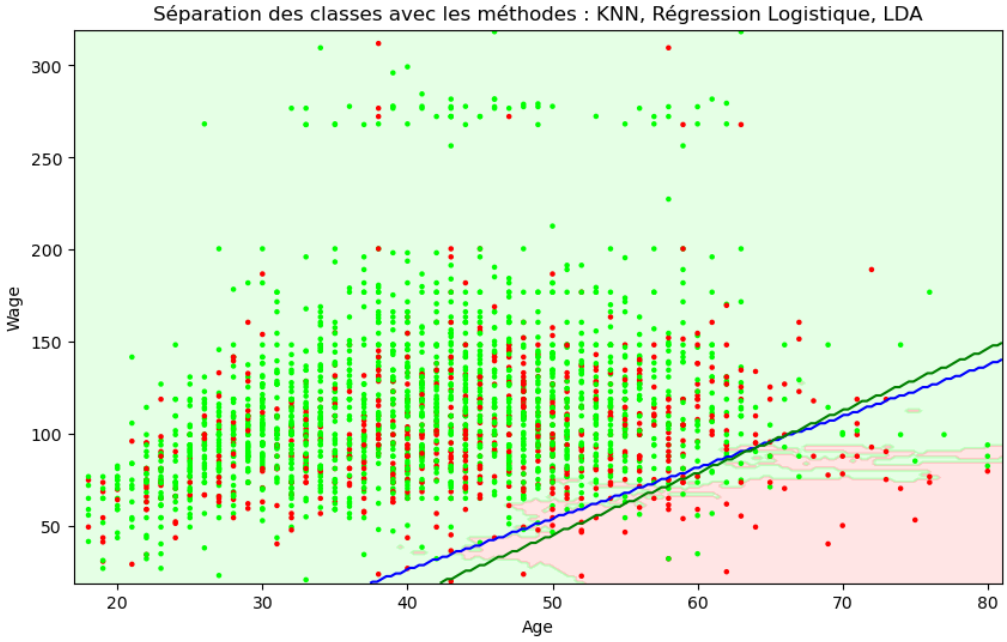

Figure 2: comparaison des méthodes pour classfier les données health en très bon (point vert) ou bon ou moins (point rouge): KNN (fond rouge et vert), régression logistique (bleu) et LDA (vert)

```
Matrice de confusion pour KNN :
Truth       0     1
Predicted
0          42   670
1          16  1772

Indices de performance KNN
              precision    recall  f1-score   support

           0       0.72      0.06      0.11       712
           1       0.73      0.99      0.84      1788

    accuracy                           0.73      2500
   macro avg       0.72      0.53      0.47      2500
weighted avg       0.73      0.73      0.63      2500

Matrice de confusion pour régression logistique:
Truth       0     1
```

```
Predicted
0          51   661
1          35  1753

Indices de performance régression logistique
              precision    recall  f1-score   support

           0       0.59      0.07      0.13       712
           1       0.73      0.98      0.83      1788

    accuracy                           0.72      2500
   macro avg       0.66      0.53      0.48      2500
weighted avg       0.69      0.72      0.63      2500

Matrice de confusion pour LDA :
Truth       0     1
Predicted
0          45   667
1          30  1758

Indices de performance LDA
              precision    recall  f1-score   support

           0       0.60      0.06      0.11       712
           1       0.72      0.98      0.83      1788

    accuracy                           0.72      2500
   macro avg       0.66      0.52      0.47      2500
weighted avg       0.69      0.72      0.63      2500
```

Pour rappel:

- La précision mesure la proportion de prédictions correctes parmi les observations prédites comme positive
- Le recall mesure la proportion d'observations positives correctement identifiées comme positive
- Le F1-score combine précision et recall
- Classe 1 : health = très bon
- Classe 0 : health = bon ou moins

Pour les 3 méthodes, un accuracy de environ 70% est cohérent avec les taux d'erreurs trouvé auparavant. Cependant cette mesure ne va pas prendre en compte le désequilibre entre les classes, qu'on peut clairement observé sur la Figure 2 (beaucoup moins de points appartenant à la classe 0 qu'à la classe 1). On va donc regarder plus près les recall et la précision de chaque classe.
On voit que le recall pour la classe 1 est très élevé pour les 3 méthodes, surtout pour le KNN qui a un recall de 99%. Le recall pour la classe 0 est cependant extrêment faible pour les 3 méthodes. Pour la précision de la classe 0, on voit qu'elle est relativement élevé pour la méthode KNN, et un peu plus faible pour les 2 autres méthodes. On peut donc conclure que le modèle a du mal à prédire la classe 0 correctement.

2 Question 2

2a.

Donner l'expression d'un estimateur $\hat{\theta}$ de θ en fonction des n observations.

On estime θ en prenant l'espérance du minimum entre les 3 équations données. Cela nous donne $\theta_{estime} = 0.010805041750993593$.

2b.

En utilisant la technique de ré-échantillonnage Bootstrap (fonction boot()) avec 3500 répétitions, donner une estimation ponctuelle $\hat{\theta}$. Donner ensuite une estimation du biais et une estimation de l'écart type (erreur-type) de $\hat{\theta}$

```
Estimation bootstrap de theta: 0.010803673955830217
Biais estimé: -1.3677951633764979e-06
Erreur-type estimée: 0.00044649208848950147
```

On voit que l'estimation bootstrap est très proche de celle initiale, ce qui comfirme que cela semble correcte. On voit également que le biais est très proche de 0, ce qui signifie que θ n'est pas biaisé de manière significative. L'erreur-type estimé est également proche de 0, ce qui montre que les estimations sont relativement précises.

2c.

Déduire des résultats qui précèdent un intervalle de confiance pour θ au niveau de confiance 95%. Commenter brièvement.

```
Intervalle de confiance:  [0.010788874690969171, 0.010818473220691266]
```

On voit que notre intervalle de confiance est très étroit. Cela signifie que les estimations de θ sont relativement précises et peu variable. On voit également que notre estimation initiale de θ se trouve dans cet intervalle, ce qui comfirme que notre estimation est stable.

3 Question 3

(a)

Use the `rnorm()` function to generate a predictor X of length $n = 100$, as well as a noise vector ε of length $n = 100$.

Ceci a été effectué en R car la fonction regsubsets() est très utile pour cet exercice. (voir Appendix). Par facilité j'ai choisi pour mes lois normales:

```
mu <- 0  #moyenne nulle -> centré autour de 0
sigma <- 1  #écart type pour x
sigma_epsilon <- 1  #écart type pour l'erreur
```

On voit que ceci est équivalent que d'utiliser rnorm().

(b)

Generate a response vector Y of length $n = 100$ according to the model

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \varepsilon,$$

where β_0, β_1, β_2, and β_3 are constants of your choice.

Les constantes choisies sont les suivantes:

```
beta_0 = 1
beta_1 = 2
beta_2 = -2
beta_3 = -0.5
```

(c)

Use the `regsubsets()` function to perform best subset selection in order to choose the best model containing the predictors $X, X^2, \ldots, X^{10}$. What is the best model obtained according to C_p, BIC, and adjusted R^2? Show some plots to provide evidence for your answer, and report the coefficients of the best model obtained. Note you will need to use the `data.frame()` function to create a single data set containing both X and Y.

```
Meilleur modèle obtenu selon:
Cp: 6 BIC: 3 Adjusted R^2: 7

Coefficient Cp:
(Intercept)          X1          X2          X4          X5          X6          X7
 1.40808315  1.68097732 -3.26027077  0.81982076 -0.29210333 -0.14400564  0.04943268
 Coefficient BIC:
(Intercept)          X1          X2          X3
  1.2261794   1.7556894  -2.1528117  -0.4310617
Coefficients R^2 ajusté
(Intercept)           X1           X2           X3           X4           X6           X8
 1.514869889  1.795195656 -4.309385271 -0.434923880  2.266358440 -0.786962353  0.104939365
         X10
-0.004704147
```

Le critère Cp exige 6 prédicteurs, BIC exige 3 prédicteurs et R^2 ajusté en exige 7. Le BIC va donc favoriser un modèle plus simple (celui initial d'ailleurs).

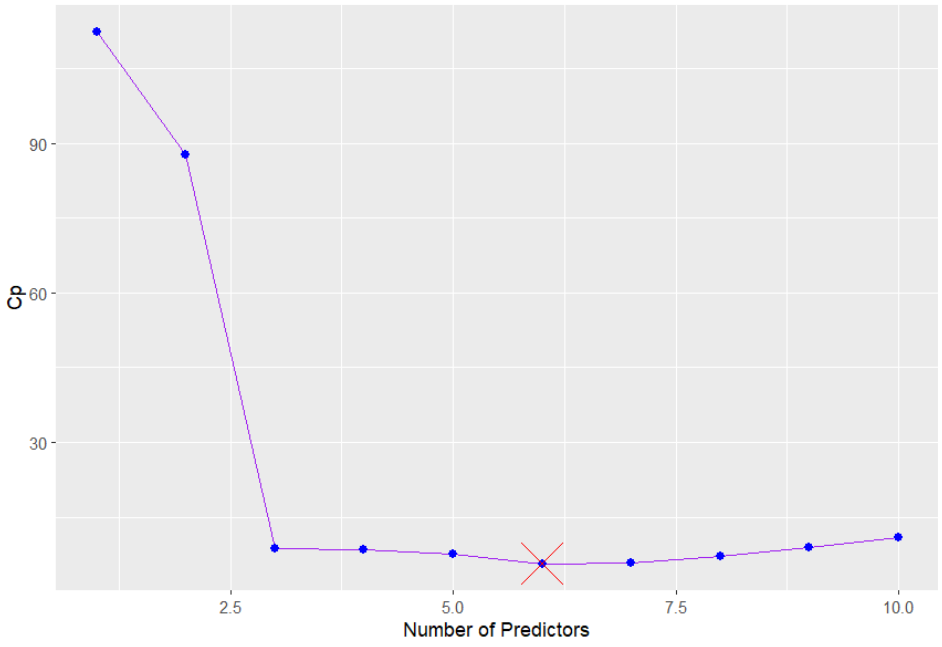

Figure 3: Cp

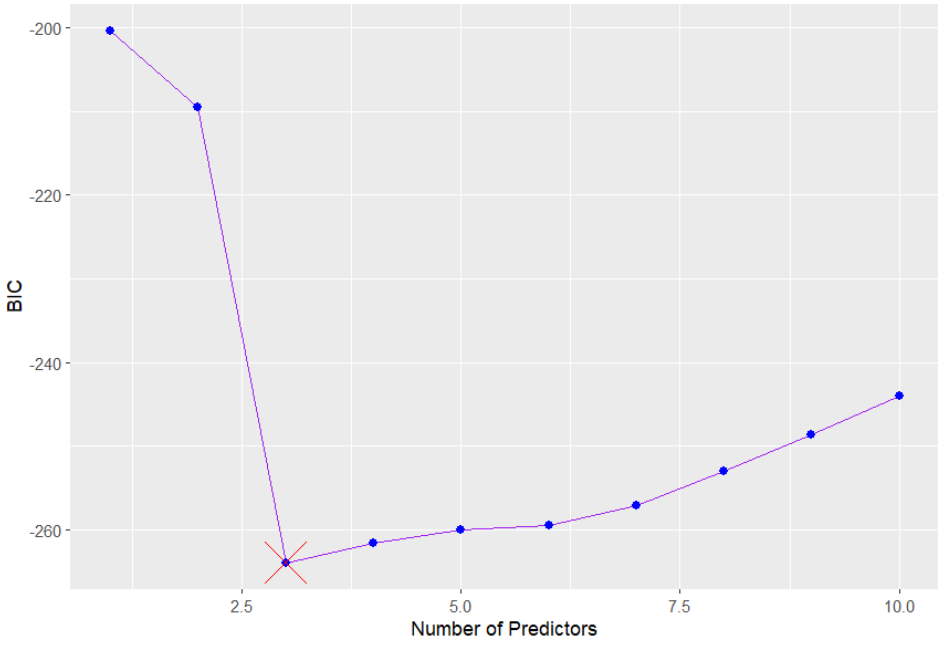

Figure 4: BIC

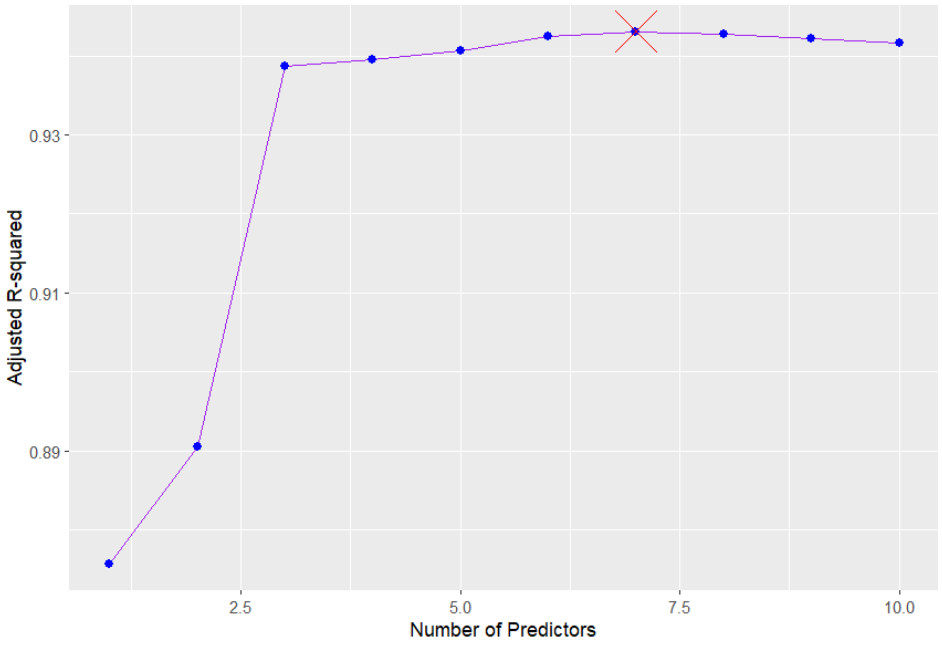

Figure 5: R2 adjusted

(d)

Repeat (c), using forward stepwise selection and also using backward stepwise selection. How does your answer compare to the results in (c)?

```
Forward:
Cp: 8 BIC: 3 Adjusted R^2: 8

Coefficients Cp:
(Intercept)            X1            X2            X3            X4            X6
 1.5093550601  1.8710459949 -4.2053751093 -0.4923692520  2.0968410282 -0.7040601418
          X8            X9           X10
 0.0910061998  0.0007609147 -0.0041968017

Coefficients BIC:
(Intercept)          X1          X2          X3
  1.2261794   1.7556894  -2.1528117  -0.4310617

Coefficients R^2 ajusté:
  (Intercept)            X1            X2            X3            X4            X6
 1.5093550601  1.8710459949 -4.2053751093 -0.4923692520  2.0968410282 -0.7040601418
          X8            X9           X10
 0.0910061998  0.0007609147 -0.0041968017
```

On voit que le nombre de prédicteurs pour BIC reste le même tandis qu'il augmente pour Cp et R^2 ajusté.

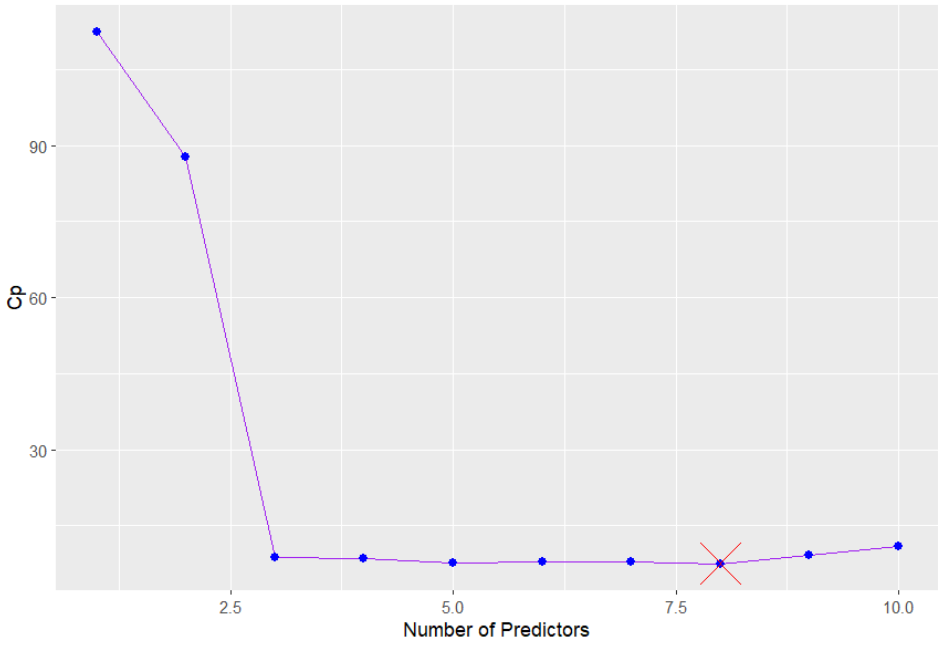

Figure 6: Cp forward

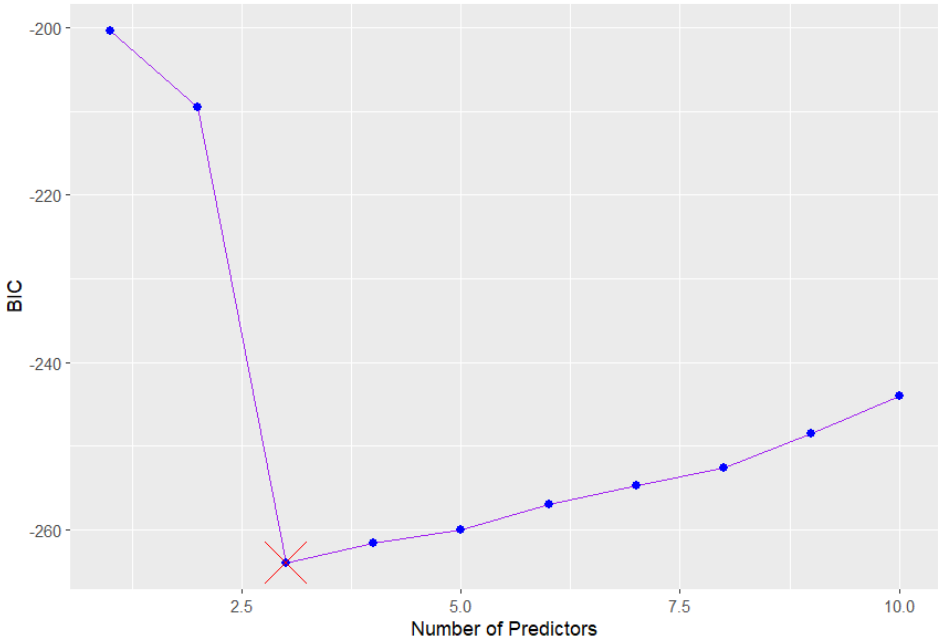

Figure 7: BIC forward

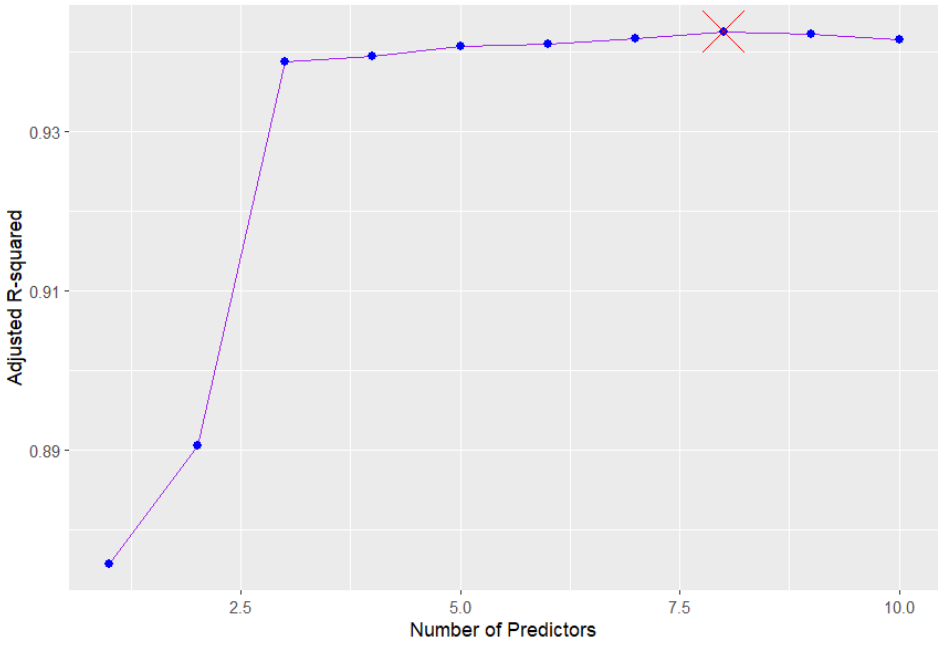

Figure 8: R2 ajusté forward

```
Backward:
Cp: 6 BIC: 4 Adjusted R^2: 8

Coefficients Cp:
(Intercept)          X1          X2          X4          X5          X6          X7
 1.40808315  1.68097732 -3.26027077  0.81982076 -0.29210333 -0.14400564  0.04943268

Coefficients BIC:
(Intercept)          X1          X2          X5          X7
 1.22977395  1.51880284 -2.14557930 -0.14776876  0.01161028

Coefficients R^2 ajusté:
(Intercept)          X1          X2          X4          X5          X6          X7
 1.53329227  1.74262270 -4.45008981  2.39532286 -0.49480279 -0.77638069  0.17833544
         X8          X9
 0.07740387 -0.02010948
```

Cette fois-ci le nombre de prédicteurs pour Cp reste le même, tandis que ceux pour BIC et R^2 ajusté augmentent de 1.

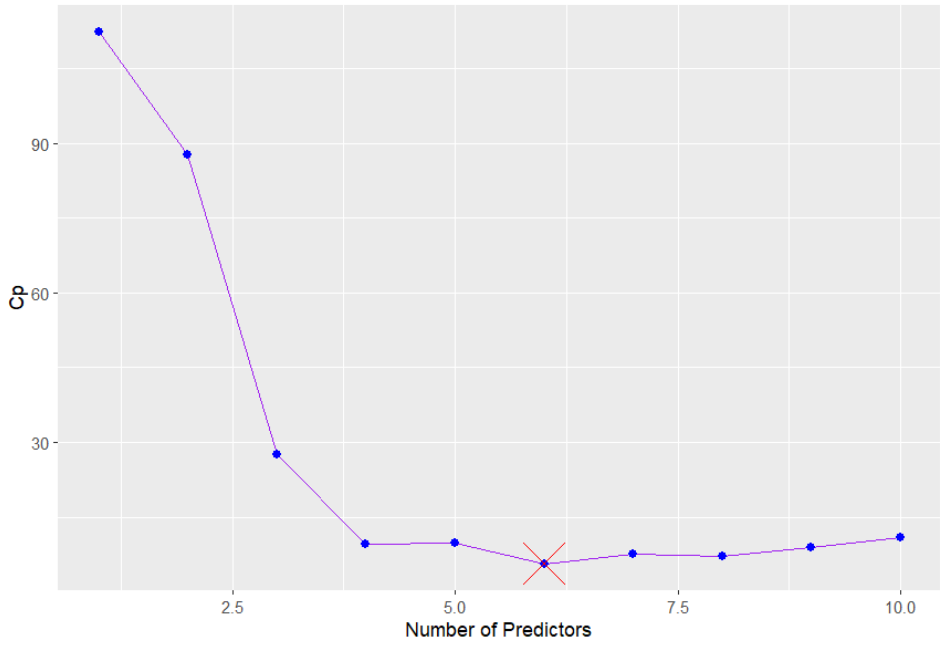

Figure 9: Cp backward

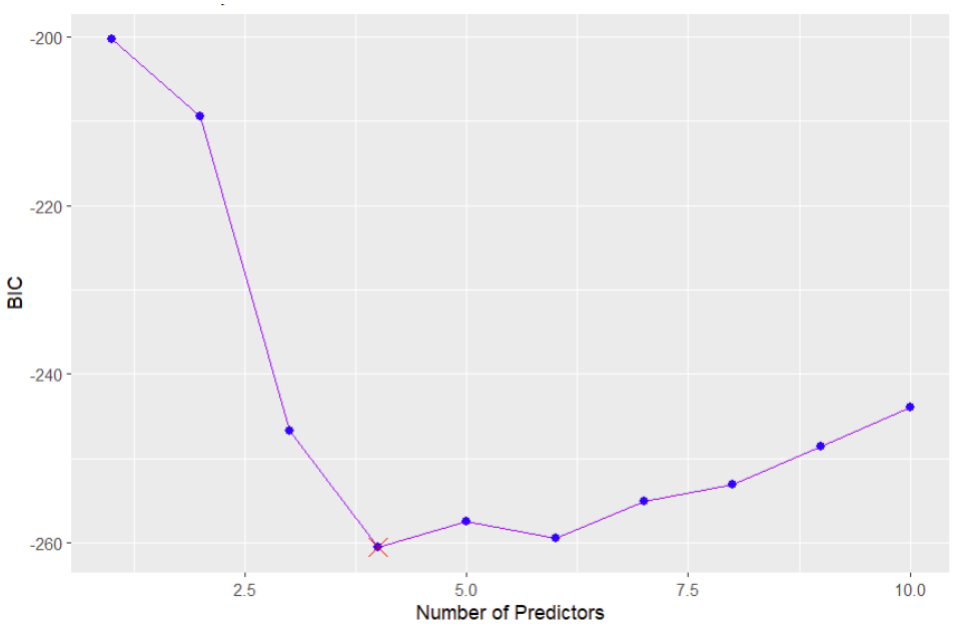

Figure 10: bic backward

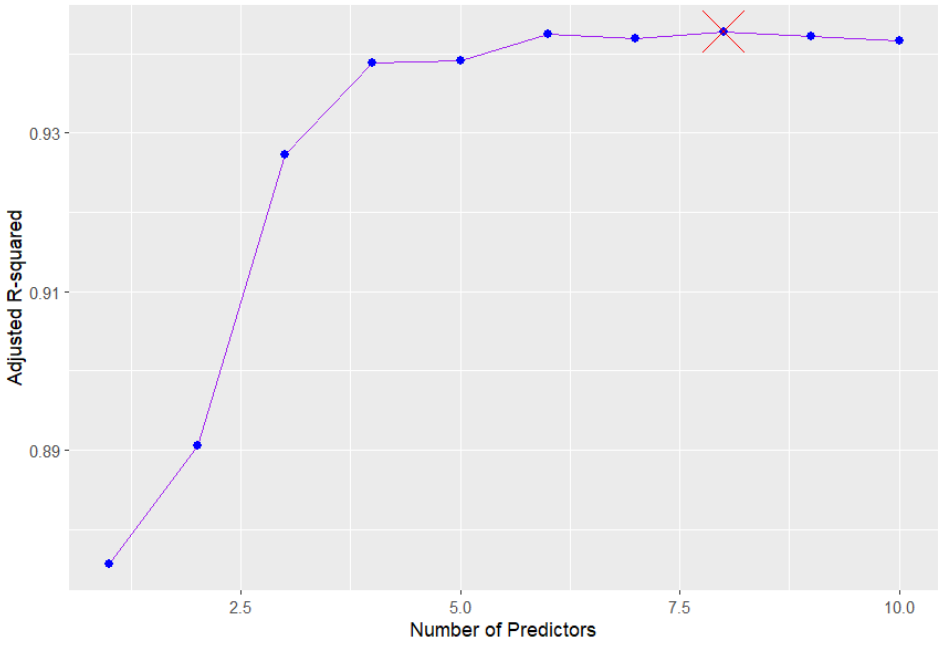

Figure 11: R2 ajusté backward

(e)

Now fit a lasso model to the simulated data, again using $X, X^2, \ldots, X^{10}$ as predictors. Use cross-validation to select the optimal value of λ. Create plots of the cross-validation error as a function of λ. Report the resulting coefficient estimates, and discuss the results obtained.

J'ai décidé de rester, comme à la question 1, à un 5-Kfold pour la validation croisée.

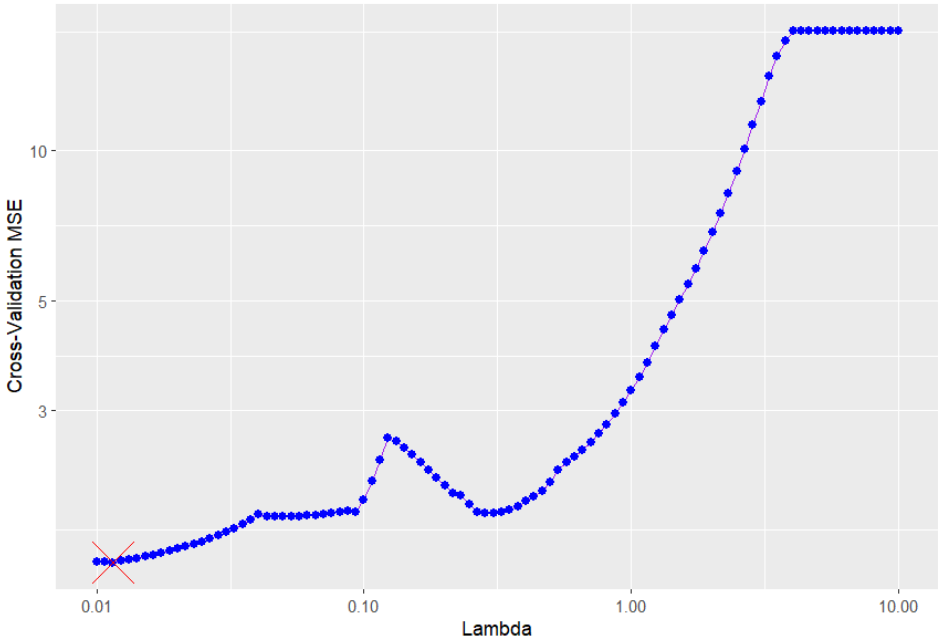

Figure 12: Lasso en utilisant à nouveau $X, X^2, \ldots, X^{10}$

```
Lambda optimal:
```

```
0.01149757

Coefficients:
                       s1
(Intercept)  1.202711e+00
1            1.766697e+00
2           -2.055732e+00
3           -4.414126e-01
4           -3.216083e-02
5            0.000000e+00
6            0.000000e+00
7            0.000000e+00
8            0.000000e+00
9            7.913510e-08
10           4.194486e-05
```

On voit que les coefficient pour X_5 et X_{10} sont relativement proche de 0 (ou égal à 0). Les coefficients pour X_3 et X_4 sont un peu plus grand mais toujours proche de 0 tandis que ceux entre X_1 et X_2 sont plus larges. Lasso réduit donc la complexité du modèle en considérant certains termes comme non significatif. On voit que les coefficients estimé sont aussi proche de nos coefficients initiaux (dans la section b). SI on considère donc les 3 premiers termes, on voit que cela correspond à notre fonction cubique initiale.

(f)

Now generate a response vector Y according to the model

$$Y = \beta_0 + \beta_7 X^7 + \varepsilon,$$

and perform best subset selection and the lasso. Discuss the results obtained.

```
Cp: 6 BIC: 3 Adjusted R^2: 7

Lambda Optimal:
0.01

Coefficients:
(Note . veut dire que le coefficient est nul)
                  s1
(Intercept) 14.93805
1            .
2            .
3            .
4            .
5            .
6            .
7            4.85440
8            .
9            .
10           .
```

On voit que les coefficients estimés sont proche des coefficients initiaux.

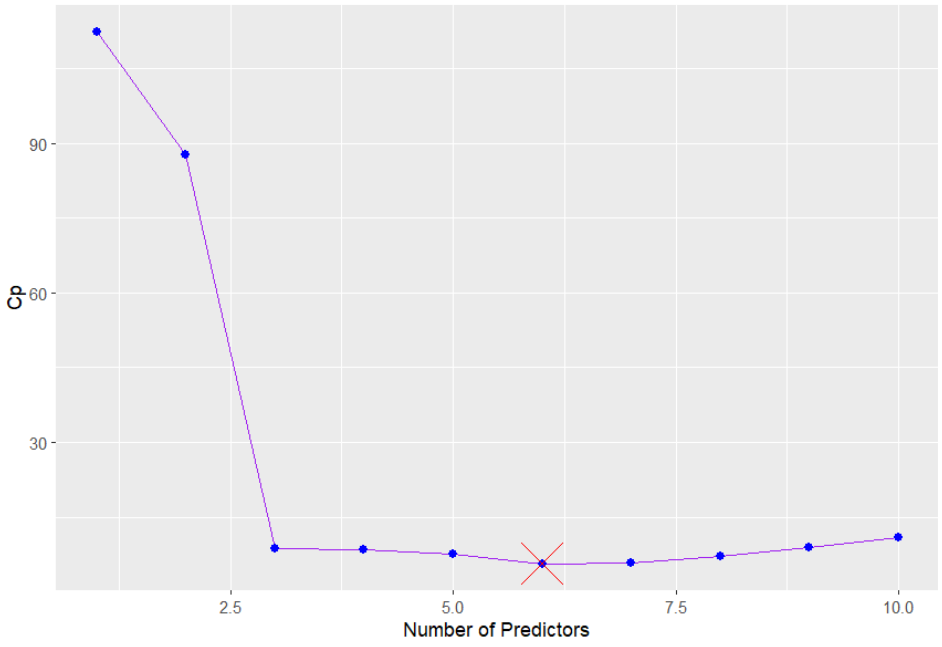

Figure 13: Cp

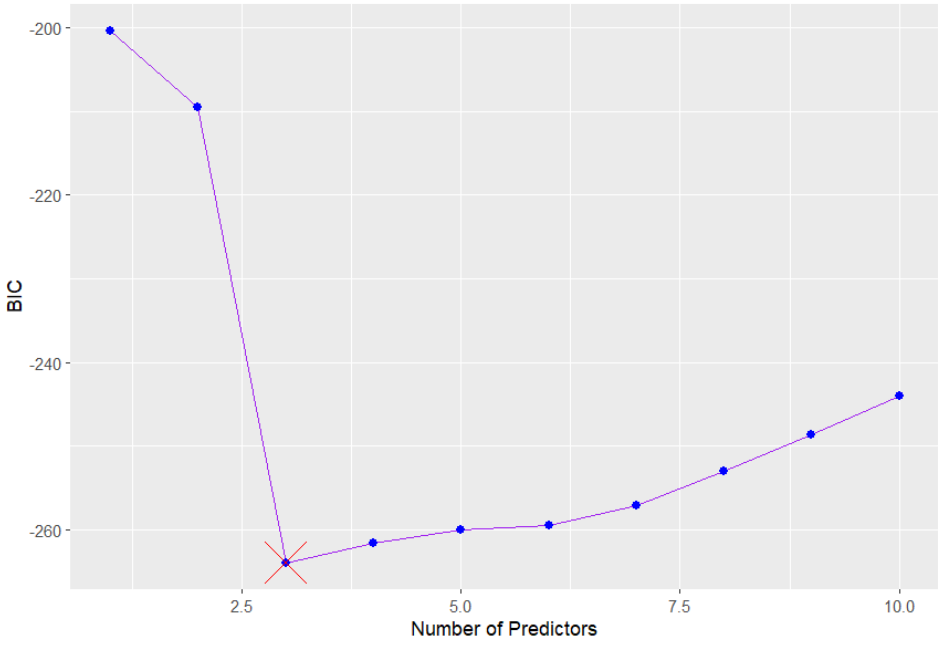

Figure 14: BIC

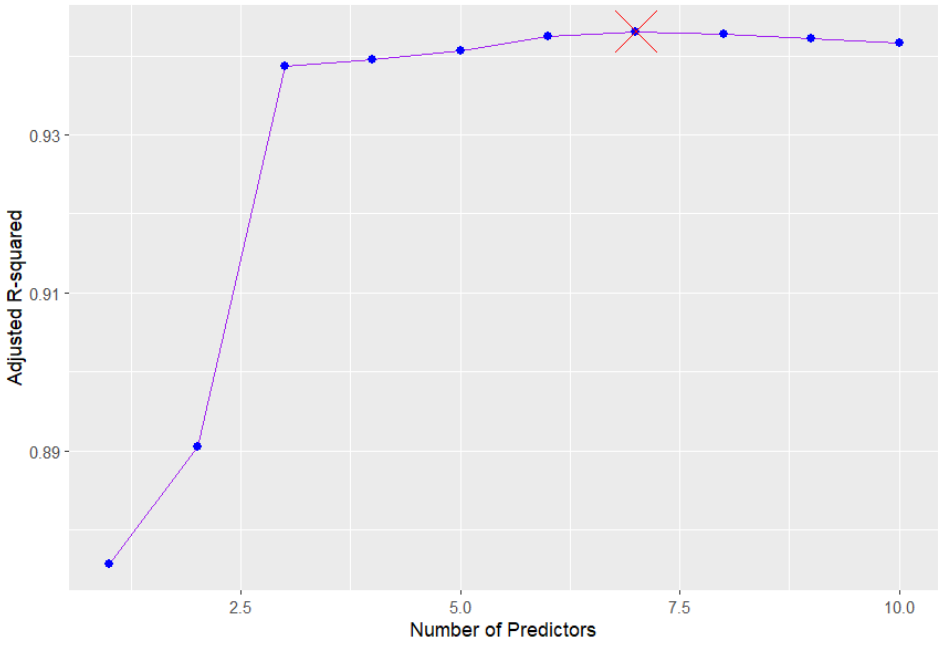

Figure 15: R2 ajusté

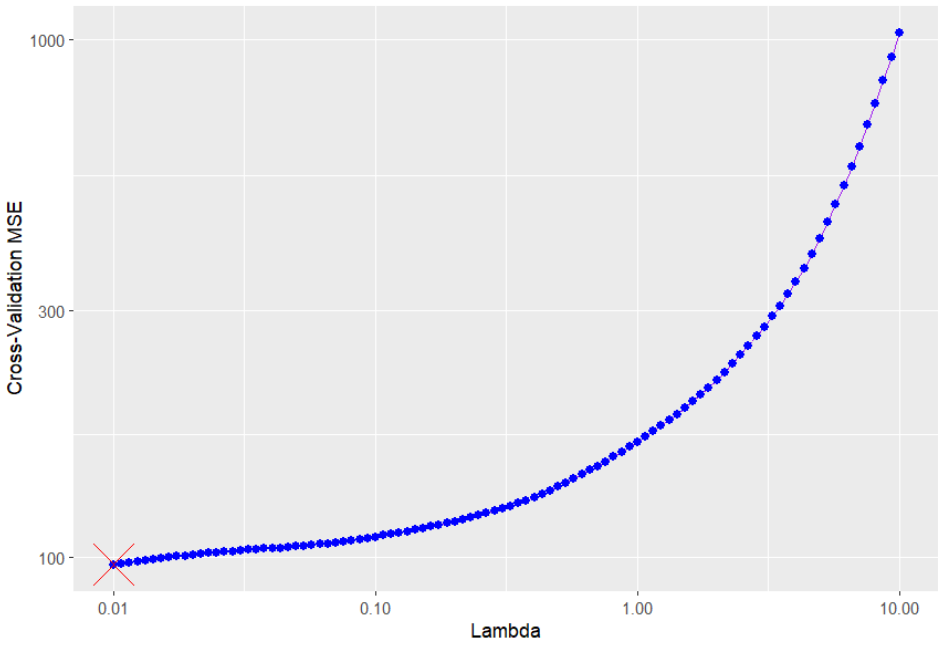

Figure 16: Lasso avec $Y = \beta_0 + \beta_7 X^7 + \varepsilon$

4 Appendix: code

```
from sklearn.model_selection import train_test_split
#Génération de données basé sur ma matricule
#utilisation du wage dataset
wage = load_data("Wage")
#print(wage.columns)
#Index(['year', 'age', 'maritl', 'race', 'education', 'region', 'jobclass','health', 'health_ins', 'log
wage['health'] = wage['health'].apply(lambda x: 1 if x == "2. >=Very Good" else 0)
matricule = 2405979
np.random.seed(matricule)
train_data, valid_data = train_test_split(wage, test_size=500, random_state=matricule)
X_train = train_data[['age', 'wage']]
y_train = train_data['health']

#loocv: on divise les données de manière à ce que chaque observation soit utilisée comme un cas de test
#-> on entraîne n-1 obs, on test avec 1 obs et ce processus est répeté n fois
# + : elle utilise toutes les données -> bonne estimation de l'erreur
# - : coûteuse si n est grand
# 5Fold: on divise les données en 5 sous-ensembles
# -> on entraine le modèle sur 4 sous-ensembles, on test sur un -> répète le processus 5 fois
# + : elle utilise moins de données que loocv -> moins coûteuse
# - : -> moins précis aussi
erreur_loocv = []
erreur_5fold = []
for k in range(1,181):
    knn = KNeighborsClassifier(n_neighbors=k)
    # loocv
    #https://www.statology.org/leave-one-out-cross-validation-in-python/
    #Provides train/test indices to split data in train/test sets. Each sample is used once as a test s
    # Note: LeaveOneOut() is equivalent to KFold(n_splits=n) and LeavePOut(p=1) where n is the number o
    loo = LeaveOneOut()
    loo_scores = cross_validate(knn, X_train, y_train, cv=loo,scoring='accuracy')
    erreur_loocv.append(1-np.mean(loo_scores['test_score']))
    # 5fold Cv
    kf = KFold(n_splits=5, shuffle=True, random_state=matricule)
    kfold_scores = cross_validate(knn, X_train, y_train, cv=kf,scoring='accuracy')
    # kfold_error = 1 - kfold_scores.mean()
    erreur_5fold.append(1- np.mean(kfold_scores['test_score']))

plt.figure(figsize=(12, 6))
plt.plot([1/k for k in range(1, 181)], erreur_loocv, label='LOOCV')
plt.plot([1/k for k in range(1, 181)], erreur_5fold, label='5-Fold CV')
plt.xlabel('1/K')
plt.ylabel('Taux d\'erreur')
plt.title('Taux d\'erreur vs. 1/K')
plt.legend()
plt.show()
optimal_k_loocv = erreur_loocv.index(min(erreur_loocv)) + 1
optimal_k_kfold = erreur_5fold.index(min(erreur_5fold)) + 1

print(f"K optimal pour LOOCV: {optimal_k_loocv}")
print(f"K optimal pour 5-Fold CV: {optimal_k_kfold}")
#section 4.7.2 from ISLP website
#modèle 1 : b0 + b1*x1+b2*x2
#modèle 2 : b0 + b1*x1+b2*x2 +b3*x1^2 ¨+b4*x2^2
```

```
X_train_1 = sm.add_constant(X_train)
X_train_2 = X_train.copy()
#on inclut les termes quadratiques
X_train_2['age^2']= X_train['age']**2
X_train_2['wage^2']= X_train['wage']**2
X_train_2 = sm.add_constant(X_train_2)
loo = LeaveOneOut()
kf = KFold(n_splits=5, shuffle=True, random_state=matricule)

def mean_error(X, y, cv):
    errors = []
    #pour chacun des fold on va calucler l'erreur puis prendre la moyenne
    for i, j in cv.split(X):
        X_train, X_test= X.iloc[i],X.iloc[j]
        y_train, y_test= y.iloc[i],y.iloc[j]
        #from ISLP: The syntax of sm.GLM() is similar to that of sm.OLS(), except that
        # we must pass in the argument family=sm.families.Binomial() in order to
        # tell statsmodels to run a logistic regression rather than some other type of
        # generalized linear model.
        #on fit un logistique régression comme dans le livre
        glm = sm.GLM(y_train, X_train, family=sm.families.Binomial())
        result = glm.fit()
        probs= result.predict(X_test)>0.5 #est-ce que ça va être utile de changer ce 0.5 ?
        #voir si c'est classifié de la même manière
        error = np.mean(probs!=y_test)
        errors.append(error)
    return np.mean(errors)
#enft j'aurais juste pu utiliser à nv cross_validate mais pas sûre si il va bien faire la logistique ré
#si je précise pas sm.families.Binomial()
model1_loo = mean_error(X_train_1, y_train, loo)
model1_kf = mean_error(X_train_1, y_train, kf)
model2_loo = mean_error(X_train_2, y_train, loo)
model2_kf = mean_error(X_train_2, y_train, kf)
print("M1 LOOCV", model1_loo)
print("M1 5-Fold", model1_kf)
print("M2 LOOCV", model2_loo)
print("M2 5-Fold", model2_kf)

#section 4.7.3 et 4.7.4
# Linear Discriminant Analysis (LDA)
# SincetheLDAestimatorautomaticallyaddsanintercept,weshouldre
# movethecolumncorrespondingtotheinterceptinbothX_trainandX_test.

loo = LeaveOneOut()
kf = KFold(n_splits=5, shuffle=True, random_state=matricule)
lda_model = LDA(store_covariance=True)
qda_model = QDA(store_covariance=True)
#on va faire pareil que l'exo précedent mais sans binomial
#(à nv je pense qu'on aurait pu utiliser cross_validate mais bon)
def mean_error(model, X, y, cv):
    errors = []
    #pour chaque fold
    for i, j in cv.split(X):
        model.fit(X.iloc[i], y.iloc[i])
        predictions = model.predict(X.iloc[j])
        error = np.mean(predictions!=y.iloc[j])
```

```
        errors.append(error)
    return np.mean(errors)

lda_loocv_error = mean_error(lda_model, X_train, y_train, loo)
lda_kfold_error = mean_error(lda_model, X_train, y_train, kf)

qda_loocv_error = mean_error(qda_model, X_train, y_train, loo)
qda_kfold_error = mean_error(qda_model, X_train, y_train, kf)

print("LDA LOOCV ", lda_loocv_error)
print("LDA 5-Fold ", lda_kfold_error)
print("QDA LOOCV ", qda_loocv_error)
print("QDA 5-Fold ", qda_kfold_error)

# comme au devoir 2
##source: https://ogrisel.github.io/scikit-learn.org/sklearn-tutorial/auto_examples/neighbors/plot_clas
# matricule = 2405979
#je vais redefinir X et y avec .values sinon j'ai des erreurs après pcq je prends pas les values
X = train_data[['age', 'wage']].values
y = train_data['health'].values

cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA'])
cmap_bold = ListedColormap(['#FF0000', '#00FF00'])
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
#arrange prend trop de mémoire donc je remplace en utilisant linspace
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200), np.linspace(y_min, y_max, 200))

#on fait comme au devoir 2 mais avec nos méthodes choisies
#knn, ici j'utilise le 72 que j'ai choisi à la question 1 a)
k = 72
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X, y)
Z_knn = knn.predict(np.c_[xx.ravel(), yy.ravel()])
Z_knn = Z_knn.reshape(xx.shape)
log_model = LogisticRegression()
log_model.fit(X, y)
Z_log = log_model.predict(np.c_[xx.ravel(), yy.ravel()])
Z_log = Z_log.reshape(xx.shape)
lda = LinearDiscriminantAnalysis()
lda.fit(X, y)
Z_lda = lda.predict(np.c_[xx.ravel(), yy.ravel()])
Z_lda = Z_lda.reshape(xx.shape)
plt.figure(figsize=(10, 6))
plt.contourf(xx, yy, Z_knn, cmap=cmap_light, alpha=0.3)
plt.contour(xx, yy, Z_log, levels=[0.5], colors='blue')
plt.contour(xx, yy, Z_lda, levels=[0.5], colors='green')
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold, s= 5)
plt.title("Séparation des classes avec les méthodes : KNN, Régression Logistique, LDA")
plt.xlabel('Age')
plt.ylabel('Wage')
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.show()
knn_prediction = knn.predict(X)
print("Matrice de confusion pour KNN :")
```

```
c1 = confusion_table(y, knn_prediction)
print(c1)
print(classification_report(y, knn_prediction))
pred_log = log_model.predict(X)
print("Matrice de confusion pour régression logistique:")
c2 = confusion_table(y, pred_log)
print(c2)
print(classification_report(y, pred_log))
pred_lda = lda.predict(X)
print("Matrice de confusion pour LDA :")
c3 = confusion_table(y, pred_lda)
print(c3)
print(classification_report(y, pred_lda))
# utiliser iris dataset qui est dans sklearn
iris = load_iris()
X = iris.data

# on veut calculer theta_prime qui est donné comme
#l'espérance du minimum entre : X2 + log(X1), X1 + X3  2X4, exp {|X1  X4|} , X2 + 3X3
#on en fait une fonction au cas ou si on veut le réutiliser plus tard
def theta_estimation(data):
    return np.mean(np.min([
        data[:, 1] + np.log(data[:, 0]),
        data[:, 0] + data[:, 2] - 2 * data[:, 3],
        np.exp(-np.abs(data[:, 0] - data[:, 3])),
        data[:, 1] + 3 * data[:, 2]
    ], axis=0))
theta_hat=theta_estimation(X)
print(theta_hat)
# from section 5_3_3 from ISLP website
def boot_SE(func, data, B=3500, seed=0):
    rng = np.random.default_rng(seed)
    first_, second_ = 0, 0
    n = data.shape[0]
    for _ in range(B):
        idx = rng.choice(n, n, replace=True)
        value = func(data[idx])
        first_ += value
        second_ += value**2
    standard_error = np.sqrt(second_ / B - (first_ / B)**2)
    return standard_error
theta_original = theta_estimation(X)

# on veut utiliser le bootstrap avec n repetitions
#voir ISLP_website
repetitions=3500
bootstrap_estimates=np.zeros(repetitions)
np.random.seed(2405979)
for i in range(repetitions):
    bootstrap_sample = X[np.random.choice(range(len(X)), size=len(X), replace=True)]
    bootstrap_estimates[i] = theta_estimation(bootstrap_sample)
bias = np.mean(bootstrap_estimates)-theta_original
std_error = boot_SE(theta_estimation, X, B=3500, seed=matricule)
print("Estimation bootstrap de :", np.mean(bootstrap_estimates))
print("Biais estimé:", bias)
print("Erreur-type estimée:", std_error)
```

```
#found this on internet:
#https://www.statsmodels.org/dev/generated/statsmodels.stats.weightstats.DescrStatsW.html
# tconfint_mean([alpha, alternative]) : two-sided confidence interval for weighted mean of data

output = DescrStatsW(bootstrap_estimates)
int_bas, int_haut = output.tconfint_mean(alpha=0.05, alternative='two-sided')
print("Intervalle de confiance: ", [int_bas, int_haut])
#------Code en R-------#
set.seed(2405979)
mu <- 0
sigma <- 1
sigma_epsilon <- 1
x <- rnorm(n = 100, mean = mu, sd = sigma)
e <- rnorm(n = 100, mean = 0, sd = sigma_epsilon)
beta_0 <- 1
beta_1 <- 2
beta_2 <- -2
beta_3 <- -0.5
y <- beta_0 + beta_1 * x + beta_2 * x^2 + beta_3 * x^3 + e

#voir section  6.5 Lab: Linear Models and Regularization Methods dans ISLr
X_matrix <- poly(x, 10, raw = TRUE)
data <- data.frame(y = y, X_matrix)
#https://www.statology.org/regsubsets-in-r/
best_subset <- regsubsets(y ~ ., data = data, nvmax = 10, method = "exhaustive")
summary_best <- summary(best_subset)
#Trouver les valeurs minimales et plot comme demandé dans ISLr
#pour les valeurs de cp,bic etc on peut les extraire de summary_best
cp_minimal <- which.min(summary_best$cp)
#----cp-----#
#https://ggplot2.tidyverse.org/reference/ggplot.html
# Note you will need to use the data.frame() function to create a single data set containing both X and
cp_minimal <- which.min(summary_best$cp)
data.frame(cp = summary_best$cp, subset_size = 1:10) %>%
  ggplot(aes(x = subset_size, y = cp)) +
  geom_line(color = "purple") +
  geom_point(size = 2, color = "blue") +
  geom_point(aes(x = cp_minimal, y = summary_best$cp[cp_minimal]), color = "red", size = 10, shape = 4)
  labs(x = "Number of Predictors", y = "Cp")

# ---- BIC ----
bic_minimal <- which.min(summary_best$bic)
data.frame(bic = summary_best$bic, subset_size = 1:10) %>%
  ggplot(aes(x = subset_size, y = bic)) +
  geom_line(color = "purple") +
  geom_point(size = 2, color = "blue") +
  geom_point(aes(x = bic_minimal, y = summary_best$bic[bic_minimal]), color = "red", size = 10, shape =
  labs( x = "Number of Predictors", y = "BIC")

# ---- Adjusted R^2 ----
r_carre_ajuste_maximal <- which.max(summary_best$adjr2)
data.frame(adj_r2 = summary_best$adjr2, subset_size = 1:10) %>%
  ggplot(aes(x = subset_size, y = adj_r2)) +
  geom_line(color = "purple") +
  geom_point(size = 2, color = "blue") +
  geom_point(aes(x = r_carre_ajuste_maximal, y = summary_best$adjr2[r_carre_ajuste_maximal]), color = '
```

```
  labs( x = "Number of Predictors", y = "Adjusted R-squared")
cat ("Optimal: \n")
cat("Cp:", cp_minimal, "BIC:", bic_minimal, "Adjusted R^2:", r_carre_ajuste_maximal, "\n")
#ici on va procéeder de la même manière mais en prenant la méthode forward et backward
#forward stepwise selection
model_fwd <- regsubsets(y ~ ., data = data, nvmax = 10, method = "forward")
summary_fwd <- summary(model_fwd)

cp_minimal_fwd <- which.min(summary_fwd$cp)
data.frame(cp = summary_fwd$cp, subset_size = 1:10) %>%
  ggplot(aes(x = subset_size, y = cp)) +
  geom_line(color = "purple") +
  geom_point(size = 2, color = "blue") +
  geom_point(aes(x = cp_minimal_fwd, y = summary_fwd$cp[cp_minimal_fwd]), color = "red", size = 10, sha
  labs(x = "Number of Predictors", y = "Cp")
bic_minimal_fwd <- which.min(summary_fwd$bic)
data.frame(bic = summary_fwd$bic, subset_size = 1:10) %>%
  ggplot(aes(x = subset_size, y = bic)) +
  geom_line(color = "purple") +
  geom_point(size = 2, color = "blue") +
  geom_point(aes(x = bic_minimal_fwd, y = summary_fwd$bic[bic_minimal_fwd]), color = "red", size = 10,
  labs(x = "Number of Predictors", y = "BIC")
r_carre_ajuste_maximal_fwd <- which.max(summary_fwd$adjr2)
data.frame(adj_r2 = summary_fwd$adjr2, subset_size = 1:10) %>%
  ggplot(aes(x = subset_size, y = adj_r2)) +
  geom_line(color = "purple") +
  geom_point(size = 2, color = "blue") +
  geom_point(aes(x = r_carre_ajuste_maximal_fwd, y = summary_fwd$adjr2[r_carre_ajuste_maximal_fwd]), co
  labs(x = "Number of Predictors", y = "Adjusted R-squared")

cat ("forward: \n")
cat("Cp:", cp_minimal_fwd, "BIC:", bic_minimal_fwd, "Adjusted R^2:", r_carre_ajuste_maximal_fwd, "\n")

#backward stepwise selection
model_bwd <- regsubsets(y ~ ., data = data, nvmax = 10, method = "backward")
summary_bwd <- summary(model_bwd)
cp_minimal_bwd <- which.min(summary_bwd$cp)
bic_minimal_bwd <- which.min(summary_bwd$bic)
r_carre_ajuste_maximal_bwd <- which.max(summary_bwd$adjr2)
data.frame(cp = summary_bwd$cp, subset_size = 1:10) %>%
  ggplot(aes(x = subset_size, y = cp)) +
  geom_line(color = "purple") +
  geom_point(size = 2, color = "blue") +
  geom_point(aes(x = cp_minimal_bwd, y = summary_bwd$cp[cp_minimal_bwd]), color = "red", size = 10, sha
  labs(x = "Number of Predictors", y = "Cp")

data.frame(bic = summary_bwd$bic, subset_size = 1:10) %>%
  ggplot(aes(x = subset_size, y = bic)) +
  geom_line(color = "purple") +
  geom_point(size = 2, color = "blue") +
  geom_point(aes(x = bic_minimal_bwd, y = summary_bwd$bic[bic_minimal_bwd]), color = "red", size = 4, s
  labs(x = "Number of Predictors", y = "BIC")
data.frame(adj_r2 = summary_bwd$adjr2, subset_size = 1:10) %>%
  ggplot(aes(x = subset_size, y = adj_r2)) +
  geom_line(color = "purple") +
  geom_point(size = 2, color = "blue") +
```

```
  geom_point(aes(x = r_carre_ajuste_maximal_bwd, y = summary_bwd$adjr2[r_carre_ajuste_maximal_bwd]), co
  labs(x = "Number of Predictors", y = "Adjusted R-squared")
cat ("backward: \n")
cat("Cp:", cp_minimal_bwd, "BIC:", bic_minimal_bwd, "Adjusted R^2:", r_carre_ajuste_maximal_bwd, "\n")

print(coef(best_subset, id = cp_minimal))
print(coef(best_subset, id = bic_minimal))
print(coef(best_subset, id = r_carre_ajuste_maximal))
print(coef(model_fwd, id = cp_minimal_fwd))
print(coef(model_fwd, id = bic_minimal_fwd))
print(coef(model_fwd, id = r_carre_ajuste_maximal_fwd))
print(coef(model_bwd, id = cp_minimal_bwd))
print(coef(model_bwd, id = bic_minimal_bwd))
print(coef(model_bwd, id = r_carre_ajuste_maximal_bwd))
#voir apd de page 282 sur ISLR
#We first set a random seed so that the results obtained will be reproducible. -> juste moi matricule
set.seed(2405979)
#ridge.mod <-glmnet(x, y, alpha = 0, lambda = grid)
#lasso.mod <-glmnet(x[train, ], y[train], alpha = 1,lambda = grid)

#fit a lasso model to the simulated data, Use cross-validation to select the optimal value of
#Create plots of the cross-validation error as a function of . Report the resulting coefficient estimat
#as for question 1 I will use 5-Kfold
model_lasso <- cv.glmnet(y = y, x = X_matrix, alpha = 1, lambda = 10^seq(1, -2, length = 100),  standar
plot_data <- data.frame(lambda = model_lasso$lambda, cv_mse = model_lasso$cvm, nonzero_coeff = model_la
#https://www.statology.org/lasso-regression-in-r/
#on va refaire comme les autres questions avec mse et lambda
ggplot(plot_data, aes(x = lambda, y = cv_mse)) +
  geom_line(color = "purple") +
  geom_point(size = 2, color = "blue") +
  geom_point(aes(x = model_lasso$lambda.min, y = min(model_lasso$cvm)), color = "red", size = 10, shape
  scale_x_continuous(trans = 'log10')+
  scale_y_continuous(trans = 'log10')+
  labs(x = "Lambda", y = "Cross-Validation MSE")
cat("Lambda optimal: \n")
cat(model_lasso$lambda.min)
meilleur_lasso <- glmnet(y = y, x = X_matrix, alpha = 1)
coefficient_pour_lasso <- predict(meilleur_lasso, s=model_lasso$lambda.min, type = "coefficients")
print(coefficient_pour_lasso)
#Now generate a response vector Y according to the model Y = b0 + b7*x^7 +e
set.seed(2405979)
beta_0 <- 10
beta_7 <- 5
y <- beta_0 + beta_7*x^7 + e
#on refait tout pour ce modèle
#perform best subset selection and the lasso. Discuss the results obtained.
X_matrix <- poly(x, 10, raw = TRUE)
model_best <- regsubsets(y ~ ., data = X_matrix, nvmax = 10, method = "exhaustive")
model_summary <- summary(model_best)

#on extrait à nv -> copié collé du premier code
cp_minimal <- which.min(summary_best$cp)
#----cp-----#
#https://ggplot2.tidyverse.org/reference/ggplot.html
# Note you will need to use the data.frame() function to create a single data set containing both X and
cp_minimal <- which.min(summary_best$cp)
```

```
data.frame(cp = summary_best$cp, subset_size = 1:10) %>%
  ggplot(aes(x = subset_size, y = cp)) +
  geom_line(color = "purple") +
  geom_point(size = 2, color = "blue") +
  geom_point(aes(x = cp_minimal, y = summary_best$cp[cp_minimal]), color = "red", size = 10, shape = 4)
  labs( x = "Number of Predictors", y = "Cp")

# ---- BIC ----
bic_minimal <- which.min(summary_best$bic)
data.frame(bic = summary_best$bic, subset_size = 1:10) %>%
  ggplot(aes(x = subset_size, y = bic)) +
  geom_line(color = "purple") +
  geom_point(size = 2, color = "blue") +
  geom_point(aes(x = bic_minimal, y = summary_best$bic[bic_minimal]), color = "red", size = 10, shape =
  labs( x = "Number of Predictors", y = "BIC")

# ---- Adjusted R^2 ----
r_carre_ajuste_maximal <- which.max(summary_best$adjr2)
data.frame(adj_r2 = summary_best$adjr2, subset_size = 1:10) %>%
  ggplot(aes(x = subset_size, y = adj_r2)) +
  geom_line(color = "purple") +
  geom_point(size = 2, color = "blue") +
  geom_point(aes(x = r_carre_ajuste_maximal, y = summary_best$adjr2[r_carre_ajuste_maximal]), color = '
  labs( x = "Number of Predictors", y = "Adjusted R-squared")
cat ("Optimal: \n")
cat("Cp:", cp_minimal, "BIC:", bic_minimal, "Adjusted R^2:", r_carre_ajuste_maximal, "\n")



model_lasso <- cv.glmnet(y = y, x = X_matrix, alpha = 1, lambda = 10^seq(1, -2, length = 100), standard
plot_data <- data.frame(lambda = model_lasso$lambda, cv_mse = model_lasso$cvm, nonzero_coeff = model_la
#https://www.statology.org/lasso-regression-in-r/
#on va refaire comme les autres questions avec mse et lambda
ggplot(plot_data, aes(x = lambda, y = cv_mse)) +
  geom_line(color = "purple") +
  geom_point(size = 2, color = "blue") +
  geom_point(aes(x = model_lasso$lambda.min, y = min(model_lasso$cvm)), color = "red", size = 10, shape
  scale_x_continuous(trans = 'log10')+
  scale_y_continuous(trans = 'log10')+
  labs(x = "Lambda", y = "Cross-Validation MSE")
  cat("Lambda optimal: \n")
cat(model_lasso$lambda.min)
meilleur_lasso <- glmnet(y = y, x = X_matrix, alpha = 1)
coefficient_pour_lasso <- predict(meilleur_lasso, s=model_lasso$lambda.min, type = "coefficients")
print(coefficient_pour_lasso)
```