Test technique réalisé par : HARTI Fatima

1- Problématique :

Ce projet est un problème de classification à travers lequel nous voulons classifier les professions de certains clients. Pour ce nous disposons de trois jeux de données :

- Clients : ce jeu de donnée contient les id des clients, leurs professions et leurs anciennetés.
- Facturation : contient l'id du client, le produit acheté et le temps de l'achat.
- Clients_to_complete : contient des id des clients dont on veut prédire la profession.

2- Préparation des données :

Pour prédire la profession de nos clients nous avons appliqué un « left join » entre les deux data sets « Facturation » et « Clients ». Nous avons ainsi retrouvé les id des clients pour qui on veut prédire la profession ainsi que leurs achats. La variable ancienneté est manquante pour tous ces clients, nous étions donc obligés de la supprimer (en plus elle paraît non informative).

Après avoir supprimé ces variables, il y avait encore 53 valeurs manquantes en produit_id. Nous avons choisi de supprimer ces lignes.

La variable dépendante « Profession » contient cinq catégorie de classe à savoir : 'Menuisier', 'Plâtrier', 'Maçon', 'Plombier', 'Couvreur'. Ces labels ont été encodés.

La variable produit_id contient quant à elle 102. Nous avons convertit cette dernière en une matrice 0/1 à l'aide de la méthode « oneHotEncoding ». On pense pouvoir reconnaître les métiers des clients en fonctions de leurs paniers d'achats.

Ensuite, le jeu de données a été divisé en deux parties, dans la première « data_to_use » on dispose des valeurs de la profession et dans l'autre « null_professions » elles sont manquantes. Nous avons utilisé ensuite un « inner join » entre les deux data « null_professions » et « clients_to_complete ».

Notre travail n'est pas encore terminé, car quand on voudra prédire la profession d'un certain client on ne pourra pas donner tous les vecteurs contenant un produit acheté par ce client à l'algorithme. Pour ce, Nous avons fait une sommation « sum » en regroupant « group by client_id , profession» selon la clé « client_id » et la « profession » , ensuite on a affecté 1 au produits où la valeur est supérieur à 1. Nous avons remplacé les valeurs supérieures à un par un, car on s'intéresse plutôt au panier acheté et non à la quantité.

Nous avons obtenu pour chaque client, un vecteur représentant ses achats des produits. Soit un nombre de variables de 103 dont la variable dépendante « Profession » et la variable « client_id » que l'on va supprimer lors l'apprentissage des modèles. L'id client a été supprimé car elle n'identifie pas la variable classe.

En somme, nous disposons maintenant de 4764 observations dans la data_to_use dont on donne un aperçu ci-dessous :

	client_id	Profession	product_id_21.0	product_id_12.0	product_id_3.0	product_id_90.0	produc
0	0	3.0	0.0	0.0	0.0	0.0	
1	1	3.0	0.0	0.0	0.0	0.0	
2	2	1.0	0.0	0.0	0.0	0.0	
3	3	1.0	0.0	0.0	0.0	1.0	
4	8589934593	2.0	0.0	0.0	0.0	1.0	

5 rows × 103 columns

Figure 1 : Extrait du jeu de données, source : Fait par l'auteur.

3- Déséquilibre du jeu de données – Split data :

Nous avons a séparé la data set en deux échantillons : train data avec un pourcentage de 75 % et 25% pour test data.

L'observation des pourcentages de chaque catégorie de classe a montré que notre jeu de données n'est pas équilibré. Ainsi certaines classes sont mal représentées par rapport aux autres.

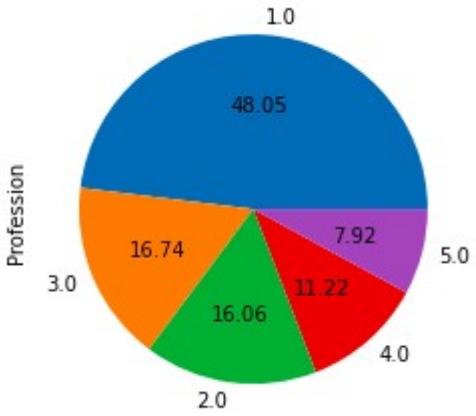

Figure 2 : Répartition du train data selon la profession, source : Fait par l'auteur

Pour remédier à ce problème, nous avons appliqué deux approches :

- Balancer la data en utilisant une combinaison entre les deux méthodes **« resampling and downsampling »**
- Utiliser des algorithmes qui s'adaptent à ce type de problème comme le **RandomForest**.

Pour chaque modèle, nous avons testé les deux approches.

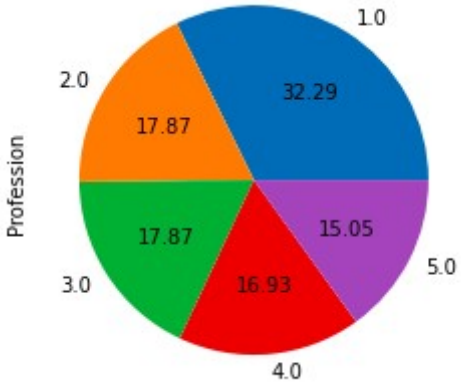

Figure 3 : Répartition de train data équilibrée

Pour comparer entre les différents modèles on s'est basé sur des métriques plus adéquates à ce type de problème comme le « Recall». Ces métriques ont été estimées par la méthode **« micro-average »**, qui est plus adéquate dans le cas d'un jeu de données non équilibré.

4-Apprentissage des modèles :

1. Modèle SVM :

Nous avons appliqué le modèle SVM en utilisant trois types de Kernels : linéaire, polynomiale et rbf.
Le modèle SVM linéaire s'avère plus performant avec une valeur en F1-score =0.603 contre 0.578 pour le SVM polynomial et 0.477 pour le rbf.
Les scores estimées sur la data équilibrée ont été plus faibles pour les trois modèles.

2. Modèle Random Forest

Pour le Random forest nous avons appliqué les deux méthodes **« RandomizedSearchCV »** et **« GridSearchCV »** qui cherche à optimiser les paramètres sur une grille de paramètres donnés.
Le modèle optimal a les paramètres suivants :

{'n_estimators': 1000, 'max_features': 'sqrt', 'max_depth': 20, 'criterion': 'gini'}

Le modèle s'avère moins performant que le SVM (un F1-score =0.576).

3. Modèle XgBoost

Nous avons appliqué pour le modèle Xgboost la recherche par grille « RandomizedSearchCV » pour trouver les paramètres optimaux. Nous avons ainsi trouvé :

{'min_child_weight': 3, 'max_depth': 3, 'learning_rate': 0.3, 'colsample_bytree': 0.4}

Le modèles s'avère moins performant aussi (F1-mesure=0.559).

5- Résultats des modèles et comparaison :

Les résultats des modèles après optimisation des paramètres sont donnés dans le tableau suivant :

	Modèle	SVM	RandomForest	XGboost
Data non équilibrée	Accuracy	0.603	0.576	0.559
	F1_score	0.603	0.576	0.559
	Recall	0.603	0.576	0.559
	precision	0.603	0.576	0.559
Data équilibrée	Accuracy	0.561	0.566	0.567
	F1_score	0.561	0.566	0.567
	Recall	0.561	0.566	0.567
	precision	0.561	0.566	0.567

Tableau1 : métriques selon le modèle, source : Fait par l'auteur.

Le modèle SVM appliqué sur la data sans modification s'avère plus performant.

En regardant la précision (micro-average) (60.3%) ceci signifie que le modèle SVM est plus performant en termes de précision (proportion des prédictions qui sont réellement correctes). En Recall aussi le SVM est meilleur (proportion des labels réels qui ont été prédits correctement). Le F1_score appuie aussi l'hypothèse que le modèle SVM est le meilleur.

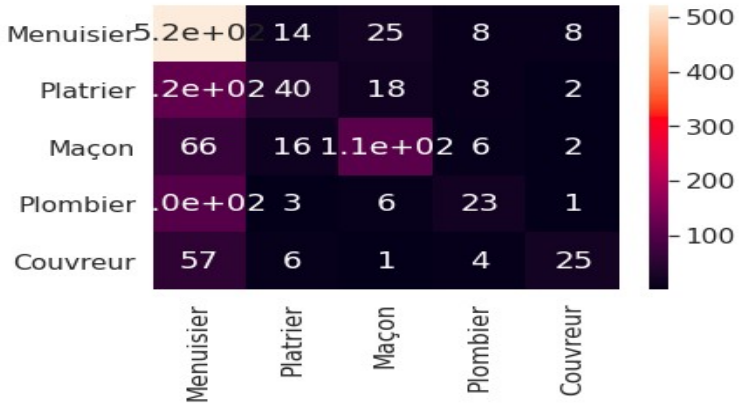

Figure 4 : Matrice de confusion du modèle SVM

5-Prédictions :

Les prédictions ont été faites en utilisant le modèle optimal trouvé auparavant, après avoir faire un « inner join » entre les tables clients_to_complete et null_professions.