SY19 - Facial Recognition

Jo COLINA & Paul GOUJON

December 2016

Chapitre 1

Introduction

Introduction Ce document rend compte de notre travail au cours du dernier TP de SY19, UV de Machine Learning de l'Université de Technologie de Compiègne. Au cours de ce TP, il nous est demandé de mettre en place l'algorithme de Machine Learning le plus performant possible pour une tâche de facial recognition, à l'aide des méthodes que nous avons apprises en cours (ou en lisant des livres tels que l'Introduction to Statistical Learning).

Objectifs du TP Les objectifs du TP sont multiples, parmi lesquels :

- Produire une analyse pertinente du dataset fourni
- Proposer une diversité de modèle
- Analyser de manière pertinente les résultats obtenus
- Tirer des conclusions les plus justes possible concernant les performances des modèles testés

Code Notre code source est fourni en annexe de ce rapport, ou disponible à l'URL `https://github.com/goujonpa/SY19TP7`.
Ce dernier s'organise comme suit (NB : les noms de fichiers sont des hyperliens, clickables) :

- SY19.R : Script principal, à partir duquel les données sont chargées, et toutes nos fonctions d'analyse lancées.
- pc.R : contient notre PCA
- fa.R : contient notre FDA
- ldaqda.R : contient nos analyses via LDA et QDA
- svm.R : contient nos analyses via SVM
- randomf.R : contient nos analyses via Random Forest
- nn.R : contient nos analyses via Neural Networks
- dossier csv : contient les worksheets de résultats exportés
- dossier plots : contient les plots de résultats exportés
- + d'autres scripts annexes que nous ne jugeons pas utile de détailler

Chapitre 2

Analyses préliminaire des données

Introduction Nous sommes en présence d'une matrice de taille 216 x 4200 d'individus et d'un vecteur de taille 216 d'étiquettes. Chaque ligne de la matrice est en réalité une photo de taille 60 x 70 pixels en niveaux de gris d'un visage exprimmant une émotion parmis les six suivantes :

- joie
- surprise
- tristesse
- dégout
- colère
- peur

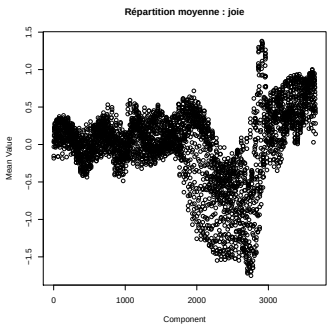
2.1 Analyse initiale

Dimensions La première chose qui nous frappe lorsque l'on est confronté à ce dataset est le ratio $p >> n$: nous sommes en présence de beaucoup plus de prédicteurs que d'individus, ce qui nous laisse penser que nous serons amenés à utiliser des méthodes de réduction de dimensions afin de contrer la fameuse "curse of dimensionality".

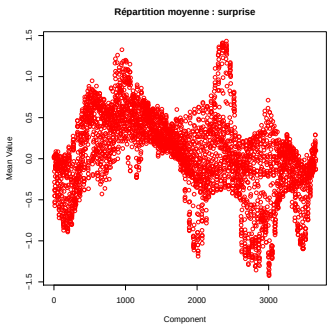
Pré-traitement En affichant les différentes images, on se rend également compte qu'une partie de chacune des images est totalement inutile dans le cadre de nos analyses, voire risquerait de la bruiter : il s'agit de tous les pixels noir en bas de chaque image. Nous décidons de les exclure de toute future analyse.

2.2 Répartition moyenne par expression

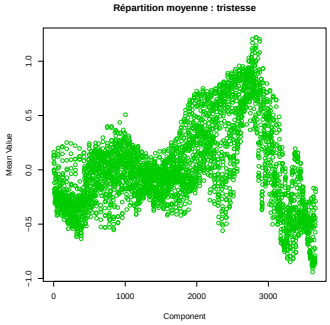
Répartition moyenne & variance Ensuite, afin de voir si les différentes expressions produisent des "vecteur-image moyens" significativement différents, nous décidons de ploter la réparition moyenne des prédicteurs pour chaque expression. Nous obtenons les courbes suivantes.



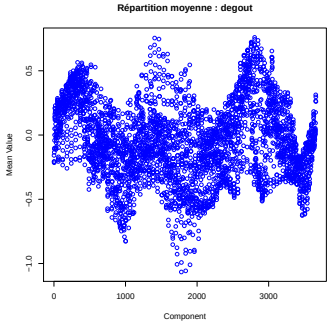
(a) Joie



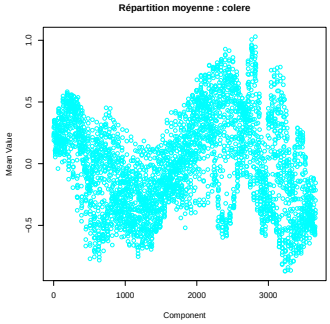
(b) Surprise



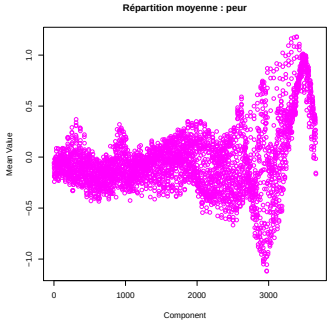
(c) Tristesse



(d) Degoût



(e) Colère



(f) Peur

FIGURE 2.1 – Répartition moyenne des prédicteurs

NB : Standardized La matrice des individus est centrée et réduite une fois effectué le pré-traitement décrit précédemment. C'est le cas pour les représentations ci-dessus.

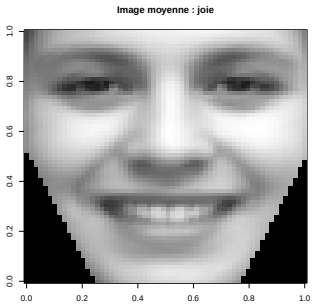
Interprétation Les répartitions obtenue semblent différer significativement selon l'expression. Nous pouvons donc envisager que l'usage de modèles de prédiction nous permettrait de distinguer les différentes expressions.

Variance CHECKER LA VARIANCE TOUT DE MEME CAR CA POURRAIT ETRE TOUT SIMPLEMENT DU A CA

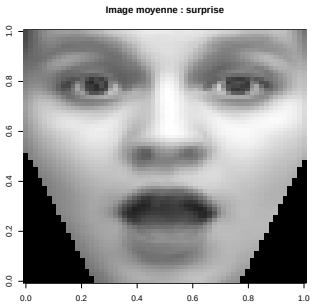
Corrélation LA HEATMAP CI-DESSOUS BLABLA CORRELATION ENTRE LES DIFFERENTS PREDICTEURS

2.3 Expression moyenne

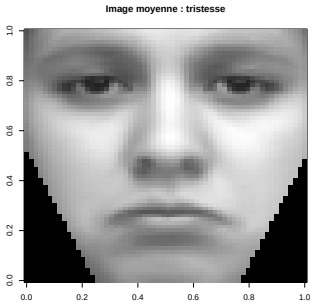
Curieux Nous avons tenté d'effectuer la moyenne pour chaque expression des images : c'est à dire produire six images issues de la moyenne mathématique des images de l'expression considérée. Ci-dessous sont les "visages moyens" issus de cette moyenne. Nous avons trouvé extrêmement curieux l'aspect humain que prend une "moyenne de visages".



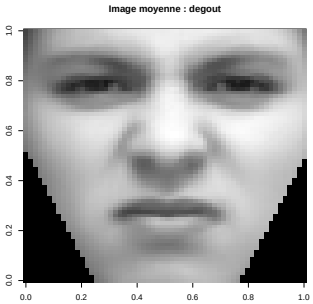
(a) Joie



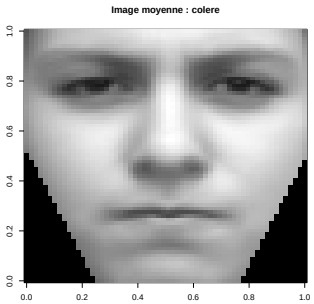
(b) Surprise



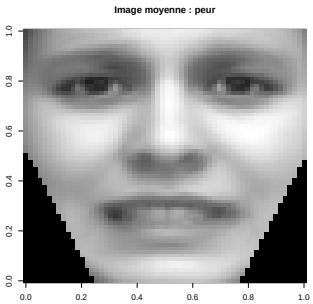
(c) Tristesse



(d) Degoût



(e) Colère



(f) Peur

FIGURE 2.2 – Répartition moyenne des prédicteurs

2.4 Première analyse : conclusion et démarche envisagée

Conclusion Nous sommes en présence d'un dataset présentant un fort nombre de prédicteurs comparé au nombre d'individus. Les différents prédicteurs sont globalement très corrélés. Nous allons donc devoir utiliser des méthodes de réduction de dimension sur ce dataset, puis par la suite tester nos différents modèles.

Démarche Nous envisageons ainsi d'utiliser plusieurs méthodes de réduction de dimension :

- Principal Component Analysis
- Factor Analysis
- Forward Stepwise Selection
- ... + combinaisons de plusieurs d'entre elles

avant d'entrainer différents modèles sur les jeux de données résultant, et les optimiser. Nous comparerons les performances de ces différents modèles :

- Linear Discriminant
- Quadratic Discriminant Analysis
- Support Vector Machines
- Random Forest
- Neural Networks

Chapitre 3

Principal Component Analysis

3.1 Principe

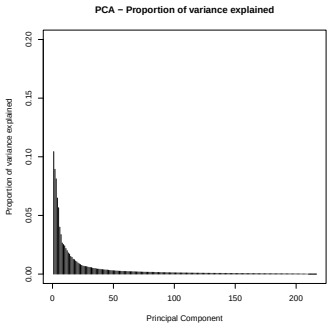
Principe L'analyse en composantes principales consiste à transformer des variables liées entre elles (dites « corrélées ») en nouvelles variables décorrélées les unes des autres. Ces nouvelles variables sont nommées « composantes principales », ou axes principaux. Elle permet de réduire le nombre de variables et de rendre l'information moins redondante.

Motivation Comme expliqué précédemment, nous sommes en présence d'un dataset comprenant un nombre restreint d'individus, chacun présentant un nombre élevé de prédicteurs, relativement corrélés. Il nous parait donc nécessaire d'utiliser un méthode de réduction de dimensions et de corrélation avant toute analyse. La PCA nous semble être l'une des méthodes s'imposant.

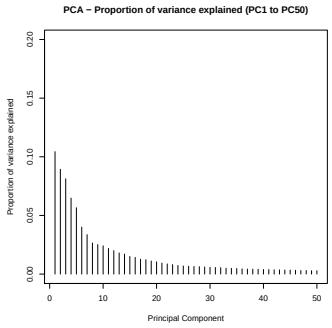
3.2 PCA - Réalisation

R Nous réalisons la PCA à l'aide de la fonction `prcomp` de R. Cette fonction effectue l'analyse en composantes principales de notre matrice d'individus de manière automatique et nous fourni ainsi la matrice de composantes principales, leurs variances, ainsi que la matrice de rotation.

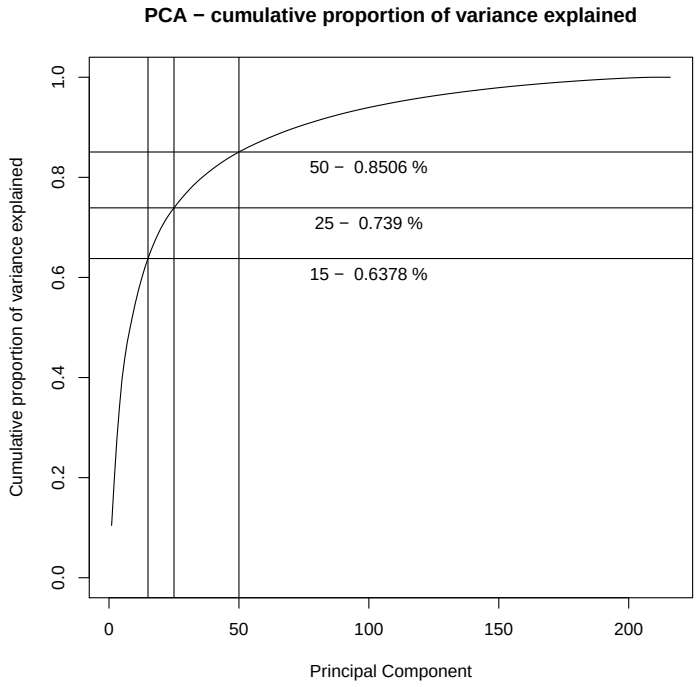
Proportion de variance expliquée Afin de ne retenir qu'un certain nombre de composantes principales, nous nous intéressons à la proportion de variance expliquée par chacune d'elles en plottant leur proportion de variance expliquée brute, et cumulée. Les courbes obtenues sont les suivantes.



(a) PCA : Proportions de variance expliquée



(b) PCA : Proportions de variance expliquée (50 premières PC)



(c) PCA : Pourcentage de variance expliquée cumulée

FIGURE 3.1 – Résultats de PCA

Interprétation Afin de tester les performances de nos modèles sur différentes sélections de Principal Component (PC), nous décidons de composer plusieurs dataset :

- **PC5** : A partir des 5 premières PC
- **PC15** : A partir des 15 premières PC, représentant environ 65% de variance expliquée
- **PC25** : A partir des 25 premières PC, représentant environ 75% de variance expliquée
- **PC50** : A partir des 50 premières PC, représentant environ 85% de variance expliquée
- **PC100** : A partir des 100 premières PC
- **PC200** : A partir des 200 premières PC

NB : Si nous disposons d'assez de temps, nous tenterons peut être une sélection de composantes principales par Forward Stepwise Selection.

Deux premières PC Nous tentons également de plotter nos deux premières principal component, dans l'espoir de constater un quelconque pattern intéressant.

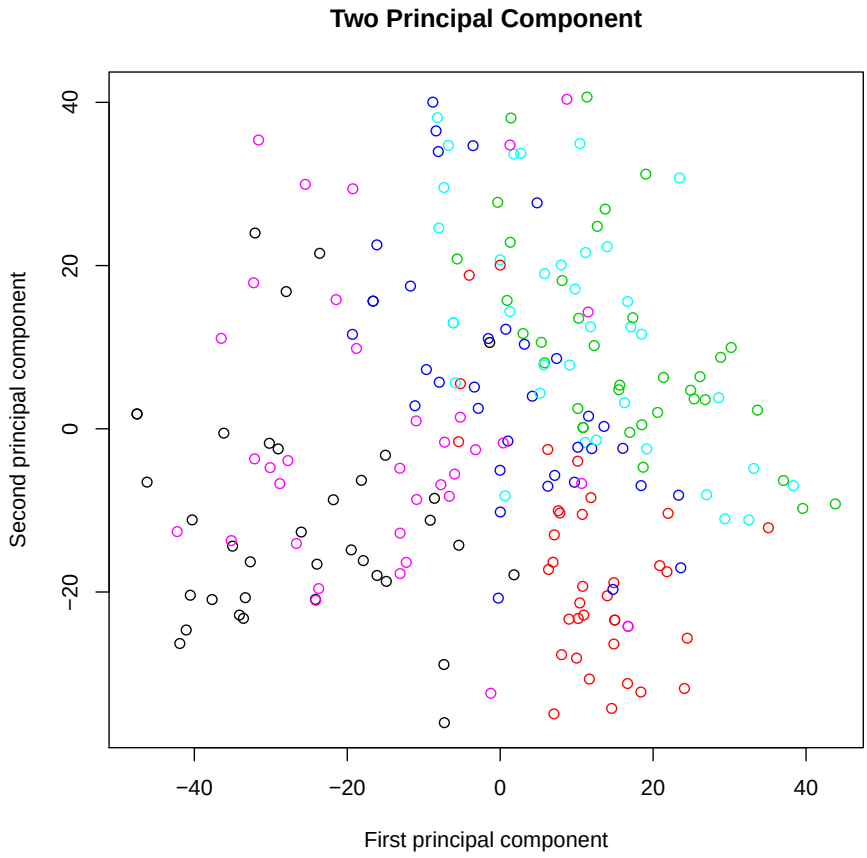


FIGURE 3.2 – Représentation des deux premières PC dans le premier plan Factoriel

Interprétation Nous remarquons une relativement bonne répartition "par classe" des points : les vert en haut à droite, les turquoise également, les bleus de préférence vers le centre, les rouges centrés en bas, les violets et noirs à gauche, avec les noirs de préférence vers le bas. Nous ne nous attendions pas à ce que les individus aient autant tendance à être rassemblés par classe (bien que cela soit loin d'être parfait).
Cela nous laisse de nouveau penser que nous pouvons espérer des performances intéressantes de la part de nos modèles de machine learning.

Chapitre 4

Factor Discriminant Analysis

4.1 Principe & motivation

Principe La FDA est un technique de réduction de dimension supervisée, adaptée pour les problèmes de classification. Elle trouve des combinaisons linéaires des prédicteurs originaux, de manière à maximiser la variance inter-classe en minimisant la variance intra-classe (c'est à dire de telle manière que les classes se "chevauchent" le moins possible).

Motivation Pour les mêmes raisons que celles décrites précédemment, nous cherchons à réduire le nombre de prédicteurs décrivant nos individus, tout en essayant de construire des prédicteurs nous permettant de les classifier le mieux possible. La FDA est l'une des méthodes nous permettant de nous approcher de ce but, c'est ce qui nous a poussé à l'utiliser.

4.2 FDA - Réalisation

R A l'image de ce qui a été vu en cours, nous utilisons la méthode `lda` du langage `R`, et multiplions la matrice X des individus par la matrice *lda.scaling* des eigenvectors retenus.

4.3 FDA - Plusieurs approches

Introduction La curiosité nous a poussé à tester la FDA selon différentes approches.

4.3.1 FDA sur dataset initial entier

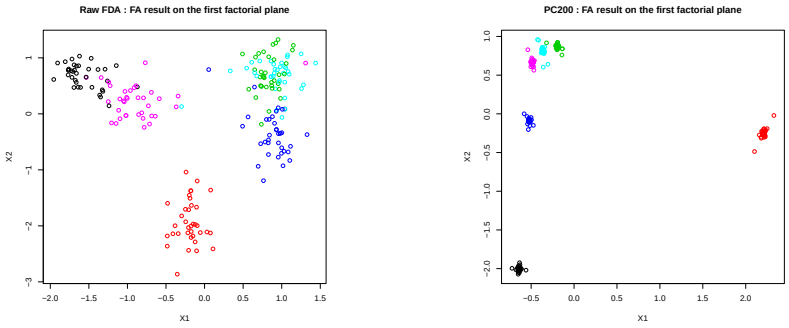
Description de l'approche Nous avons tout d'abord tenté d'utiliser tous les individus du dataset initial pour la détermination du scaling à appliquer à nos individus. Nous avons donc appliqué la méthode décrite dans le cours, obtenu la matrice d'eigenvectors retenus par laquelle nous avons multiplié celle de nos individus.

Résultats Nous avons donc appliqué la FDA deux datasets :

- Raw FDA : dataset contenant l'ensemble de tous les individus du dataset initial
- PC200 FDA : dataset contenant les 200 PC de tous les individus du dataset initial

Et avons ensuite fit un modèle LDA et un modèle QDA sur les individus ainsi transformés.

Résultats erronés, Overfitting Nous nous sommes rapidement rendus compte que les résultats ainsi obtenus paraissaient douteux : nos taux d'erreurs étaient très bas, et lorsque nous avons représenté les individus post-FDA dans le premier plan factoriel, les classes nous paraissaient "un peu trop bien séparées". Ci-dessous, sont représentés les individus des deux datasets sus-cités dans le premier plan factoriel. Le résultat de la FDA sur nos PC 200 nous a clairement fait comprendre que nous n'utilisions pas la bonne approche concernant la FDA : l'overfitting crève les yeux.



(a) Raw FDA

(b) PC 200 FDA

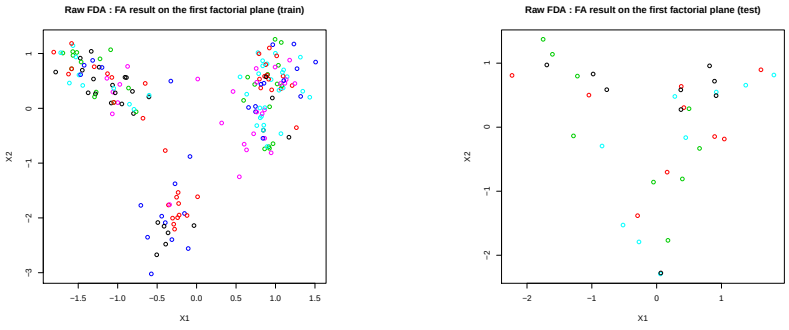
FIGURE 4.1 – Représentation des résultats de FDA dans le premier plan factoriel

Conclusion - Nécessité de la CV Nous nous sommes rapidement rappelés que **la FDA est une méthode supervisée**, il est donc évident que nous avions affaire ici à des taux d'erreur de training et non de test, et qu'il nous fallait intégrer la FDA à notre cross-validation. Ce qui nous amène à l'approche que nous avons adoptée à partir de ce moment.

4.3.2 6-folds CV sur FDA

Description de l'approche C'est en constatant les "un peu trop excellents" résultats de l'approche précédente que nous décidons d'effectuer la même manipulation en utilisant la 6-folds Cross Validation sur la FDA. De cette manière, nous obtenons une estimation robuste du taux d'erreur de test de la FDA.
Note : Nous reviendrons plus en détails sur la méthode de k-folds Cross Validation au sein du chapitre 5.

Résultats A partir du moment ou nous avons utilisé cette méthode de 6-folds CV, nous avons constaté que nos résultats étaient bien plus cohérents, et cela était confirmé par les représentations des résultats ainsi obtenus dans le premier plan factoriel. Ci-dessous sont représentés les individus de train et de test d'une itération de 6-folds CV après FDA, issus du dataset initial (et n'ayant pas subis de PCA).



(a) Raw FDA

(b) PC 200 FDA

FIGURE 4.2 – Représentation des résultats de FDA dans le premier plan factoriel en utilisant la 6-folds CV

Interprétation Cette fois-ci, fini l'overfitting. Nous observons des taux d'erreur correspondant réellement à des estimations de l'erreur de test.

4.3.3 Grain de folie : FDA sur résultat de PCA

Description de l'approche Ayant résolu notre problème d'overfitting, nous n'avons pas perdu notre idée/curiosité initiale de tester la FDA sur un résultat de PCA. Nous décidons donc de comparer les résultats via 6-folds CV de nos modèles LDA et QDA, entrainés sur les résultats de FDA appliquée sur différentes données :

- Raw FDA : individus du dataset initial non transformés
- PC5 à PC200 FDA : 5PC à 200PC issues de la PCA appliquée aux individus du dataset initial

Représentation La figure ci-dessous représente les taux d'erreurs de test ainsi obtenus.

TABLE 4.1 – Tableau de correspondance

Dataset	LDA	QDA
Raw FDA	15	16
PC5 FDA	17	18
PC15 FDA	19	20
PC25 FDA	21	22
PC50 FDA	23	24
PC100 FDA	25	26
PC200 FDA	27	28

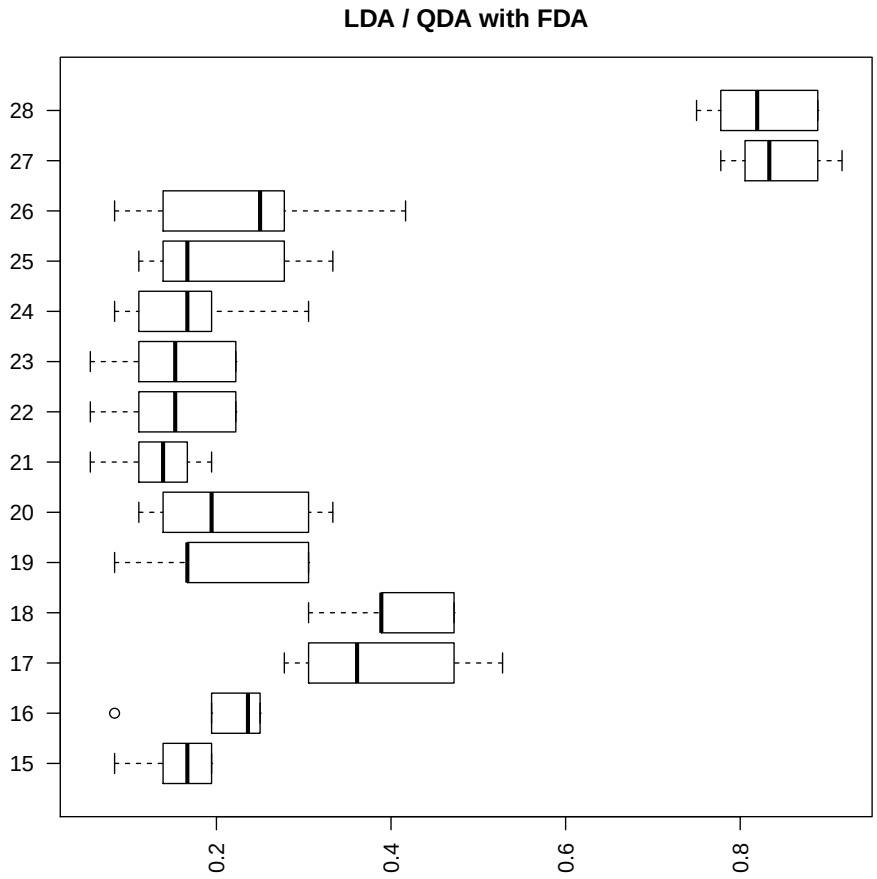


FIGURE 4.3 – Représentation des taux d'erreur de test de LDA et QDA après application de FDA sur les datasets décrits précédemment

Interprétation & références Nous remarquons qu'en appliquant la FDA aux datasets PC15, PC25 et PC50, nous améliorons les performances de nos classifieurs, comparé à a FDA sur le dataset Raw. En revanche, PC100, PC5 et PC200 ne les améliorent pas, voire les dégradent.
Les figures ci-dessus et la consultation de différentes références, notamment http://stats.stackexchange.com/questions/106121/does-it-make-sense-to-combine-pca-and-lda semble suggérer que l'utilisation de la PCA avant application de la FDA peut améliorer les effets de cette dernière, cependant, le choix du nombre de PC retenues est important : un trop grand nombre de PC risque de reproduire notre problème d'overfitting précédent, tandis qu'un trop petit nombre d'entre elles ne permet pas une bonne représentativité du dataset initial, étant donné qu'elles contiennent une trop faible proportion de variance expliquée dans ce cas.

4.4 FDA - Conclusion & utilisation dans la suite du TP

Conclusion Il semblerait donc que la FDA nous permette d'obtenir des résultats différents et parfois meilleurs que la PCA. Nous continuerons donc de l'utiliser au sein des différents modèles de notre TP et de présenter les résultats ainsi obtenus.

Chapitre 5

Protocoles experimentaux

Notations Dans la suite du TP, nous présenterons nos résultats en suivant les conventions de naming suivantes :

- **Raw** : utilisation du dataset initial sans transformation
- **PCxxx** : utilisation des xxx Pr. Comp. issues de la PCA sur le dataset initial
- **FDA** : application de la FDA au sein d'une 6 folds CV
- **FullDS** : utilisation de la FDA sur tous les individus du dataset initial (scientifiquement non pertinent)

Protocoles experimentaux - général Globalement, nous allons pour la suite de ce TP tester différents modèles et en évaluer les performances. Pour cela nous adoptons une démarches en trois étapes pour chacun des classifieurs :

1. **Analyse préliminaire** : nous effectuons un premier balayage, en testant un large panel de paramètres (choisis généralement du même ordre de grandeur que les paramètres par défaut choisis par les différents packages `R`) afin de visualiser les performances globales en fonction de ces derniers de nos modèles
2. **Optimisation** : nous optimisons par la suite les paramètres un à un, en les fixant tour à tour puis faisant varier les paramètres non encore fixés
3. **Evaluation du taux d'erreur et matrices de confusions** : pour finir, nous estimons le taux d'erreur optimum obtenu pour le modèle sur le jeu de données considéré, et construisons sa matrice de confusion.

Estimation du taux d'erreur Selon le dataset utilisé, nous utilisons différentes méthodes d'estimation du taux d'erreur :

- **6-folds Cross Validation** : est la méthode que d'estimation d'erreur que nous avons utilisée pour toutes les fonctions que nous avons écrites nous même (ne faisant pas partie de packages...). Pourquoi 6 ? Car 216 divisé par 6 donne un chiffre rond : 36.

— **10-folds CV** : la 10-folds CV est utilisée par la fonction `tune` lors du "tuning" des paramètres

k-folds Cross Validation - Principe La k-folds cross validation consiste à diviser l'échantillon original en k échantillons, puis sélectionner un des k échantillons comme ensemble de validation et les (k-1) autres échantillons qui constitueront l'ensemble d'apprentissage. On calcule le taux d'erreur de classification du modèle entrainé selon cette répartition, puis on répète l'opération en sélectionnant un autre échantillon de validation parmi les (k-1) échantillons qui n'ont pas encore été utilisés pour la validation du modèle. L'opération se répète ainsi k fois pour qu'en fin de compte chaque sous-échantillon ait été utilisé exactement une fois comme ensemble de test. La moyenne des k taux d'erreur de test est enfin calculée pour estimer l'erreur de test finale.

Concision du compte rendu Nous avons chronologiquement commencé par éprouver tous les modèles sur le dataset "PC15" (donc testé, débugué, optimisé), avant de réutiliser nos fonctions d'analyse sur les autres dataset & méthodes (ex : FDA).
Dans un soucis de concision (vous ne voulez surement pas un rapport de 200 pages), nous ne montrerons dans la suite de ce rapport que les figures obtenues via l'analyse de ce dataset, et récapitulerons les résultats obtenus avec d'autres datasets & méthodes dans un tableau de performances en fin de chaque section. **L'ensemble de nos résultats sont néanmoins disponibles au sein des dossiers "plots" et "csv" fournis en annexe.**

Chapitre 6

Linear Discriminant Analysis, Quadratic Discriminant Analysis

Introduction Les deux premier modèles que nous décidons d'éprouver sont la LDA et la QDA (que nous maitrisons le mieux étant donné qu'il s'agit des premiers étudiés en SY19 de ce TP).

6.1 LDA & QDA : Protocol experimental

6-folds CV Nous souhaitons estimer le taux d'erreur de test de ces deux modèles. Pour cela nous allons utiliser la méthode de resampling dite "6-folds cross validation" (Pourquoi 6 plutôt que 5 ? Parce que 216/6 est un chiffre rond : 36).

Apprentissage / Prédiction Nous effectuons donc pour chaque modèle 6 apprentissages sur les 5 folds de train, suivis de leur prédiction sur le fold de test.

Estimation de l'erreur de test Nous estimons par la suite l'erreur de test en calculant l'erreur de cross validation.

Confusion matrix Nous construisons par ailleurs la matrice de confusion issue de la prédiction.

6.2 LDA & QDA : Résultats

Test error rate La boxplot ci-dessous présente le taux d'erreur des deux modèles pour le jeu de données "PC15".

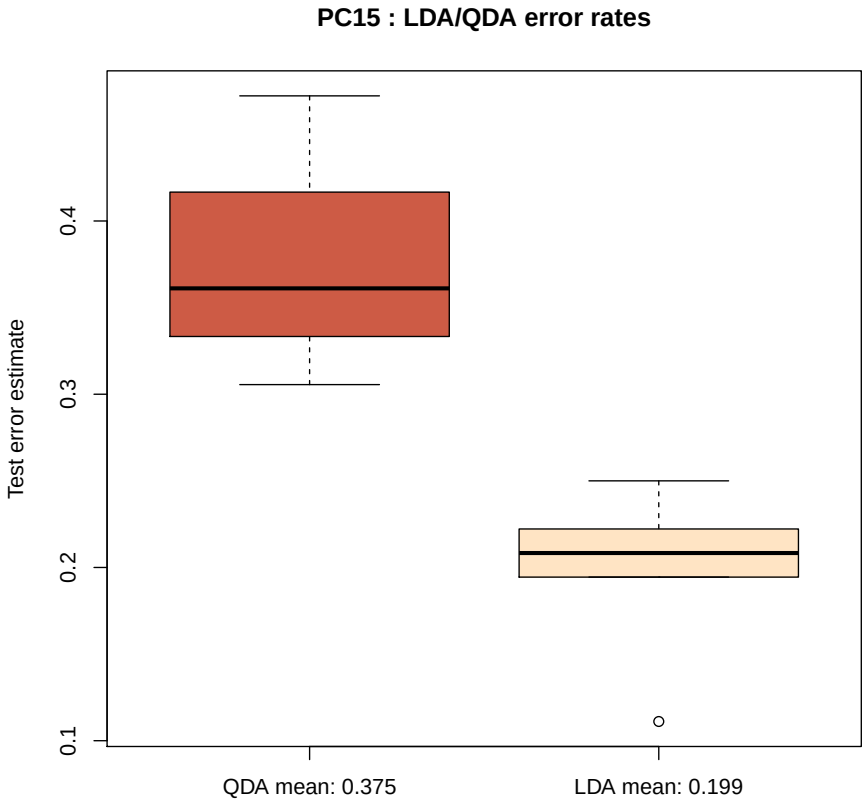


FIGURE 6.1 – LDA & QDA : Taux d'erreurs de test

Matrice de confusion Les matrice de confusion ci-dessous peuvent être obtenues par LDA et QDA.

TABLE 6.1 – LDA : Matrice de confusion

Actual \ Predicted	Joie	Surprise	Tristesse	Dégoût	Colère	Peur	Total
Joie	31	0	0	1	0	4	36
Surprise	0	34	0	0	2	0	36
Tristesse	0	0	29	2	4	1	36
Dégoût	0	0	1	29	4	2	36
Colère	0	0	3	6	27	0	36
Peur	7	0	2	0	1	26	36
Total	38	34	35	38	38	33	216

TABLE 6.2 – QDA : Matrice de confusion

Actual \ Predicted	Joie	Surprise	Tristesse	Dégoût	Colère	Peur	Total
Joie	21	0	0	1	0	14	36
Surprise	0	32	1	0	0	3	36
Tristesse	1	0	22	3	6	4	36
Dégoût	2	0	1	21	5	7	36
Colère	0	0	4	12	19	1	36
Peur	8	0	1	3	1	23	36
Total	32	32	29	40	31	52	216

6.3 LDA & QDA : Interprétations

Observations La LDA semble délivrer de meilleures performances que la QDA. La "joie" a tendance a être étiquetée "peur". La "tristesse" a tendance a être confondue avec "dégoût", "colère", ou "peur". Le "dégoût" est quant à lui confondu avec "colère" ou "peur". La colère est très souvent classifiée "dégoût". Pour finir, la "peur" est régulièrement étiquetée "joie".

Interprétation Les classes concernées par ces confusions semblent correspondre aux classes que nous voyons superposées sur la représentation des deux principales composantes analysée précédemment.
Nous pouvons émettre l'hypothèse que certaines expressions sont difficilement différentiables les unes des autres car relativement superposées sur les composantes principales choisies.

LDA vs QDA il semblerait que la LDA nous permette d'obtenir de meilleures performances que la QDA.

Il semblerait donc que ce modèle, moins flexible, propose de meilleures frontières de décision, réduisant la variance de notre modèle (au prix, certainement, d'un certain biais).

6.4 Autres datasets - Résultats

Résultats Les figures ci-dessous présentent les résultats que nous obtenons pour les modèles de FDA et QDA

TABLE 6.3 – Tableau de correspondances

Dataset	LDA	QDA
Raw	1	2
PC15	3	4
PC25	5	6
Raw FDA FullDS	7	8
PC5 FDA FullDS	9	10
PC25 FDA FullDS	11	12
PC200 FDA FullDS	13	14
Raw FDA	15	16
PC5	17	18
PC15	19	20
PC25	21	22
PC50	23	24
PC100	25	26
PC200	27	28

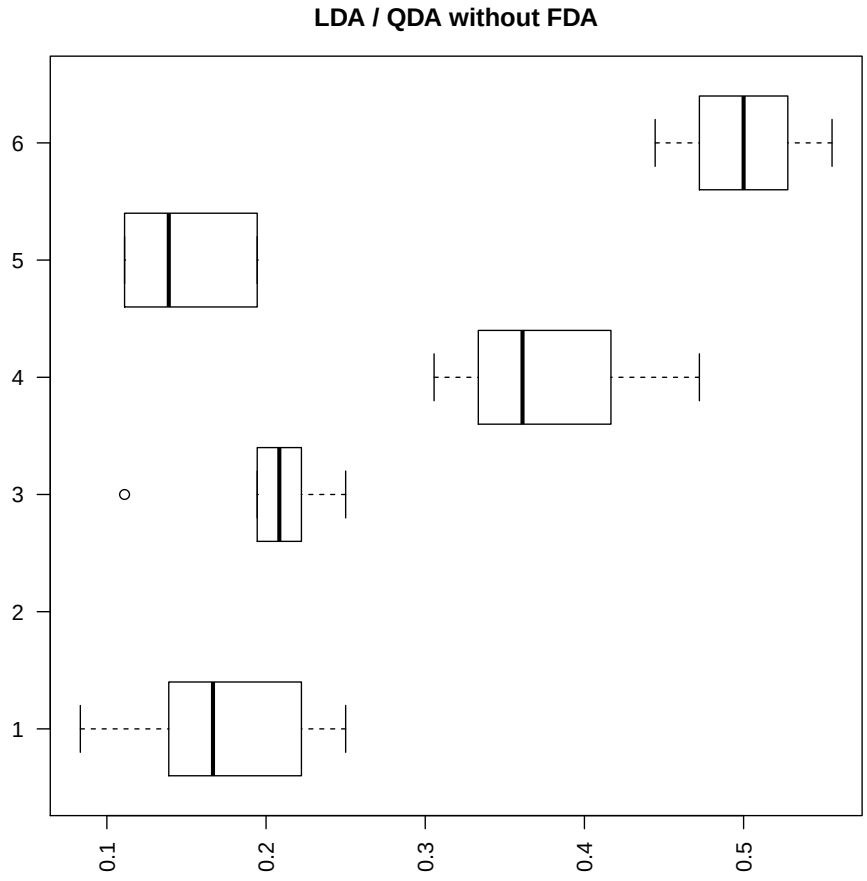


FIGURE 6.2 – LDA & QDA : Taux d'erreurs de test sans application de FDA

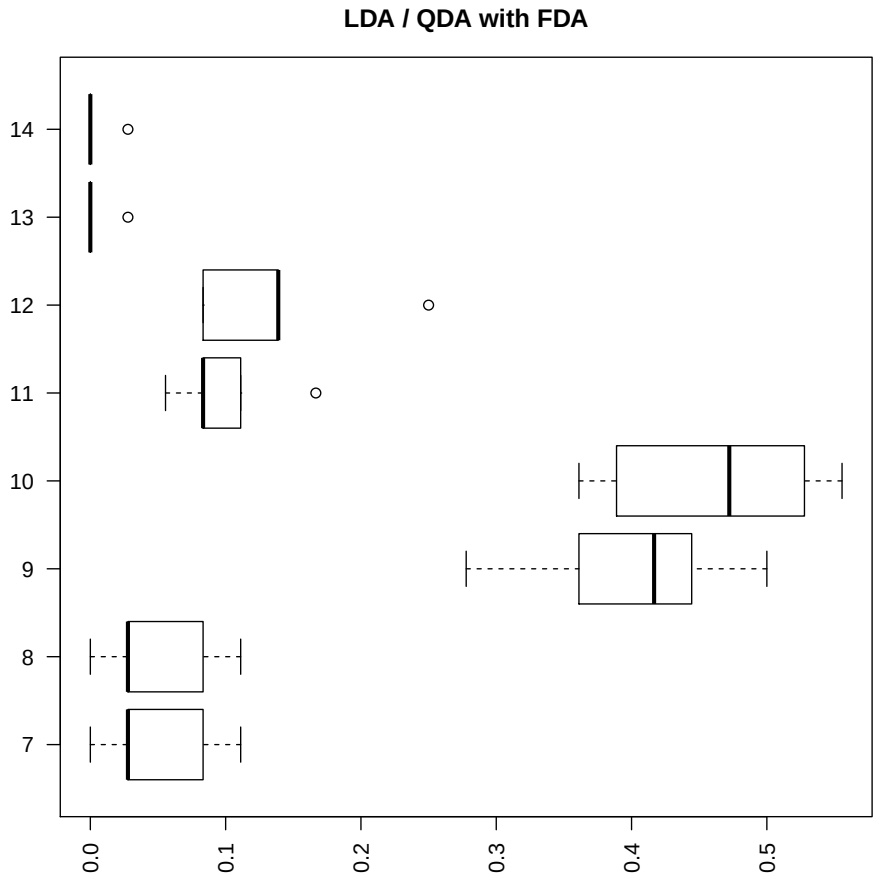


FIGURE 6.3 – LDA & QDA : Taux d'erreurs de train avec FDA sur tous les individus initiaux (scientifiquement peu pertinent)

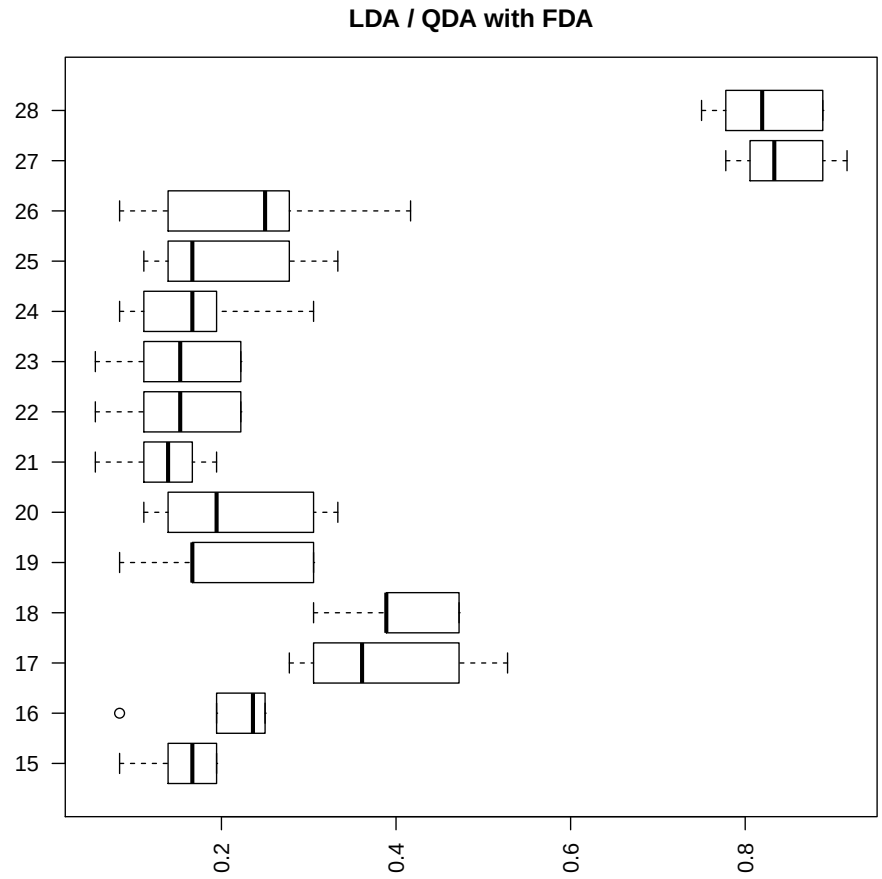


FIGURE 6.4 – LDA & QDA : Taux d'erreurs de test avec FDA intégrée à la 6 folds CV

Détail des résultats Les détail de nos résultats sont disponible au sein des différents exports finaux du dossier https://github.com/goujonpa/SY19TP7/tree/master/csv/final_res disponible sur notre repo github.

Interprétation Ces résultats semblent corroborer nos hypothèses précédentes :

- La LDA s'en sort mieux que la QDA
- La FDA conduit à de l'overfitting détecté dès que l'on effectue sont fitting seulement sur une partie du dataset initial
- Doubler le nombre de PC n'améliore que relativement nos performances
- Des patterns permettent bien de différencier les différentes expressions, sans quoi nous serions incapables d'avoir un taux d'erreur de test atteignant les 13% (LDA, PC25)

Chapitre 7

Support Vector Machine

7.1 SVM - Principe

Principe En machine learning, les SVMs sont un modèle d'apprentissage supervisé adapté à la classification. Un modèle SVM représente les individus d'apprentissage dans l'espace en tentant de séparer les différentes classes par une marge aussi grande que possible.
Tout individu de test est projeté dans cet espace et classifié en fonction de son positionnement par rapport à cette marge.

7.2 Multinomial classification - Strategy

Stratégies Plusieurs stratégies peuvent être utilisées pour utiliser les SVM dans la cas de tâches de classification entre plusieurs classes.

- One vs One : la stratégie One versus One consiste à construire $\binom{K}{2}$ SVMs, comparant les classes par paires, une à une. Les individus de test sont ensuite classés en utilisant chacun des classifieurs, et on leur attribue la classe la plus fréquemment attribuée par ces derniers.
- One vs All : la stratégie One versus All consiste à construire K SVMs, comparant à chaque fois l'une des classes à toutes les autres. On attribue par la suite aux individus de test la classe ayant l'indice de confiance le plus élevé.

Librairie R - Stratégie Nous utilisons dans le cadre de notre étude la librairie `e1071`, nous fournissant une fonction `svm`. Il nous paraissait donc important de préciser la stratégie utilisée pour la classification multinomiale. La documentation du package nous a permis de trouver la réponse : la stratégie "one versus one" est utilisée dans le cadre de notre étude.

7.3 Protocole experimental, résultats & interprétations

Démarche Nous partons sans hypothèse a priori concernant le modèle le plus approprié pour notre dataset. Notre démarche va donc consister à effectuer un premier "balayage", en testant un large spectre de paramètres, puis sélectionner un ou plusieurs jeux de paramètres nous ayant donné de bons résultats puis essayer d'optimiser ces jeux de paramètres.

Première analyse Nous commençons donc au cours d'une première analyse à tester ce large spectre de paramètres. Grace à la méthode `tune` du package `R`, nous pouvons faire varier les paramètres suivants :

- **Kernel** : Le noyau utilisé par la SVM
- **Cost** : Le "budget", agissant sur le "bias-variance trade-off" du modèle en permettant plus ou moins de violation de "margin"
- **Degree** : Le degré à utiliser (seulement dans le cas d'un noyau polynomial)

Résultats La figure ci-dessous présente le résultat de ce premier "parameters tuning".

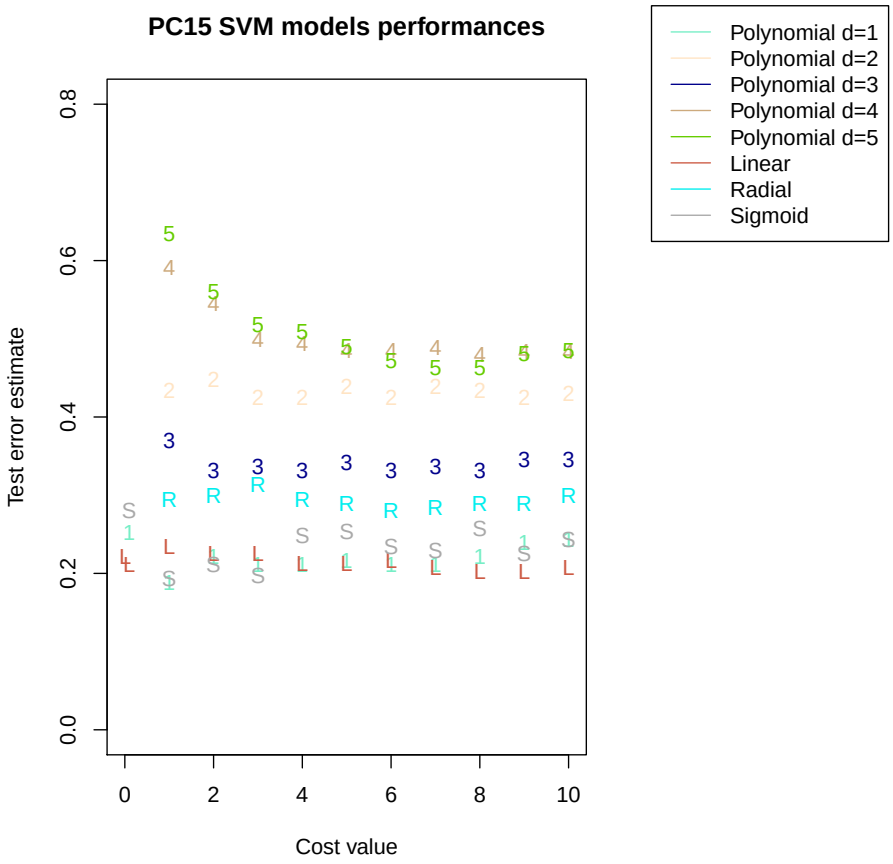


FIGURE 7.1 – SVM : Première analyse

Interprétation - modèles approfondis Au vu de la figure ci-dessus et des performances du tuning que nous avons exportées au format xlsx (annexes), il semblerait que les modèles les plus performants soient le modèle utilisant un noyau de forme `sigmoid` pour un `cost` égal à 1, et le modèle `polynomial` de degré 1 (linear donc), pour un `cost` avoisinant la valeur 5.
On remarque également les performances relativement médiocres de nos modèles pour un `cost` nul. Cela corrobore ce que nous avons vu en cours : l'ajout du `cost` permet d'optimiser notre "bias-variance tradeoff".
Nous sélections ces deux modèles, afin de leur faire passer une phase d'optimisation supplémentaire.

Modèle Sigmoid - Optimisation Nous commençons par optimiser le modèle utilisant un noyau `sigmoid`. nous lui faisons passer deux phases supplémentaires de "tuning". Au cours de la première, nous faisons varier la valeur de `gamma` et de `cost`, deux paramètres utilisés par ce modèle. Nous obtenons les meilleures performances pour les valeurs suivantes :

TABLE 7.1 – Résultats de première optimisation du modèle à noyau sigmoid

Gamma	Cost	Test error estimate
0.007	1.5	0.24

Optimisation du cost Nous fixons par la suite gamma à la valeur obtenue et réalisons notre dernière optimisation, sur le paramètre `cost`. La courbe ci-dessous représente l'évolution de de l'estimation du taux d'erreur de test en fonction de la valeur du paramètre `cost`.

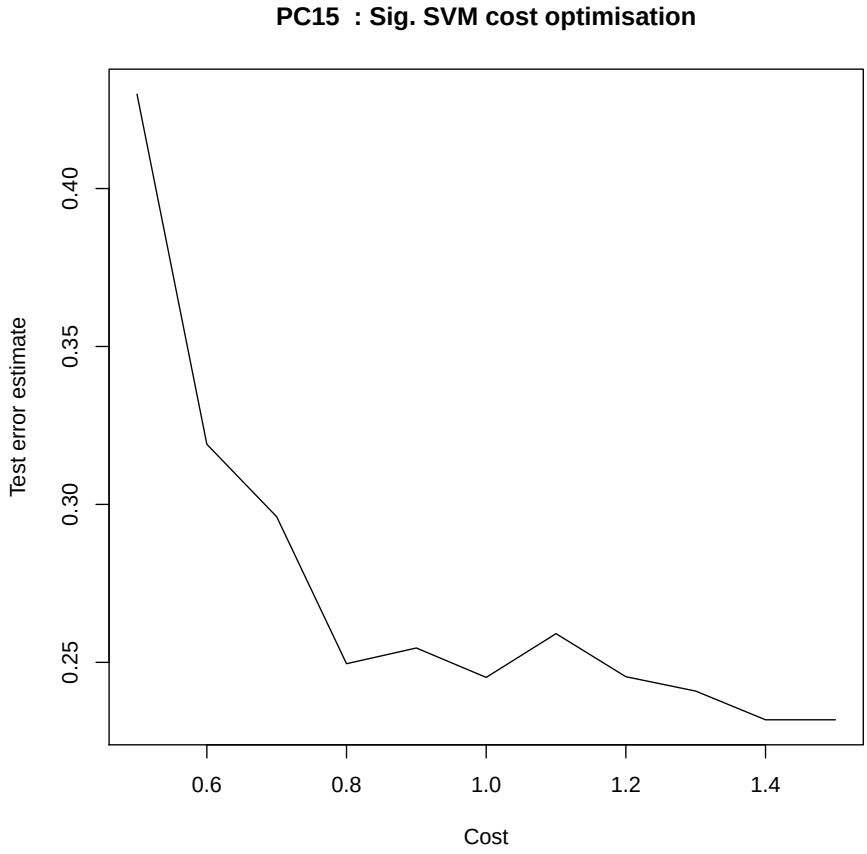


FIGURE 7.2 – SVM : optimisation du cost pour le modèle à noyau sigmoid

Performances finales - noyau sigmoid Finalement, nous obtenons le jeu de paramètre et les performances suivantes :

TABLE 7.2 – Résultats de seconde optimisation du modèle à noyau sigmoid

Gamma	Cost	Test error estimate
0.007	1.1	0.21

Modèle polynomial - optimisation Tout d'abord, il est à noter que le modèle polynomial, au degré 1 est un modèle linéaire, à un facteur `gamma` près. De la même manière que nous l'avons fait pour le modèle à noyau sigmoid, nous "tunons" les deux paramètres de ce modèle. Après la première optimisation, nous obtenons le jeu de paramètres suivant :

TABLE 7.3 – Résultats de première optimisation du modèle à noyau polynomial de degré 1

Gamma	Cost	Test error estimate
0.01	4.3	0.22

Cost optimisation Une nouvelle fois, nous fixons notre `gamma` et effectuons le tuning du `cost`. Nous obtenons la courbe ci-dessous.

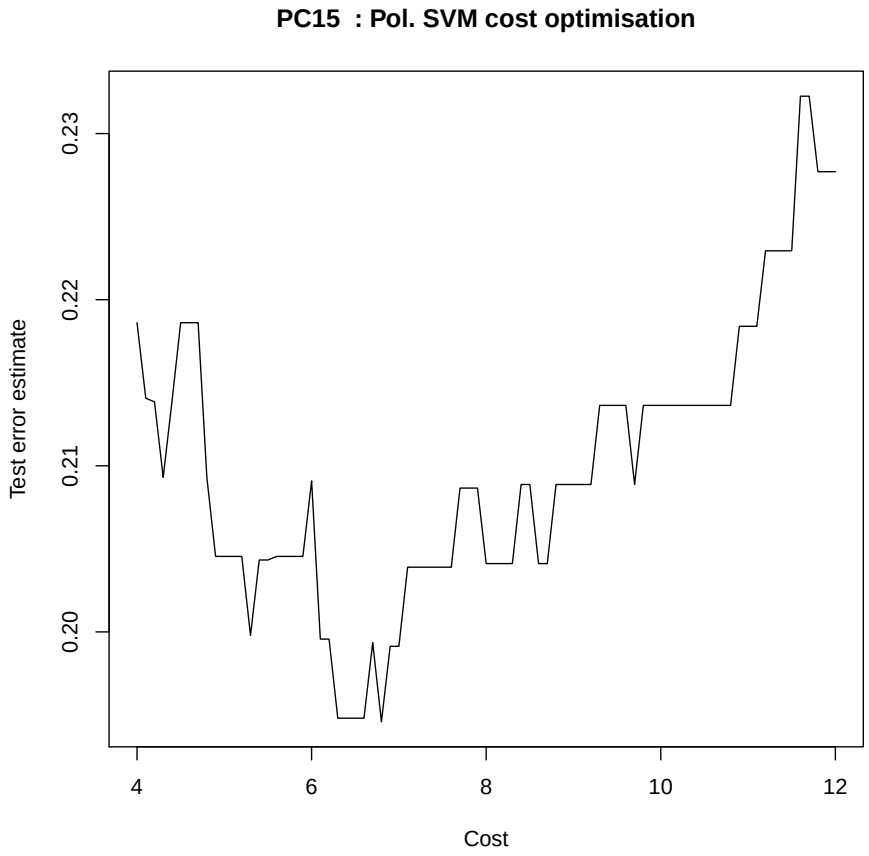


FIGURE 7.3 – SVM : optimisation du cost pour le modèle à noyau polynomial de degré 1

Performances finales - noyau polynomial Après cette nouvelle optimisation, nous obtenons le jeu de paramètres et les performances suivantes :

TABLE 7.4 – Résultats de seconde optimisation du modèle à noyau polynomial de degré 1

Gamma	Cost	Test error estimate
0.01	6.8	0.21

Test errors & confusion matrix Pour finir, nous utilisons notre 6-fold cross-validation "homemade" afin dessiner, pour chacun des deux modèles, boxplots d'estimations d'erreur de test, et matrices de confusion ci-dessous.

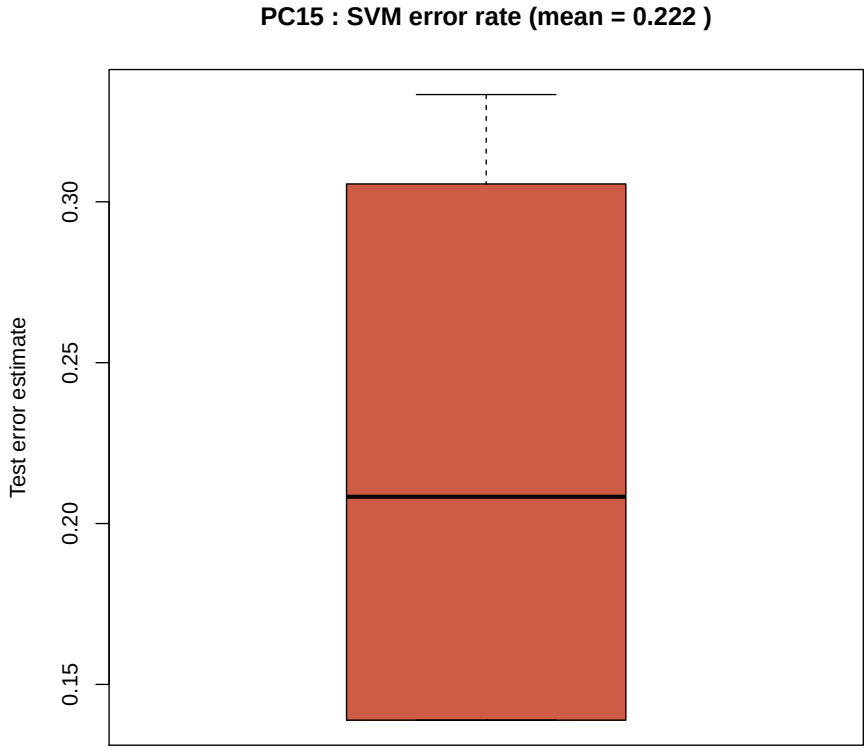


FIGURE 7.4 – SVM : estimation du taux d'erreur pour le modèle à noyau polynomial de degré 1

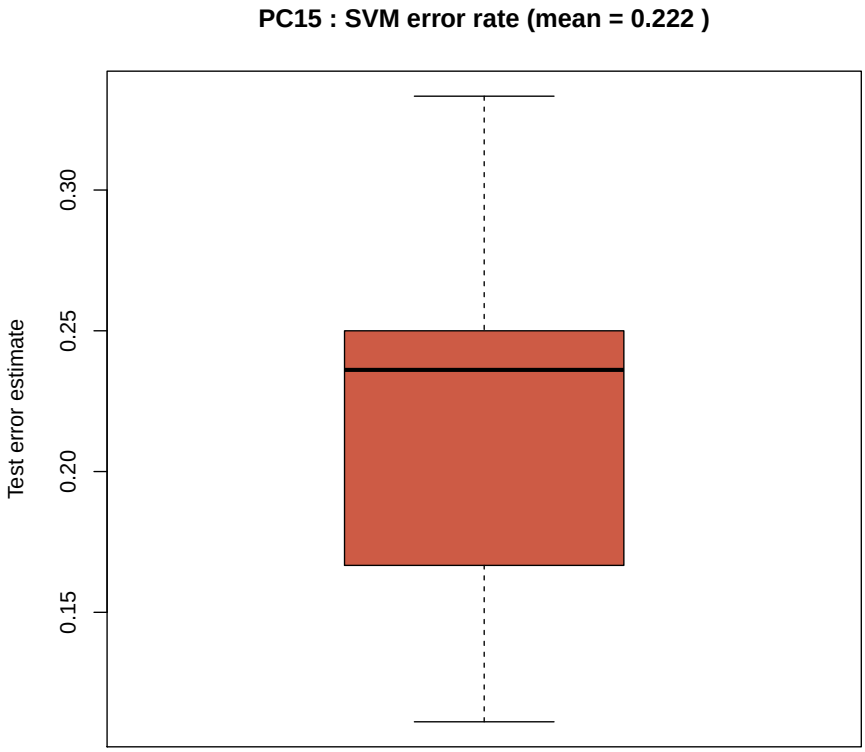


FIGURE 7.5 – SVM : estimation du taux d'erreur pour le modèle à noyau sigmoid

TABLE 7.5 – SVM - polynomial d=1 : Matrice de confusion

Actual \ Predicted	Joie	Surprise	Tristesse	Dégoût	Colère	Peur	Total
Joie	33	0	0	0	0	3	36
Surprise	0	36	0	0	0	0	36
Tristesse	0	0	27	1	6	2	36
Dégoût	1	0	0	28	6	1	36
Colère	0	0	4	6	26	0	36
Peur	8	1	1	2	2	22	36
Total	42	37	32	37	34	28	216

Table 7.6 – SVM - sigmoid : Matrice de confusion

Actual \ Predicted	Joie	Surprise	Tristesse	Dégoût	Colère	Peur	Total
Joie	32	0	0	1	0	3	36
Surprise	0	32	0	0	4	0	36
Tristesse	0	0	27	2	7	0	36
Dégoût	0	0	1	23	12	0	36
Colère	0	0	4	11	20	1	36
Peur	6	0	3	3	1	23	36
Total	38	32	35	40	44	27	216

Interprétations finales Le modèle à noyau polynomial de degré 1 (donc linéaire) performe mieux que le modèle à noyau sigmoid. Son taux d'erreur de test est moins important, sa variance également. Cela semble corroborrer les résultats obtenus précédemment, lors de la LDA.
Notons également de fortes similitudes entre la matrice de confusion obtenue via SVM à noyau polynomial de degré 1, et celle de la LDA (sensiblement les mêmes confusions ont lieu).
Tout comme celles de la LDA, les performances, de la SVM linéaire sont relativement convenables pour cette tâche de facial recognition ! Après avoir exposé nos résultats pour nos autres datasets, nous poursuivons notre analyse à l'aide des Random Forest.

7.4 Autres datasets - Résultats

Résultats Les figures ci-dessous présentent les résultats que nous obtenons pour toutes les formes de SVM testées, sur nos différents datasets.

Table 7.7 – Tableau de correspondances

Dataset	SVM Sigmoid	SVM Polynomial
PC15	29	31
PC15 FDA	30	32
PC25	33	35
PC25 FDA	34	36

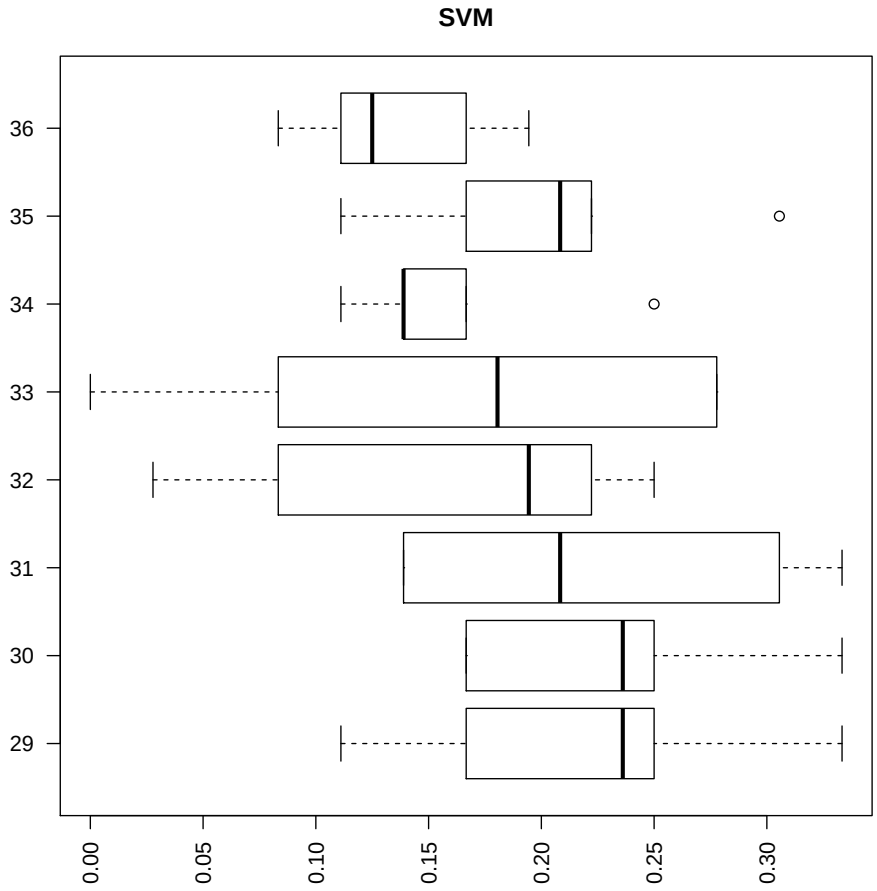


FIGURE 7.6 – SVM : Taux d'erreurs de test

Détail des résultats Les détail de nos résultats sont disponible au sein des différents exports finaux du dossier https://github.com/goujonpa/SY19TP7/tree/master/csv/final_res disponible sur notre repo github.

Interprétation On constate ainsi les très bonnes performances des modèles SVM, rivalisant avec les performances de la LDA.
Nous obtenons nos meilleures performances en utilisant le modèle SVM l'application de la FDA au dataset obtenu en sélectionnant les 25 premières PC de la PCA sur le dataset initial.
Une nouvelle fois, c'est un noyau polynomial de degré 1 (donc linéaire) qui nous fourni les meilleures performances.

Chapitre 8

Random Forest

8.1 Introduction

Principe Les Random Forest, ou Random Decision Forest sont un ensemble de méthodes d'apprentissage adaptées aux tâches de classification (notamment). Elles consistent à construire une multitude d'arbres de décision au moment de l'apprentissange, et renvoyer la classe prédite la plus souvent pour un individu test donné au moment de la prédiction.
Les Random Forest corrigent le principal défaut des arbres de décision seuls : l'overfitting.

8.2 Protocole expérimental, résultat & interprétations

Démarche Une nouvelle fois, nous partons avec aucun a priori concernant le modèle de Random Forest optimal pour notre dataset. De la même manière que nous l'avons fait pour les modèles de type SVM, nous allons effectuer un premier "balayage" des différents paramètres à optimiser, afin de nous faire un première idée du jeu de paramètres optimal. Les paramètres sur lesquels nous allons travailler dans la cadre de la Random Forest sont :

- **Ntree** : Le nombre d'arbres dans notre forêt
- **Mtry** : Le nombre de variables candidates à chaque split

Optimisation Construisant ici la Random Forest sur les composantes principales de notre dataset, les prédicteurs sont au nombre de 15. La construction des Random Forest n'étant pas trop contraignante du point de vue temps de calcul nous allons pouvoir immédiatemment tester toutes les valeurs entre 1 et 15 pour le paramètre `mtry`. Ainsi, notre résultat de tuning sera notre résultat final d'optimisation.
Le "tuning" du jeu de paramètres nous donne ainsi la plot suivante :

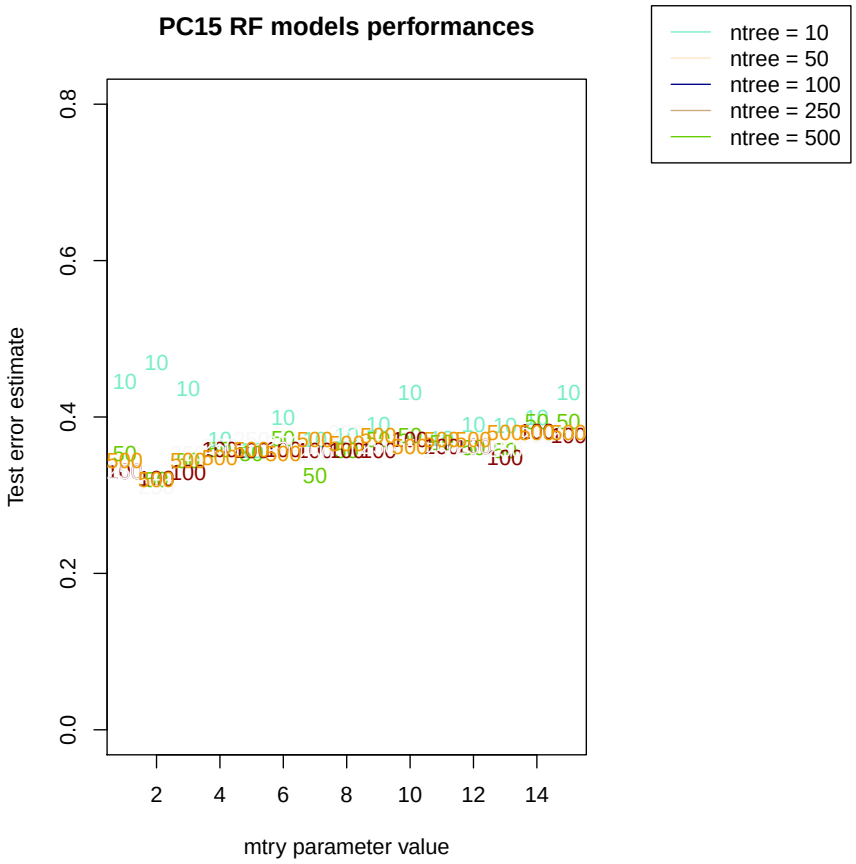


FIGURE 8.1 – RF : Résultat de tuning du jeu de paramètres

Interprétation Cette figure nous permet de voir que les performances de la Random Forest s'améliorent bien au fur et à mesure de l'augmentation du nombre d'arbres dans la forêt, jusqu'à se stabiliser pour un nombre élevé d'entre eux. Notre tuning nous retourne en tant que meilleur jeu de paramètres :

TABLE 8.1 – Résultat de première optimisation de la Random Forest

ntree	mtry	Test error estimate
100	2	0.31

Matrice de confusion et taux d'erreur de test Une fois encore, nous plottons l'estimation du taux d'erreur de test du modèle, ainsi que sa matrice de confusion en utilisant une 6-folds cross validation.

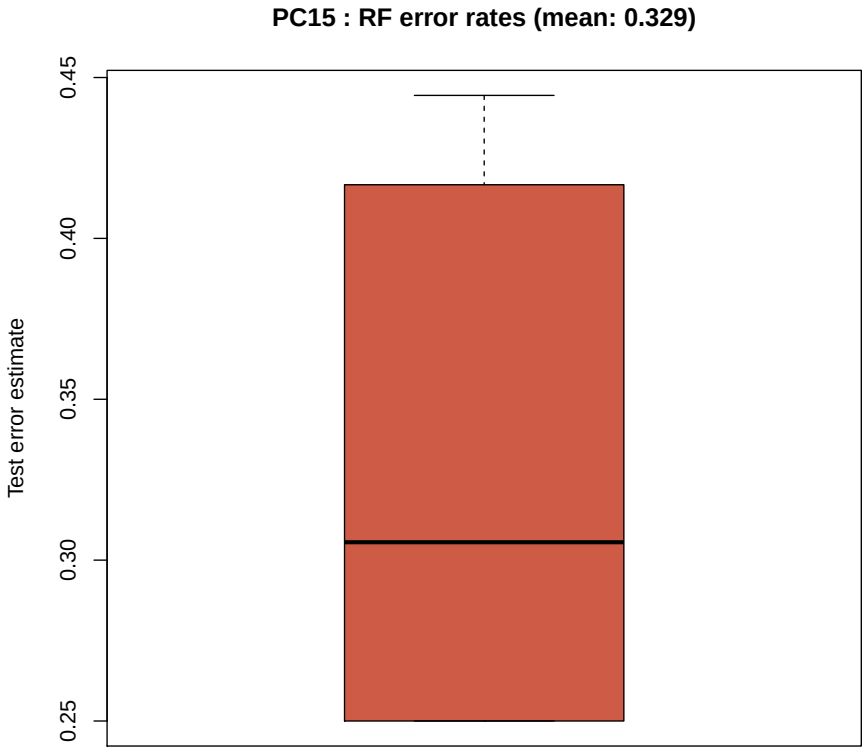


FIGURE 8.2 – RF : Estimation du taux d'erreur de test

Table 8.2 – Random Forest : Matrice de confusion

Actual \ Predicted	Joie	Surprise	Tristesse	Dégoût	Colère	Peur	Total
Joie	27	0	1	0	1	7	36
Surprise	0	32	0	2	1	1	36
Tristesse	0	0	26	5	4	1	36
Dégoût	0	2	4	20	7	3	36
Colère	0	0	5	6	24	1	36
Peur	7	1	2	2	3	21	36
Total	34	35	38	35	39	33	216

Interprétation Avec la Random Forest, nous n'obtenons des performances bien moins intéressantes qu'avec la LDA, ou la SVM polynomiale de degré 1. De plus la variance de ce modèle semble relativement élevée.
Nous pouvons émettre l'hypothèse d'un éventuel overfitting : peut être qu'en prunant les arbres composant la forêt, nous ajouterions un léger biais mais améliorerions le "bias-variance tradeoff" du modèle.

8.3 Autres datasets - Résultats

Résultats Les figures ci-dessous présentent les résultats que nous obtenons pour toutes les formes de Random Forest testées, sur nos différents datasets.

Table 8.3 – Tableau de correspondances

Dataset	Random Forest
PC15	37
PC15 FDA	38
PC25	39
PC25 FDA	40

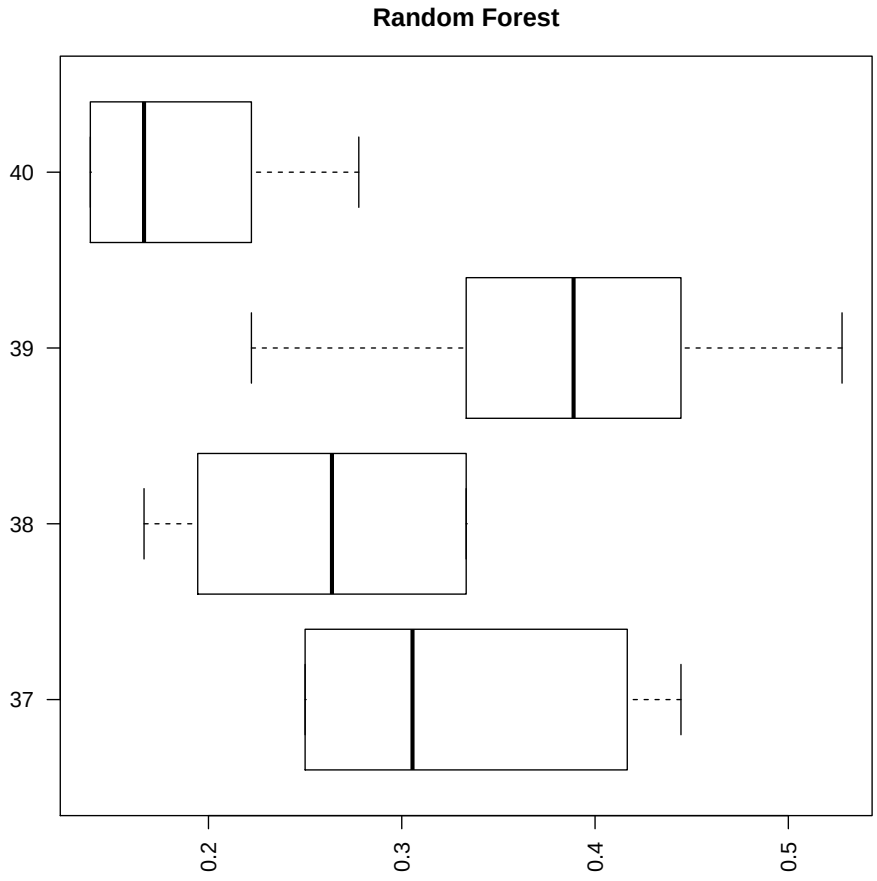


FIGURE 8.3 – RF : Taux d'erreurs de test

Détail des résultats Les détail de nos résultats sont disponible au sein des différents exports finaux du dossier https://github.com/goujonpa/SY19TP7/tree/master/csv/final_res disponible sur notre repo github.

Interprétation Nous n'obtenons pas, avec les Random Forest, d'aussi bonnes performances qu'avec les modèles LDA et SVM.
La variance de ces modèles est elle aussi relativement élevée, comparée à ce que nous avons pu obtenir avec les modèles sus-cités.
Une nouvelle fois, l'application de la FDA semble améliorer les performances les performances de notre modèle.
Il est a noter cependant cette fois ci, que dans le cas ou nous n'appliquons pas de FDA, le fait de sélectionner un plus grand nombre de composantes principales n'améliore pas notre modèle mais semble de dégrader.

Chapitre 9

Neural Networks

9.1 Introduction

Principe Les réseaux de neurones artificiels, sont un ensemble d'algorithmes dont la conception est à l'origine très schématiquement inspirée du fonctionnement des neurones biologiques, et qui par la suite s'est rapproché des méthodes statistiques.
Les réseaux de neurones sont généralement optimisés par des méthodes d'apprentissage de type probabiliste, en particulier bayésien. Ils sont placés d'une part dans la famille des applications statistiques, qu'ils enrichissent avec un ensemble de paradigmes permettant de créer des classifications rapides.

9.2 Protocole expérimental, résultats & interprétations

Démarche Une nouvelle fois, nous n'avons pas d'a priori au moment de commencer nos essais sur le modèle convenant le mieux pour la résolution de notre problème. Nous allons donc procéder, comme précédemment, à un premier balayage suivi d'une optimisation du jeu de paramètre choisis suite au premier balayage. Les paramètres que nous nous apprêtons à optimiser sont les suivants :

- **Size** : Le nombre d'unité internes au Neural Network
- **Decay** : Valeur utilisée pour la dégénérescence des poids

NB : Package R Nous utilisons le package `R` le plus "basique" pour notre implémentation de Neural Network. Ce dernier ne nous permet d'implémenter qu'un simple "Single-Hidden-Layer Neural Network", c'est à dire un NN ne présentant qu'une seule couche cachée. Ainsi, nous n'implémentons pas (pour l'instant du moins) de réseau de neurones présentant des features telles qu'une "connectivité locale", ou des "poids partagés" ("Locally connected Neural Network" et "Convolutional Neural Network"). Peut être prolongerons nous cet TP

par leur implémentation, si le temps ne nous manque pas.

Première optimisation Suite à notre premier "balayage", nous obtenons la plot suivante :

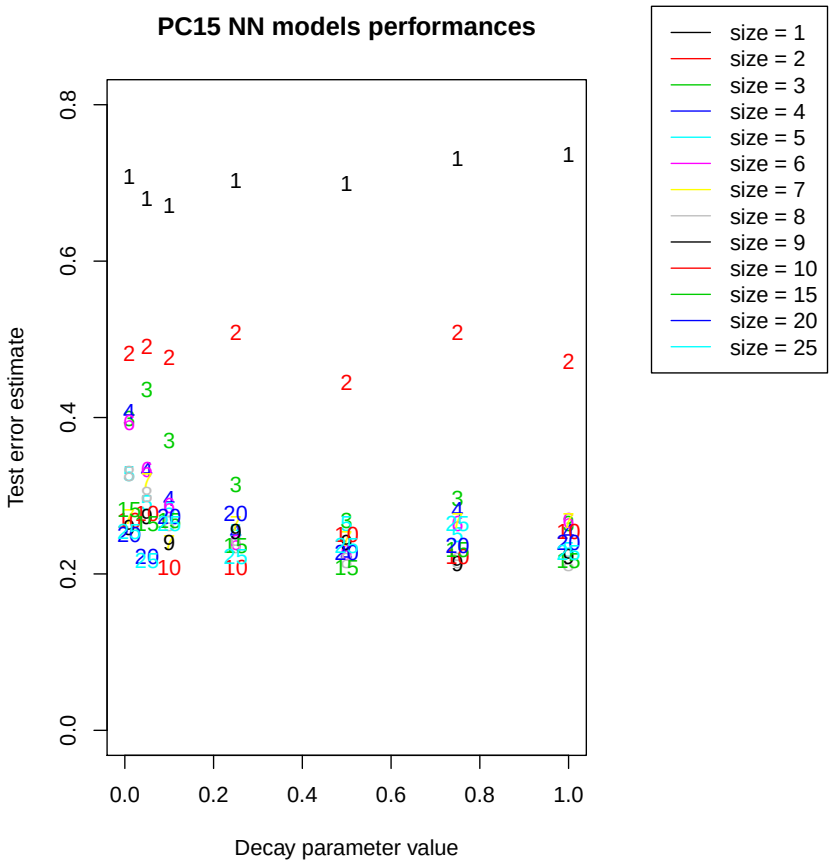


FIGURE 9.1 – NN : Résultat de tuning du jeu de paramètres

Interprétation Les performances de notre réseau de neurone s'améliorent avec l'augmentation du nombre d'unités au sein de la "hidden layer", jusqu'à ce stabiliser pour un nombre élevé.
Cela se confirme, lorsque nous examinons l'output xlsx (ci-dessous) de notre tuning, selon lequel c'est lorsque nous avons choisi un paramètre de taille égal à 25 (le maximum que nous ayons testé) que nous obtenons les meilleures performances.

TABLE 9.1 – Résultat de première optimisation du Neural Network

size	decay	Test error estimate
25	1	0.20

Optimisation du Weight Decay Fixant la taille de notre NN à 25, nous optimisons le paramètre de `weight decay` et obtenons la courbe ci-dessous.

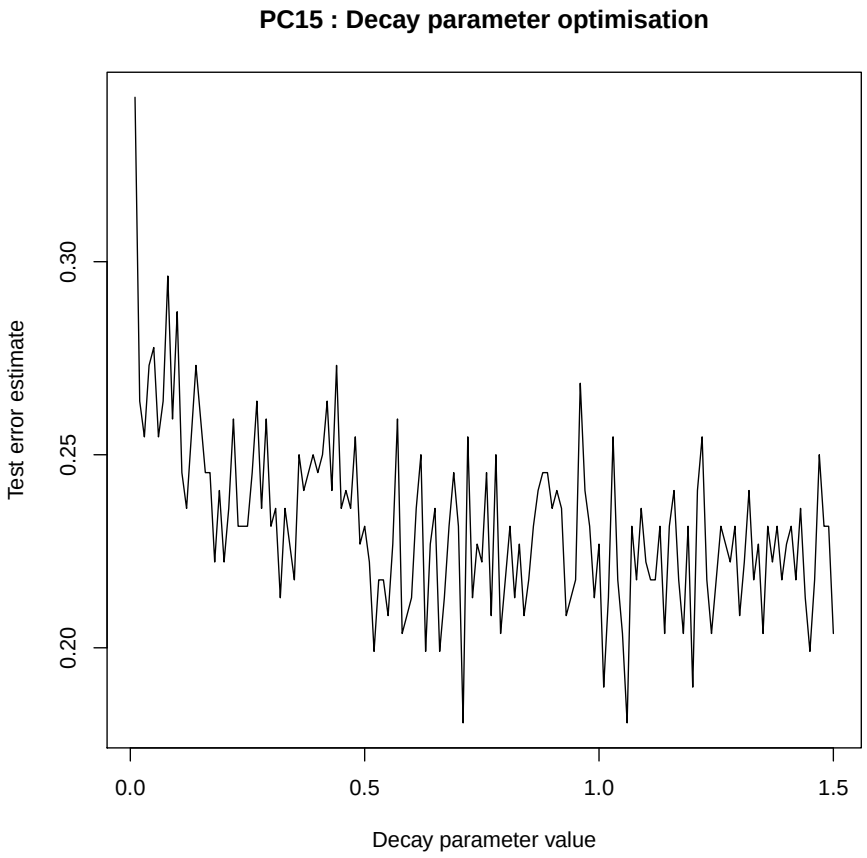


FIGURE 9.2 – NN : optimisation du weight decay

Jeu de paramètres optimal Nous obtenons après optimisation le jeu de paramètres suivant :

TABLE 9.2 – Résultat de seconde optimisation du Neural Network

size	decay	Test error estimate
25	1.27	0.19

Erreur de test et matrice de confusion Pour finir, nous utilisons une dernière fois (peut être pas la dernière) la 6-fold cross validation pour plotter notre estimation de l'erreur de test de notre NN, et sa matrice de confusion.

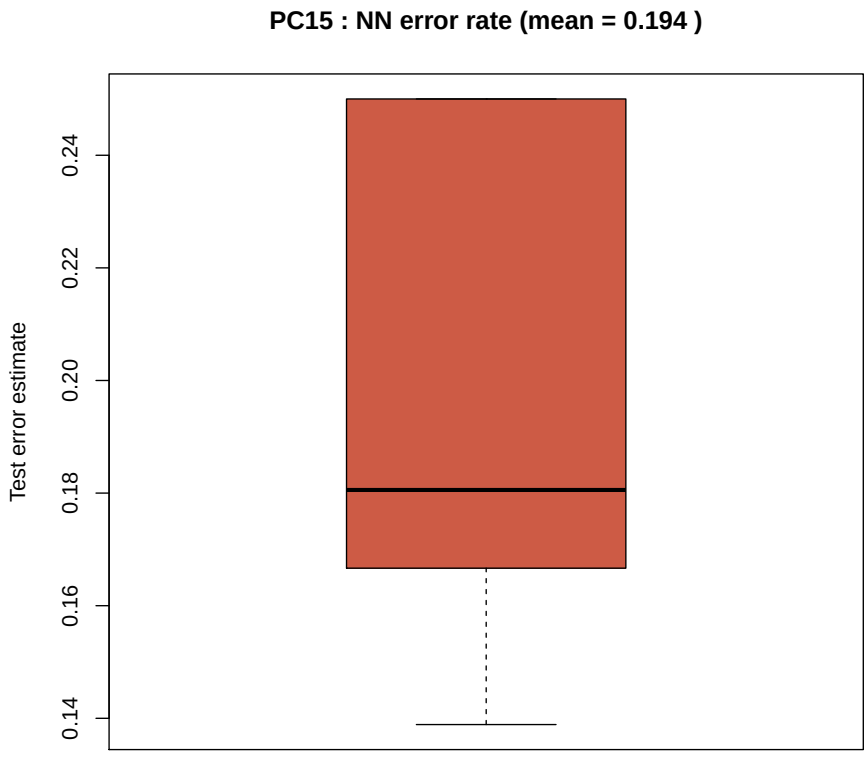


FIGURE 9.3 – NN : Estimation du taux d'erreur de test

Table 9.3 – Neural Network : Matrice de confusion

Actual \ Predicted	Joie	Surprise	Tristesse	Dégoût	Colère	Peur	Total
Joie	31	0	0	1	0	4	36
Surprise	0	36	0	2	1	1	36
Tristesse	0	0	27	4	4	1	36
Dégoût	1	1	1	28	4	1	36
Colère	0	0	5	8	21	2	36
Peur	7	0	2	0	2	25	36
Total	34	35	38	35	39	33	216

Interprétation L'estimation du taux d'erreur de test du NN optimisé ainsi obtenu est relativement basse, cependant sa variance est relativement élevée, ce qui n'en fait pas le modèle le plus sûr.

9.3 Autres datasets - Résultats

Résultats Les figures ci-dessous présentent les résultats que nous obtenons pour toutes les formes de Neural Network testées, sur nos différents datasets.

Table 9.4 – Tableau de correspondances

Dataset	Neural Network
PC15	41
PC15 FDA	42
PC25	43
PC25 FDA	44

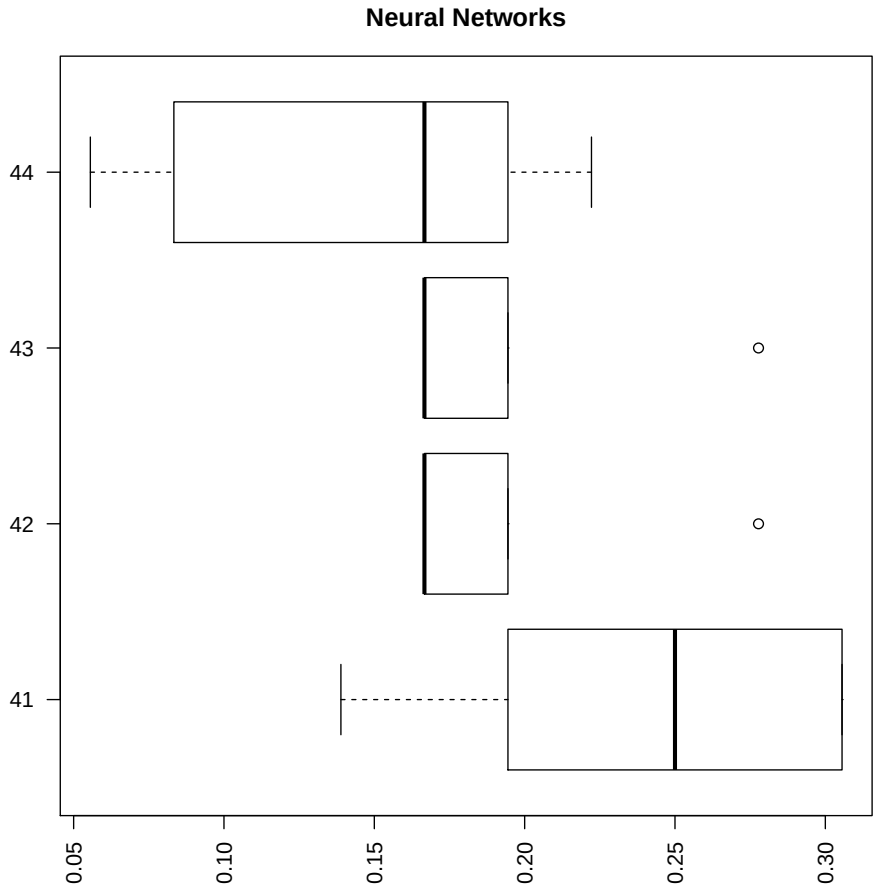


FIGURE 9.4 – NN : Taux d'erreurs de test

Détail des résultats Les détail de nos résultats sont disponible au sein des différents exports finaux du dossier https://github.com/goujonpa/SY19TP7/tree/master/csv/final_res disponible sur notre repo github.

Interprétation Les réseaux de neurones offrent eux aussi des performances plus que respectables.
On remarque même que le NN entrainé sur le jeu de données auquel on applique la FDA atteint parfois un taux d'erreur proche de 5% !
Cependant la variance de ces modèles est relativement élevé, ce qui n'en fait pas le meilleur modèle selon nos critères.

Chapitre 10

Conclusion

10.1 Comparaison des classifieurs

Meilleur classifieur Une fois testées différentes hypothèse, sur différentes transformation du dataset initial, données en input à différents modèles, le travail du data scientist consiste à sélectionner le modèle qui lui semble le plus approprié à son modèle.
Nous allons donc sélectionner le modèle dont l'estimation de l'erreur de test est la plus basse, et de variance la plus faible possible.
Au vu des résultats de notre batterie de test (cf exports xlsx), il semblerait que le modèle donnant les meilleures performances est le modèle SVM linéaire, appliqué sur les 25PC auxquelles on applique la FDA, donnant les taux d'erreur de test et variance du taux d'erreur de test suivants :

Table 10.1 – Tableau de correspondances

Modèle	Test Err. Rate	Test err. rate s.d.
Linear SVM	13%	3%

10.2 Perspectives

Volonté vs temps L'implémentation de cette batterie de test a pris beaucoup de temps, de par la rigueur que nous avons essayé de mettre à la fois au centre de notre démarche scientifique, mais aussi de notre manière de coder.
Nous avions donc initialement envisagé l'implémentation d'autres modèles, que nous n'avons pu réaliser, faute de temps. Ci-dessous voici un aperçu des perspectives d'amélioration de notre TP. Nous aurions pu :

— Tenter la sélection des composantes principales par Forward Stepwise Selection

— Utiliser la librairie Neuralnet pour essayer une plus grande diversité d'implémentations des Réseaux de Neurones
— Utiliser d'autres méthodes de réduction de dimensions
— Utiliser d'autres méthodes d'estimation de l'erreur de test telles que le Bootstraping
— ... etc

10.3 Conclusion

Conclusion Au cours de ce TP, nous avons adopté la démarche du data scientist, ayant un objectif de classification à atteindre, avec dataset fourni.
Nous avons pu tester différentes hypothèse quant à la meilleure transformation du dataset initial à effectuer. Nous avons travaillé sur l'implémentation de différents modèles et leur optimisation. Finalement nous avons essayé de tirer des conclusions quant aux performances des différents modèles, et la sélection du plus adapté d'entre eux.
Ce TP a été très formateur, aussi bien d'un point scientifique que d'un point de vue technique, et nous aura permis de faire un tour d'horizon très large des connaissances enseignées en SY19.