Janvier 2022 Ecole des Mines de Saint-Etienne

Auteur: Mbaye DIONGUE TB3: Image and Pattern Recognition

RAPPORT DE PROJETS

PLAN:

Introduction:

Le but principal de ces deux projets est d'utiliser des **descripteurs morphologiques**, d'**intesnité** et de **texture** les plus pertinents afin de classifier des images en groupes distincts (développement d'un cancer de la peau ou non par exemple). Pour effectuer cette classification, les outils d'apprentissage supervisés tels que la **régression logistique**, les **SVM** (Support Vector Machine), les **réseaux de neurones** et les **forêts aléatoires** seront utilisés. Dans chacun des cas, la qualité des prédictions sera jugée à l'aide d'indicateurs comme l'***accuracy*** *(justesse ou non de la prédiction),* le ***F1-score*** et la courbe **ROC**.

NB: Dans le premier projet où les descrpteurs sont déjà fournis, nous insisterons particulièrement sur les algorithmes de classification utilisés en les présentant sommairement et en comparant leurs résultats grâce à des indicateurs. Dans le deuxième projet par contre, nous mettrons plus l'accent sur la sélection de descripteurs pertinents, ces derniers étant génerelement choisis parmi les descripteurs qu'on a pu rencontrer lors des TPs.

I- Projet 1: Classification de lames d'outils de coupe

I-1 Mise en contexte

Dans l'industrie de métallique, la découpe des métaux entraine généralement une isure de la pièce coupante qui a alors tendance à se déformer. Cela entraine alors des *défauts* dans les pièces découpées. On comprend alors l'importance en *milieu industriel* de disposer d'un **outil de détections automatiques** qui soit capable de dire si la bordure de la lame coupante est trop usée ou non pour effectuer correctement les découpes et ainsi de décider éventuellement de la remplacer.

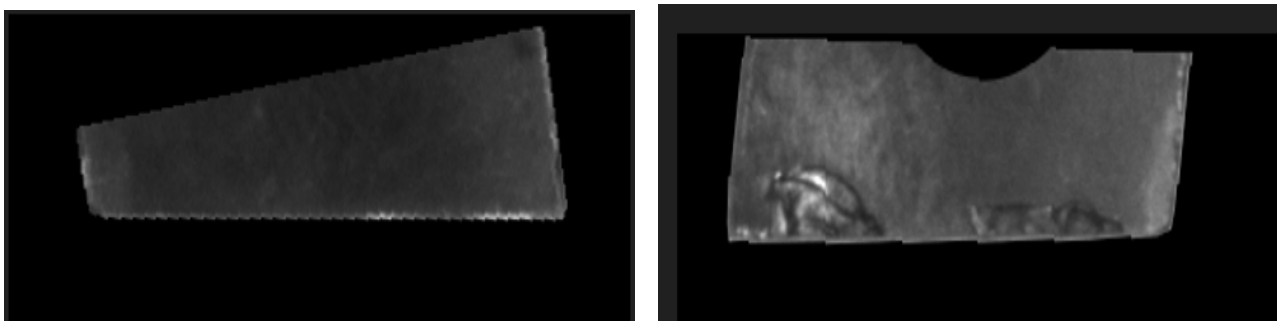


Figure 1: Lame peu usée (à guache) et lame très usée (à droite)

Dans cette première partie, la problématique consiste alors à classer les lames de découpe d'une machine en fonction qu'elles sont peu usées (classe **'*low*'**) ou très usées (classe '***high***'). Pour cela, nous disposons d'images de différentes lames labélisées **low/high.** Les différents *prétraitements* effectués pour ne garder que le contour de la région d'intérêt sont résumés dans la figure 2 ci-dessous.

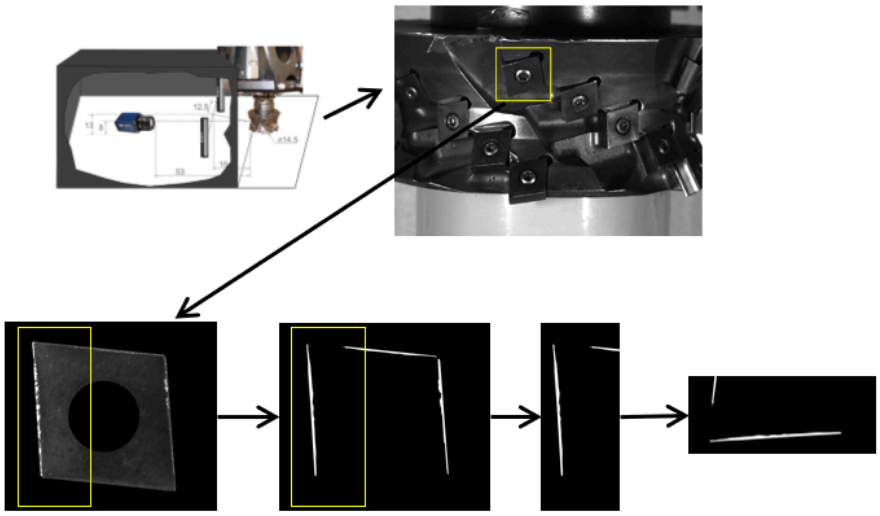


Figure 2: Extraction de la face coupante

L'objectif est alors de construire un classifieur $C_\theta : D \rightarrow \{0, 1\}$ qui soit capable de prédire la classe d'appartenance d'une image non encore labélisée ($C_\theta(x_{new}) = 1$ si x_{new} appartient à la classe 'high' et 0 sinon)

θ est l'ensemble des paramètres du classifieurs qu'il faudra optimiser à partir des images déjà labélisées (ensemble d'apprentissage) pour qu'il puisse faire des prédictions optimales. Typiquement, pour un réseau de neurones θ est une matrice de poids à ajuster, pour une régression il s'agira d'un vecteur de paramètres à optimiser...

Pour l'ensemble de départ D, une idée serait de le prendre comme l'espace des images soit l'ensemble des matrices de taille 1024*1024. Mais l'optimisation du paramètre θ du classifieur C_θ demanderait alors un **très grand nombre de données d'apprentissage** sur un espace d'aussi grande dimension (il faudrait typiquement plusieurs milliers d'exemples). Cela n'est pas envisageable pour notre étude puisqu'on ne dispose que de ***n=202*** données au total. Il est alors nécessaire de ***compresser l'information*** (***réduction de la dimension***), en extraiyant les caractéristiques intéressantes au sein de chaque image.

I-2 Extraction des descripteurs morphologiques

La réduction de dimension évoquée précédemment se fait en extrayant de chaque image un ensemble de descripteurs morphologiques. Dans l'article de support ***"Combining shape and contour features to improve tool wear monitoring in milling processes"***, 10 descripteurs appelés ***ShapeFeat*** sont proposés. Ces 10 descripteurs considérés sont : le *périmètre de la région*, la *surface*, le *diamètre équivalent*, l'*excentricité*, le *demi-grand axe* et le *demi-petit axe* de l'ellipse circonscrite, le *ratio R de ces 2 axes*, la *solidité* et enfin le pourentage de pixels dans la région par rapport au reste de l'image.

L'utilisation de ces ***descripteurs morphologiques*** déjà rencontrés en TP permet alors de construire un classifieur performant avec relativement ***peu de données d'apprentissage***.

I-3) Classification supervisée

Dans cette partie nous rentrons plus en détail dans la construction du classifieur C_θ. Comme rappelé plus haut, le problème de la classification supervisée consiste à trouver un paramètre θ optimal à partir de données d'apprentissage labélisées afin de prédire au mieux la classe d'individus non encore observés (ensemble de test). Dans notre cas, les individus seront les images représentés chacune par un vecteur formé des descripteurs morphologiques extraits précédemment. Nous présentons ici très sommairement les différents algorithmes de classification utilisés dans ce projet.

a. Régression logistique

Un régresseur logistique binaire est une fonction $h_\theta : D \rightarrow [0, 1]$ telle que pour tout individu X dans D ($D = R^{10}$ dans notre exemple car il y'a 10 features) on a:

$$h_\theta(X) = \frac{1}{1 + e^{-\theta^T * X}}$$

Plus précisément pour chaque nouvelle observation h_θ fourni une probabilté $p \in [0, 1]$ d'appartenance à la classe '0' (low) ou à la classe '1' (high). Tout le but du jeu est de régler le vecteur $\theta = [\theta_0, ..., \theta_{10}]$ pour maximiser la performance du classifieur h_θ. Ce paramétrage peut se faire grâce à l'algorithme de **descente de gradient** dont le but principal est de *minimiser* une ***fonction de coût*** f_{cost}.

Dans le cas d'une classification, une fonction de coût naturelle serait:

$$f_{cost}(h_\theta) = \frac{1}{n} \sum_{k=1}^{n} 1_{y_k \neq \hat{y_k}}$$

où y_k est la vraie classe (donnée par un expert par exemple) et $\hat{y_k}$ est la classe prédite par le régresseur (obtenue en appliquant un seuil sur la probailité d'appartenance renvoyée par h_θ). On remarquera qu'il s'agit simplement du ***taux de mauvais classements.***

Mais généralement, il est préférable d'utiliser la fonction de perte ***Entropie*** qui provient du *Principe du Maximum de Vraisemblance* et s'écrit:

$$Entropie(h_\theta) = -\frac{1}{n} \sum_{k=1}^{n} y_k * log(\hat{y_k}) + (1 - y_k) * log(1 - \hat{y_k})$$

Au cours de ce projet, nous avons implanté une fonction de régression logistique. Ci-dessous sont représentées l'évolution de la fonction de la perte au cours de l'apprentisssage et la matrice de confusion sur les données de test:

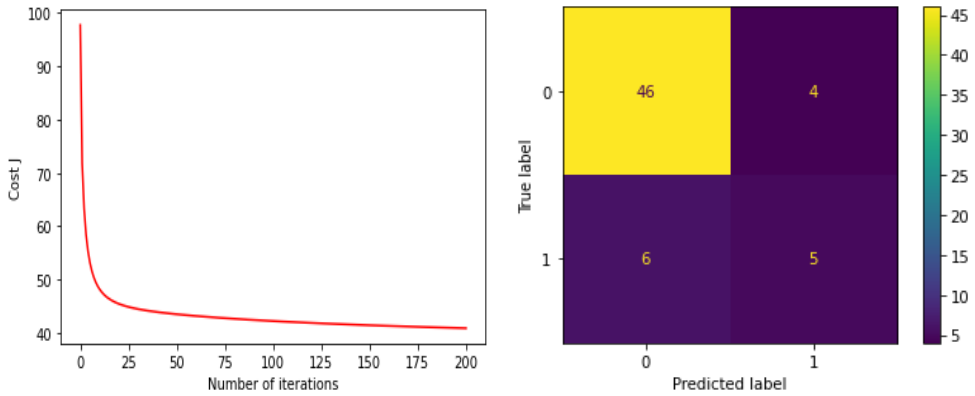


Figure 3.1: Evolution de la fonction de coût au cours de l'apprentissage et matrice de confusion

```
 Initial cost : 97.73375245895242
[[46  4]
 [ 6  5]]
***************
The accuracy of the Logistic regression classifier is 0.836
***************

***************
The F1-score of the Logistic regression classifier is 0.902
***************
 AUC=0.831
```

Figure 3.2: Détail des résultats régression logistique

Comme on pouvait s'y attendre, la fonction de perte diminue au cours de l'apprentissage grâce au mécanisme de descente de gradient dont le rôle est de minimiser cette fonction de perte. Cependant, cette diminution est ralentie au fur du temps car on atteint un minimum (éventuellement local) et que les pas de gradient sont suffisamment. Ainsi, même si on est tenté d'augmenter le nombre d'itérations, on voit que ce ne serait pas d'une grande utilité. D'autant plus que cela ferait apparaître un problème de surappprentissage ou overfitting (modèle incapable de généraliser correctement sur de nouvelles données de test).

La matrice de confusion dressée permet alors de comparer les performances du classifier en comparant ses prédictions avec les vraies classes observées sur les donnnées de test. On constate alors qu'on a **10 *erreurs*** de classification ***sur 61 individus***, soit un taux de bonnes **prédictions de 83.6%**. Nous détaillerons, plus loin dans ce rapport la performance des classifieurs (en utilisant d'autres critères) mais on peut déjà constater qu'il fournit un résultat 'très satisfaisant'. Celà montre en particulier que les **10 descripteurs morphologiques** utilisés est bien pertinent pour extraire l'information importante des photos.

b. SVM (Séparateurs à Vaste Marge)

Dans le cas des SVM, nn dispose d'un échantillon d'apprentissage $X_1, X_2, ..., X_p$ correspondant à p vecteurs de R^n (n=10 est le nombre de descripteurs pour notre cas). L'objectif sera de constuire une fonction f à partir de cet ensemble d'apprentissage de telle sorte pour tout nouvel individu $x_{new} \in R^n$ la fonction lui associe sa classe d'appartenance (par exemple: $f(x_{new}) = 1$ si la cellule est saine et $f(x_{new}) = 0$ si la cellule est cancéreuse).

Dans le cas linéaire par exemple, l'objectif est de construire un hyper **plan de séparation** $\pi = w^T x + b$ **de marge maximale** où les variables recherchées sont $w \in R^n$ et $b \in R$. Dans le dessin en 2D ci-dessous, π est une droite de pente ω et d'ordonnée à l'origine b.

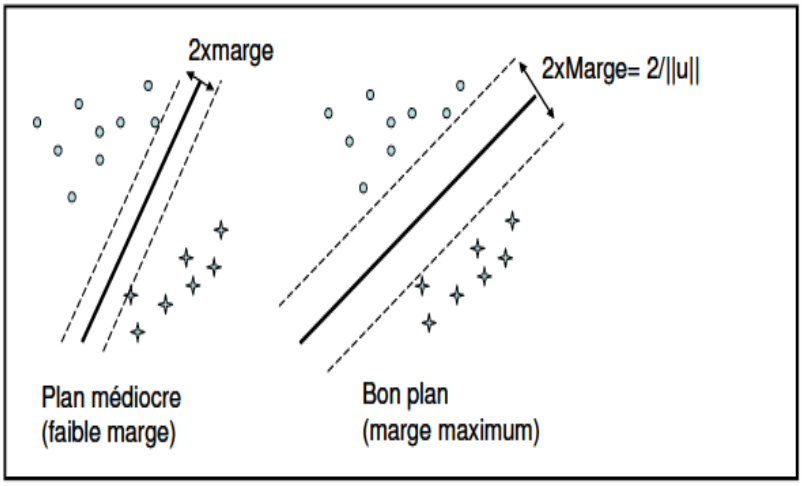


Figure 4: Illustration SVM en 2D

La fonction de décision est alors $f(x) = w^T x + b$ telle que: $f(x_{new}) = \begin{cases} 1 \text{ si lame trop défectueuse} \\ 0 \text{ sinon} \end{cases}$

On peut montrer que l'obtension paramètres w et b optimaux se fait en *résolvant un problème d'optimisation sous contrainte*. Nous fournissons **en annexe** l'ensemble de ces détails techniques qui ne constituent pas forcément le premier objectif de ce rapport.

Voici la matrice de confusion ainsi que la courbe ROC obtenue dans la classification des lames usées et non usées.

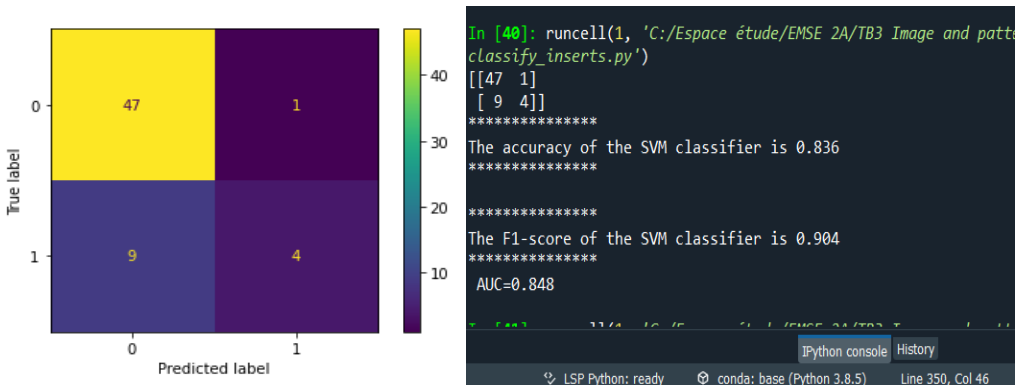


Figure 5: Résulats SVM

Même si on a un seul faux positif, le SVM fait une performance comparable à la régression logistique. L'essentiel des erreurs (9 sur 10) provient de faux négatifs (individus classés à '0' alors qu'ils appartiennent à la classe "1"). Ceci est typiquement un problème de seuillage, puisque'on utilise un seuil de 0.5 par défaut pour transformer les probabilités de sortie en classe d'affectation. Nous y reviendrons sur ce problème dans la partie consacrée à la courbe ROC.

c. Réseau de neurones

Les réseaux de neurones connaissent un grand succès dans la classification des images grâce à l'utilisation d'architecture particulière comme les réseaux convolutifs. Mais ces derniers font apparaitre un grand nombre de paramètres (ce sont poids) et nécessite une très grande quantité d'exemples. De plus, comme évoqué princédemment nous remplaçons chaque image par un ensemble de descripteurs caractérisques, ce qui n'est pas le cas dans le cadre du deep learning.

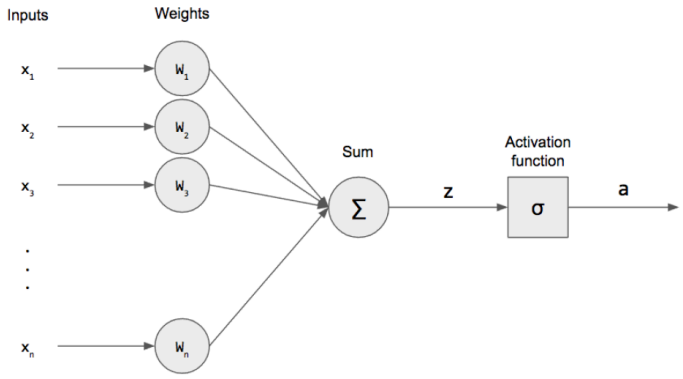


Figure 6: Schéma de principe d'un réseau de neurones simple (n entrée, une sortie)

Les réseaux de neurones artificiels s'inspirent du fonctionnement des neurones du cerveau. Chaque noeud du réseau (neurone) reçoit de la part des m neurones de la couche précédente un vecteur d'entrée$x_{in} = [x_{in}^1, ..., x_{in}^m] \in R^m$pondérées par des poids $w = [w_1, ..., w_m]$. Le neurone fait l'agrégation de ces entrées pondérées, applique une fonction d'activation σ (une sigmoide par exemple) et renvoie une sortie y_{out} aux neurones de la couche suivante. La phase d'entrainement consiste à optimiser les poids du réseau grâce à une descente de descente de gradient comme dans le cas de la régression logistique.

On obtient encore une fois un taux de bonnes prédictions supérieur à 80% et une AUC (aire sous la courbe ROC) du même ordre de grandeur. On remarquera que les réseaux de neurones sont moins performants que les précédents méthodes, cela étant essentiellement dues à la présence de beaucoup de paramètres (85)

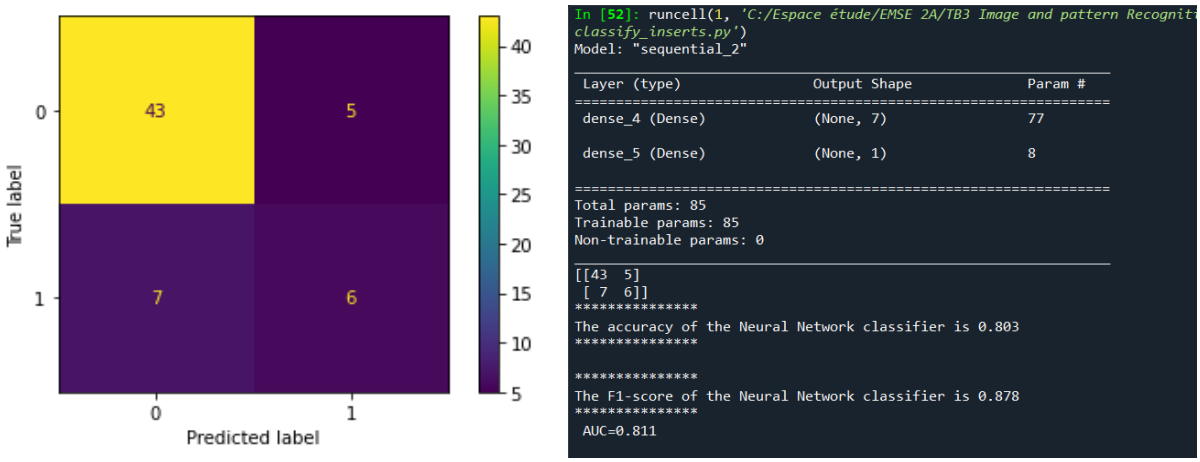


Figure 7: Resultats réseau de neurones

d. Forêts aléatoires

Une forêt aléatoire comme son nom l'indique est une agrégation de plusieurs arbres de décision. Chaque arbre fait une prédiction pour la classe de sortie et la décision finale est donnée par la classe ayant obtenue le 'plus de vote' (on peut cepeandant pondérant l'importance de chaque arbre dans la décision finale à partir de ses performances sur les données de test). Chaque arbre n'utilise pas forcément toutes les variables, mais le regroupement de tous les arbres font la robsutesse du modèle.

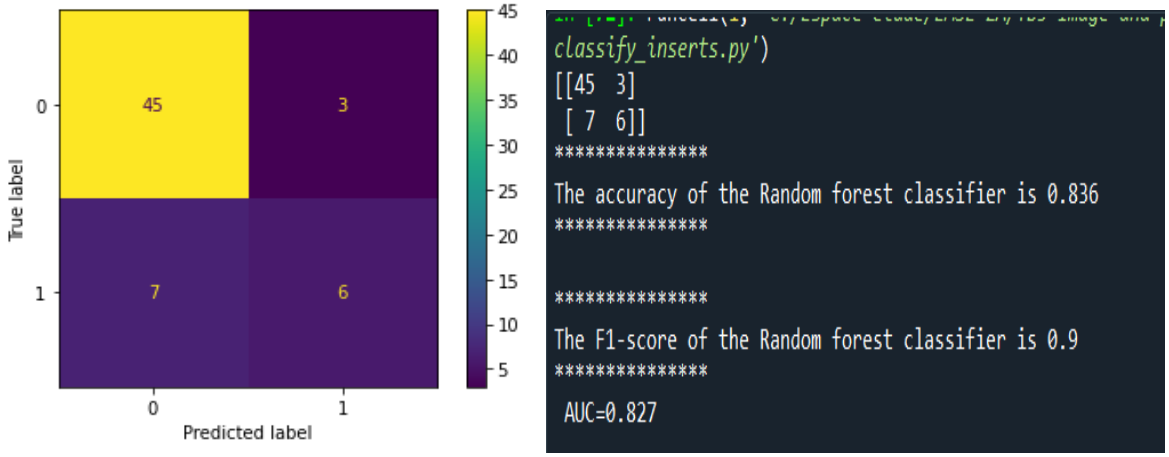


Figure 8: Résultast forêts aléatoires

Encore une fois on retrouve un résultat, analogue que dans les autres modèles, avec un taux de positifs toujours aussi négatifs deux faux plus élevés que celui de faux positifs.

I.4 Mesures de performance et optimisation

Dans la partie précédente, on a observé que les algorithmes avaient à peu près les mêmes performances en terme de justesse des prédictions (accuracy). Mais pour mieux les juger leur performance il faudrait considérer le F1-score, la précision et le recall défini comme suit.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

avec TP, FP qui sont respectivement le nombre de vrais positifs et de vrais négatifs.

On s'intéresse aussi à la courbe ROC qui correspond au tracé de TP en fonction FP. Voici par exemple la courbe ROC obtenue avec la régression logistique et les SVM.

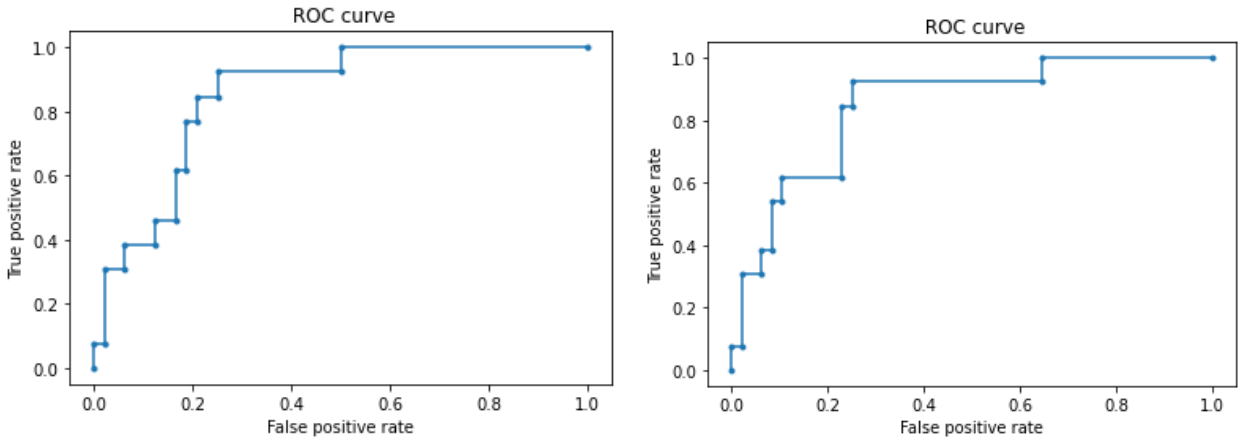
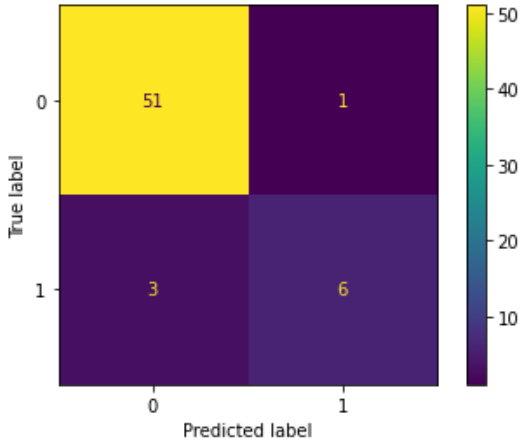


Figure 9: Courbes ROC avec la régression logistique (à gauche) et les SVM (à droite)

Comme on pouvait si attendre $TP = f(FP)$ est une fonction fonction croissante. Cette ROC est obenue en choisissant des seuils différents dans l'affectation des classes. En effet, tous les algorithmes présentés ici donnent des probabilité $\hat{y}_i \in [0,1]$ d'appartenance à une classe. Il faut alors choisir un seui s pour affecter la classe définitive. En faisant varier ce seuil, on obtient alors différentes prédictions finales. Pour un classifieur idéal, le taux de faux positifs serait TP=0 et celui de vrais positifs TP=1 (donc une aire sous la courbe $AUC_{opt} = 1$). Soit le point optimal de coordonnées $P_{opt} = (0,1)$ sur la courbe ROC, le seuil optimal est alors donné .donné par le point de la courbe le plus proche de ce point. Nous avons alors trouvé un seuil optimal grâce à cette méthode:

```
404    test_fpr, test_tpr, threholds = roc_curve(Y_test, Y_test_hat)
405    # The best threhold is given by the point of the ROC curve that is the
406    # nearest to the point in the UP-left corner with coodinate (0,1)
407    best_threhold=threholds[np.argmin((1 - test_tpr) ** 2 + test_fpr ** 2)]
408
```

En appliquant cela à l'algorithme de régression logistique (au lieu du seuil par défaut égal à 0.5), les résultats sont très grandement améliorés. Le taux de bonne prédiction atteint 93.4% avec un F1 score de 96.2%. (contre 83.6% et 90.2% précédemment).




```
In [30]: runcell(1, 'C:/Espace étude/EMSE 2A/TB3 Image and pattern Recognition/Projects/
classify_inserts.py')

 Initial cost : 97.73375245895242
[[51  1]
 [ 3  6]]
***************
The accuracy of the Logistic regression classifier is 0.934
***************

***************
The F1-score of the Logistic regression classifier is 0.962
***************
 AUC=0.966

In [31]:
```

DISCUSSIONS:

Les 10 descripteurs morphologiques utilisés pour classifier les lames de coupantes selon qu'elles sont très usées ou peu usées, se sont révélés très efficace. En effet, en optimisant le classifieur utilisé on aboutit à un très faible taux d'erreurs de classement. Cependant, il pourrait être intéressant de rajouter d'autres desripteurs prenant en compte les des ***caractéristiques locales*** des images comme par exemple les **"local binary patterns"** ou encore le gradient ainsi que des notions de **voisinages** et de contour.

II. Détection de cancer de la peau

II.1 Mise en contexte

Dans ce deuxième projet, nous utilisons les techniques vus en analyse d'images pour prédire au mieux la présence de mélanoma (type de cancer de la peau). Celà permettra en particulier aux médecins disposer d'un outil complémentaire de diagnostic et de prévenir le développement de formes sévères.

Nous disposons donc d'une base d'images labélisées selon la présence de melano (label=1) ou non (label=0). De la même manière que dans le premier projet, nous séparons notre base de n=200 données en ensemble d'apprentissage et de test, pour ensuite juger la qualité des classifieurs utilisés.

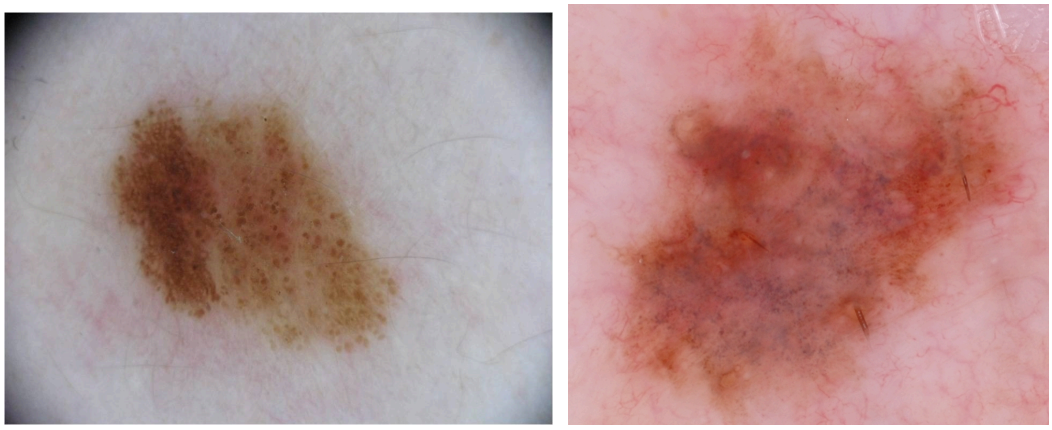


Figure 10: Lésion bénigne VS mélanoma

Comme rappelé plus haut, dans cette partie nous n'insisterons pas beaucoup sur le type de classifieur, mais mettons plutôt l'accent sur la sélection de descripteurs pertinents.

II.2 Sélection des descripteurs et classification

Pour chaque individu, nous disposons d'une image original, de superpixels ainsi que de l'image segmentée. Le type d'image utilisée dépend évidemment de la nature du descripteur choisi.

a. Descripteurs morphologiques

On utilise les images segmentées pour sélectionner les descripteurs morphologiques. Pour commencer, on choisit les 10 descripteurs que le TP précédent (**ShapeFeat)**

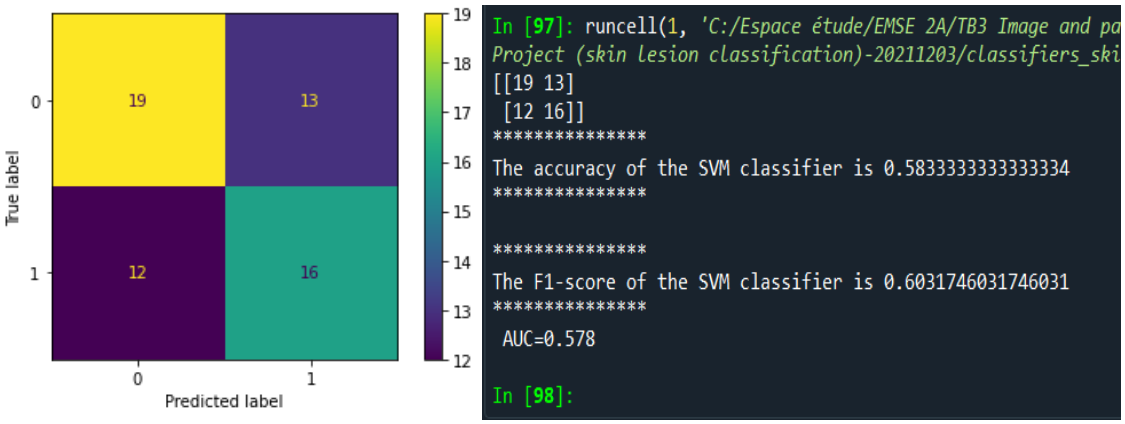


Figure 11: Résultats en utilisant les 10 descripteurs ShapeFeat

Le résultat est alors **très médiocre**, l'**accuracy étant de 58% et l'AUC de 57%.** Ce modèle ne fait pas très largement mieux qu'un modèle naïf qui consisterait à prédire les classes de manière aléatoire. Celà se justifie par le fait que les descripteurs utilisés (périmètre et aire de la région d'intérêt, longueur des axes de l'ellipse circonscrite...) ne concernent que des **caractéristiques assez globales** de l'image et surtout sont des grandeurs **non normalisées** . Ainsi, il est nécessaire d'exploiter des **caractéristiques plus locales** de l'image et aussi de définir des **grandeurs normalisées indépendantes du niveau de zoom des photos.**

Nous choisissons alors certains **descripteurs normalisés** qui se construisent déjà rencontrés en TP. Notons A, P, d, D et z respectivement l'aire, le périmètre, le grand diamètre de Frénet et le périmètre de Crofton. Nous définissons:

$$R_1 = \frac{4\pi A}{P^2}$$

$$e = d/D \ (elongation)$$

$$t = \frac{r}{D} \ (thinness)$$

$$rd = \frac{4 * A}{\pi * D^2} \ (roundness)$$

Nous gardons aussi les grandeurs déjà normalisées comme l'excentricité dans l'ensemble des features de *ShapeFeat et* utilisons *le* ***périmètre de crofton.*** Voici les résultats obtenus:

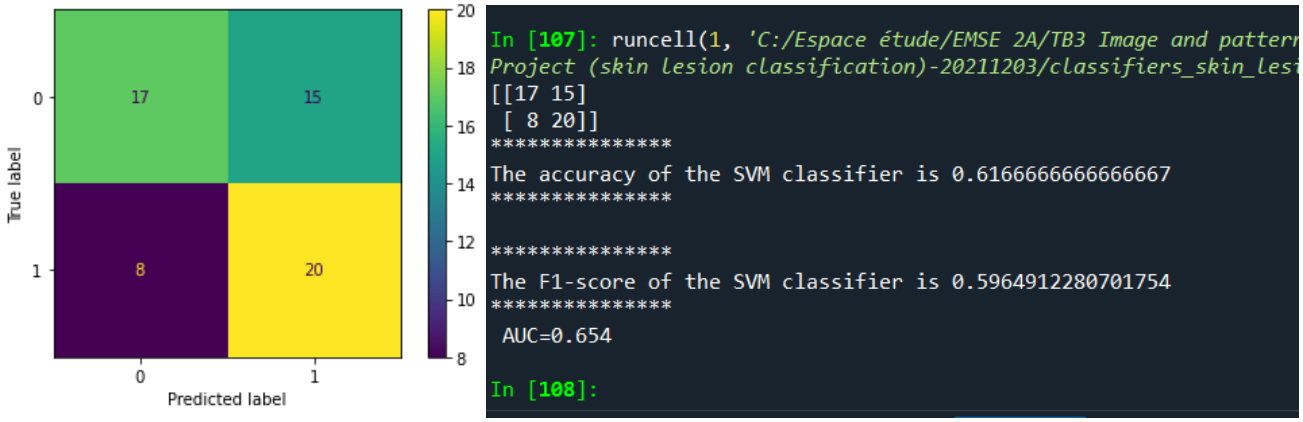


Figure 12: Utilisation de 8 descripteurs morphologiques normalisés

Bien qu'on note une amélioration, le résultat n'est toujours pas très satisfaisant. On se rend compte alors des ***limites*** *d'utiliser des descripteurs morphologiques* uniquement pour la classification de lésions cutanées. Il faut donc utiliser des ***descripteurs d'intensité*** et de texture qui ont l'avantage d'exploiter **des informations locales**.

b. Descripteurs d'intensité et de texture

Nous utilisons dans cette partie les images originales ainsi que les superpixels pour les descripteurs d'intensités. Le premier descripteur de texture concerne les LBP (Local Binary Pattern). Il permet de comparer chaque pixel à l'ensemble de ses 8 voisins comme illustré sur la figure 13 ci-dessous.

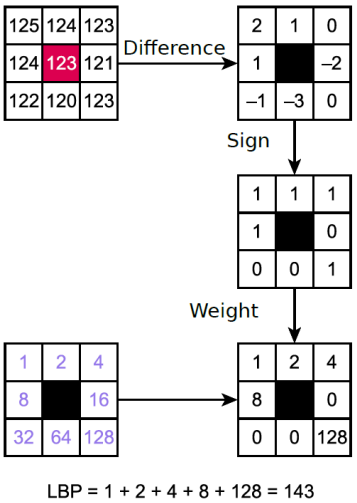


Figure 13.1: Local binary pattern. From wikipedia, author Xiawi, CC-By-SA.

On peut alors générer un ***histogramme de fréquence*** pour chaque image, en comptant la fréquence des valeurs de LBP de ses pixels (le code est le même que celui du TP). Voici un histogramme les histogrammes obtenus pour de cas "confirmé" et "non confirmé" de melanoma.

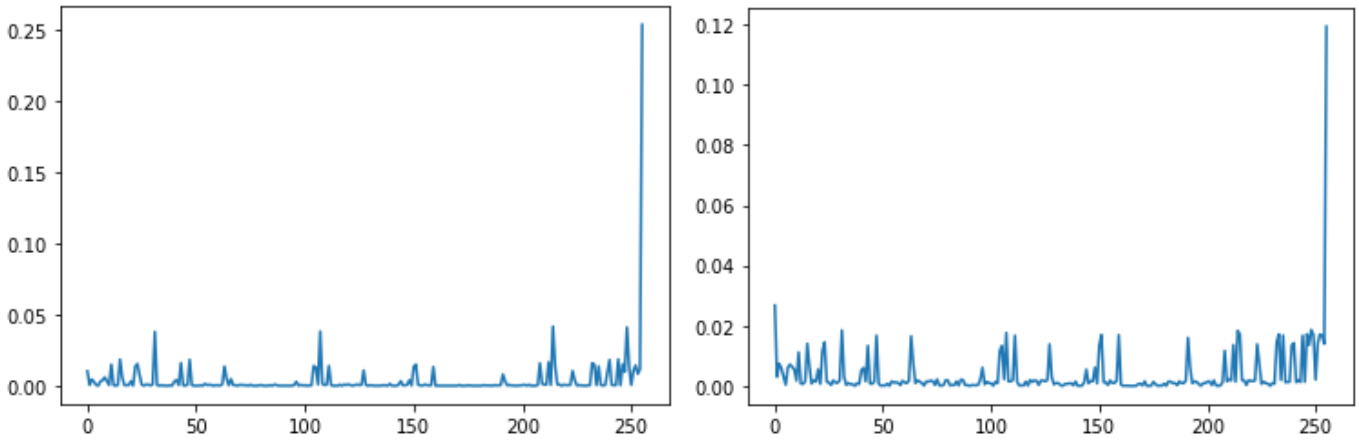


Figure 14: Histogramme LBP d'un cas non confirmé (à gauche) et d'un cas confirmé de melanoma (à droite)

Les deux histogrammes assez distincts permettent d'espérer qu'on pourra bien séparer les deux classes grâce aux LBP.

Toutefois, le problème est qu'on ne peut **ni donner en entrée aux classifieurs les points de l'histogramme** (nombre de features=256 beaucoup trop grand vu le nombre de données n=200), ***ni utiliser la distance entre histogrammes*** comme en TP où l'on utiliser l'algorithme de clustering K-means qui n'utilise que des distances entre individus.

Comme expliqué dans [1], nous choisissons d'extraire que les caractéristiques essentielles de chaque histogramme en utilisant les descripteurs statistiques suivants (on note $P(g)$ l'histogramme de fréquence précédente, c'est donc le pourcentage des pixels qui ont un LBP égal à g):

$$Moyenne : m = \sum_{g=0}^{255} g * P(g)$$

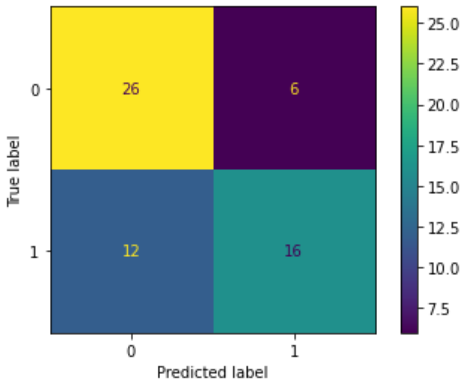
$$Ecart\ type : \sigma = \sum_{g=0}^{255} (g-m)^2 * P(g)$$

$$asymetrie : skew = \frac{1}{\sigma^3} \sum_{i=0}^{255} (g-m)^3 * P(g)$$

$$Energie : E = \sum_{g=0}^{255} P(g)^2$$

$$Entropie : Ent = -\sum_{g=0}^{255} P(g) * log(P(g))$$

Voici les résultats obtenus:



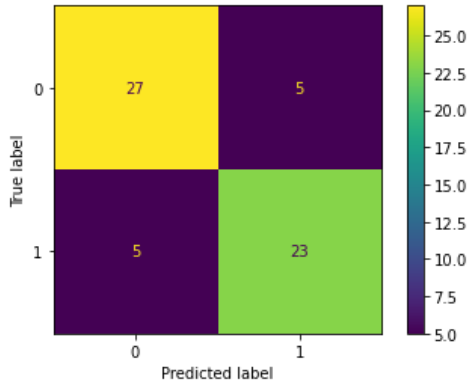

```
In [121]: runcell(1, 'C:/Espace étude/EMSE 2A/TB3 Im
Project (skin lesion classification)-20211203/classi
[[26  6]
 [12 16]]
***************
The accuracy of the SVM classifier is 0.7
***************

***************
The F1-score of the SVM classifier is 0.743
***************
 AUC=0.765

In [122]:
```

Le résultat se trouve très nettement amélioré avec l'utilisation des LBP. On a un taux de 70% de bonne prédictions et plus de 76% d'AUC.

Les 5 valeurs statistiques précédentes (moyenne, écart-type, énergie...) peuvent aussi être directement calculées à partir de l'image d'origine drirectement. En utilisant 2 des 3 composantes de des images d'origine (elles sont en RGB), on obtient l'histogramme $P(g)$ des fréquences de chacun des 256 niveaux de gris (pour plus de détails voir fonction **'histogramProperties(I)'** du fichier '*project_skin_lesion.py*'). Les nouveaux features obtenus sont alors des ***descripteurs d'intensité.***

On <u>combinant</u> ces nouveaux descripteurs avec les précédents on obtient un ensemble de **19 descripteurs** caractérisant aussi bien des ***aspects morphologiques*** *que les* ***variations*** **d'*intensité et de texture.***

Voic le résultat obtenu.




```
In [172]: runcell(1, 'C:/Espace étude/EMSE 2A/TB3 Image and patter
Project (skin lesion classification)-20211203/classifiers_skin_les
[[27  5]
 [ 5 23]]
***************
The accuracy of the SVM classifier is 0.833
***************

***************
The F1-score of the SVM classifier is 0.844
***************
 AUC=0.862

In [173]:
```

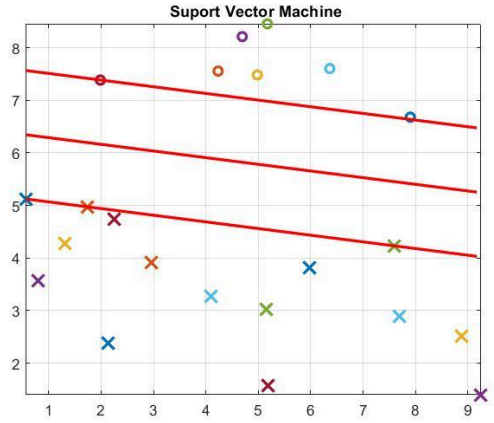
Cette méthode qui combine des descripteurs de plusieurs types donnne **un taux de bonnes prédictions de 83.3%** *largement meilleur que les autres* (plus de 10% de différence). Le F1 score ainsi que l'AUC sont aussi du même ordre (respectivement 84.4% et 86.2%). Nous retenons cette dernière approche comme la méthode finale. Certes on pourrait essayer d'optimiser le seuil mais on est dissuadé par le fait que le nombre de faux positifs et celui de faux négatifs soient égaux (FP=FN=5): un post-traitement n'améliora pas grandement le résultat.

<u>Conclusion:</u>

Ce dernier projet a permis de mettre en valeur la complémentarité des divers types de descripteurs. En effet, détecter un cancer de la peau nécessite de connaitre des informations géométriques sur la zone intérêt mais

aussi des aspects de texture et d'intensité. Pour améliorer de modèle on peut envisager d'augmenter le nombre de données d'apprentissage pour assurer la robustesse des algorithmes. Il pourrait aussi être intéressant d'intégrer d'autres descripteurs issus des opérateurs morphologiques d'ouverture et de fermeture.

ANNEXE: Détail sur les SVM



Dans le cas linéaire, on construit un hyper plan de séparation $\pi = w^T x + b$ de marge maximale où les variables recherchées sont $w \in R^n$ et $b \in R$. La fonction de décision est alors $f(x) = w^T x + b$. La distance d'un point à l'hyperlan est alors $d(x_k, \pi) = \frac{|W^T x_k + b|}{||W||} = \frac{l_k(W^T x_k + b)}{||W||}$**. Les points vérifiant** $l_k(W^T x_k + b) > 1$ sont situés hors de la marge de largeur $\frac{2}{||w||}$ que l'on veut maximiser. Ce qui est équivlent à minimiser ||w|| ou encore $\frac{1}{2}||w||^2$. On en déduit la formulation primale du problème:

$$Min(\frac{1}{2}||w||^2) \; avec \; \forall k \; 1 - l_k(w^T x_k + b) \leq 0$$

Le lagrangien associé est alors:

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 + \sum_{k=1}^{p} \alpha_k (1 - l_k(w^T x_k + b))$$

A l'optimim, l'annulation du gradient permet de déduire une formulation duale du problème:

$$\begin{cases} Max(H(\alpha)) \\ \sum_{k=1}^{p} l_k \alpha_k = 0 \\ \alpha_k \geq 0 \end{cases}$$

où $H(\alpha) = -\frac{1}{2}\alpha^T A \alpha + < u, \alpha >$ avec $u = (1, ..., 1)^T$ et $[A]_{i,j} = l_i l_j < x_i, x_j >$

A étant symétrique défini positif. En effet, on a:

$$\forall\, v \in R^n, on\ a\ v^T A v = \sum_{i,j} v_i v_j < x_i, x_j >= ||\widehat{v}||^2 > 0 \quad \text{avec } \widehat{v} = \sum_{i=1}^{p} v_i l_i x_i$$

Comme la matrice hessience est $Hess(H) = -A$, on en déduit que H est une forme quadratique concave (d'où l'existence de son maximum). On en conclut l'***existence de la solution*.** De plus, l'ensemble des contraintes est convexe (cadran de R^n) d'où l'***unicité***. Le problème est donc **bien posé.**

L'algorithme du gradient conjugué permet alors de trouver le point selle α^*

Référence:

[1] https://www.researchgate.net/publication/4322456_Color_histogram_features_based_image_classification_in_content-based_image_retrieval_systems