Université mohammed premier oujda

Ecole Nationale des sciences appliquée oujda

Projet de Machine Learning

Prédiction de la maladie de Parkinson en Python

Elaboré par :

Ilyas RIAH
Mohammed KHATTALA
Moussa MOHAMED MOUBARAK
Said OUBAASSINE

Encadré par :

Toumi Bouchentouf
Haja Zakaria

Tables des matières

1. Introduction

1-1. Qu'est ce que c'est le Machine Learning ?

Le Machine Learning ou apprentissage automatique est un domaine scientifique, et plus particulièrement une sous-catégorie de l'intelligence artificielle. Elle consiste à laisser des algorithmes découvrir des " patterns ", à savoir des motifs récurrents, dans les ensembles de données. Ces données peuvent être des chiffres, des mots, des images, des statistiques…

Pour résumer, les algorithmes de Machine Learning apprennent de manière autonome à effectuer une tâche ou à réaliser des prédictions à partir de données et améliorent leurs performances au fil du temps. Une fois entraîné, l'algorithme pourra retrouver les patterns dans de nouvelles données.

1-2. Comment fonctionne le Machine Learning ?

Le développement d'un modèle de Machine Learning repose sur quatre étapes principales. En règle générale, c'est un Data Scientist qui gère et supervise ce procédé.

La première étape consiste à sélectionner et à préparer un ensemble de données d'entraînement. Ces données seront utilisées pour nourrir le modèle de Machine Learning pour apprendre à résoudre le problème pour lequel il est conçu.

Les données peuvent être étiquetées, afin d'indiquer au modèle les caractéristiques qu'il devra identifier. Elles peuvent aussi être non étiquetées, et le modèle devra repérer et extraire les caractéristiques récurrentes de lui-même.

La deuxième étape consiste à sélectionner un algorithme à exécuter sur l'ensemble de données d'entraînement. Le type d'algorithme à utiliser dépend du type et du volume de données d'entraînement et du type de problème à résoudre.

La troisième étape est l'entraînement de l'algorithme. Il s'agit d'un processus itératif. Des variables sont exécutées à travers l'algorithme, et les résultats sont comparés avec ceux qu'il aurait dû produire. Les " poids " et le biais peuvent ensuite être ajustés pour accroître la précision du résultat.

La quatrième et dernière étape est l'utilisation et l'amélioration du modèle. On utilise le modèle sur de nouvelles données, dont la provenance dépend du problème à résoudre.Par exemple, un modèle de Machine Learning conçu pour détecter les spams sera utilisé sur des emails.

1-3. Quels sont les différents types de Machine Learning ?

On distingue trois techniques de Machine Learning : l'apprentissage supervisé, l'apprentissage non-supervisé, et l'apprentissage par renforcement. Dans le cas de l'apprentissage supervisé, le plus courant, les données sont étiquetées afin d'indiquer à la machine quelles patterns elle doit rechercher.

Le système s'entraîne sur un ensemble de données étiquetées, avec les informations qu'il est censé déterminer. Les données peuvent même être déjà classifiées de la manière dont le système est supposé le faire.

Au contraire, dans le cas de l'apprentissage non supervisé, les données n'ont pas d'étiquettes. La machine se contente d'explorer les données à la recherche d'éventuelles patterns. Elle ingère de vastes quantités de données, et utilise des algorithmes pour en extraire des caractéristiques pertinentes requises pour étiqueter, trier et classifier les données en temps réel sans intervention humaine.

Plutôt que d'automatiser les décisions et les prédictions, cette approche permet d'identifier les patterns et les relations que les humains risquent de ne pas identifier dans les données. Cette technique n'est pas très populaire, car moins simple à appliquer. Elle est toutefois de plus en plus populaire dans le domaine de la cybersécurité.

Enfin, l'apprentissage par renforcement consiste à laisser un algorithme apprendre de ses erreurs pour atteindre un objectif. L'algorithme essayera de nombreuses approches différentes pour tenter d'atteindre son but.

En fonction de ses performances, il sera récompensé ou pénalisé pour l'inciter à poursuivre dans une voie ou à changer d'approche. Cette technique est notamment utilisée pour permettre à une IA de surpasser les humains dans les jeux.

Par exemple, AlphaGo de Google a battu le champion de Go grâce à l'apprentissage par renforcement. De même, OpenAI a entraîné une IA capable de vaincre les meilleurs joueurs du jeu vidéo Dota 2.

1-4. Problématique

La maladie de Parkinson est une affection dégénérative, lentement évolutive, caractérisée par des tremblements de repos, une rigidité musculaire, des mouvements lents et diminués (bradykinésie) et finalement une instabilité de la démarche et/ou posturale. Le diagnostic est clinique.

La maladie de Parkinson touche environ

- 0,4% des personnes de > 40 ans
- 1% des personnes de ≥ 65 ans
- 10% des personnes ≥ 80 ans

L'âge moyen au moment du début est d'environ 57 ans.

Le but de notre projet est de pouvoir prédire si un patient est atteint de la maladie de Parkinson.

2. Méthodologie

2-1. Présentation de Data set

Ensemble de données sur la maladie de Parkinson Cet ensemble de données est composé d'une gamme de mesures biomédicales de la voix de 31 personnes, dont 23 atteintes de la maladie de Parkinson (MP). Chaque colonne du tableau est une mesure de voix particulière, et chaque ligne correspond à l'un des 195 enregistrements vocaux de ces individus (colonne "nom"). L'objectif principal des données est de discriminer les personnes en bonne santé de celles atteintes de MP, selon la colonne "statut" qui est fixée à 0 pour sain et 1 pour MP.

Les données sont au format ASCII CSV. Les lignes du fichier CSV contiennent une instance correspondant à un enregistrement vocal. Il y a environ six enregistrements par patient, le nom du patient est identifié dans la première colonne.

Les colonnes de data set :
Nom - nom du sujet ASCII et numéro d'enregistrement.
MDVP:Fo(Hz) - Fréquence fondamentale vocale moyenne.
MDVP:Fhi(Hz) - Fréquence fondamentale vocale maximale.
MDVP:Flo(Hz) - Fréquence fondamentale vocale minimale.
MDVP:Jitter(%), MDVP:Jitter(Abs), MDVP:RAP, MDVP:PPQ, Jitter:DDP - Plusieurs mesures de variation de la fréquence fondamentale.
MDVP : Shimmer, MDVP : Shimmer (dB), Shimmer : APQ3, Shimmer : APQ5, MDVP : APQ, Shimmer : DDA - Plusieurs mesures de variation d'amplitude.
NHR, HNR - Deux mesures du rapport entre le bruit et les composantes tonales de la voix.
Statut - L'état de santé du sujet (un) - Parkinson, (zéro) - en bonne santé.
RPDE, D2 - Deux mesures de complexité dynamique non linéaire.
DFA - Exposant de mise à l'échelle fractale du signal.
Spread1, Spread2 ,PPE - Trois mesures non linéaires de variation de fréquence fondamentale

2-1. Collecte et analyse des données

La variable 'status' est la variable de sortie. Il s'agit d'une variable entière binaire, 0 correspondant à une personne en bonne santé et 1 à une personne atteinte de la maladie de Parkinson.

```
#nom des colonnes de dataset
data.columns
```

```
Index(['name', 'MDVP:Fo(Hz)', 'MDVP:Fhi(Hz)', 'MDVP:Flo(Hz)', 'MDVP:Jitter(%)',
       'MDVP:Jitter(Abs)', 'MDVP:RAP', 'MDVP:PPQ', 'Jitter:DDP',
       'MDVP:Shimmer', 'MDVP:Shimmer(dB)', 'Shimmer:APQ3', 'Shimmer:APQ5',
       'MDVP:APQ', 'Shimmer:DDA', 'NHR', 'HNR', 'status', 'RPDE', 'DFA',
       'spread1', 'spread2', 'D2', 'PPE'],
      dtype='object')
```

Les autres variables sont des variables prédictives, toutes sauf 'name' sont de type numérique, 'name' est de type caractère.

```
# Le type des données
data.dtypes
```

```
name                object
MDVP:Fo(Hz)        float64
MDVP:Fhi(Hz)       float64
MDVP:Flo(Hz)       float64
MDVP:Jitter(%)     float64
MDVP:Jitter(Abs)   float64
MDVP:RAP           float64
MDVP:PPQ           float64
Jitter:DDP         float64
MDVP:Shimmer       float64
MDVP:Shimmer(dB)   float64
Shimmer:APQ3       float64
Shimmer:APQ5       float64
MDVP:APQ           float64
Shimmer:DDA        float64
NHR                float64
HNR                float64
status               int64
RPDE               float64
DFA                float64
spread1            float64
spread2            float64
D2                 float64
PPE                float64
dtype: object
```

La variable 'name' semble inutile, car elle enregistre le nom du patient avec le numéro d'enregistrement, qui ne semble pas avoir de corrélation avec la cible.

Il n'y a aucune valeur manquante dans aucune observation.

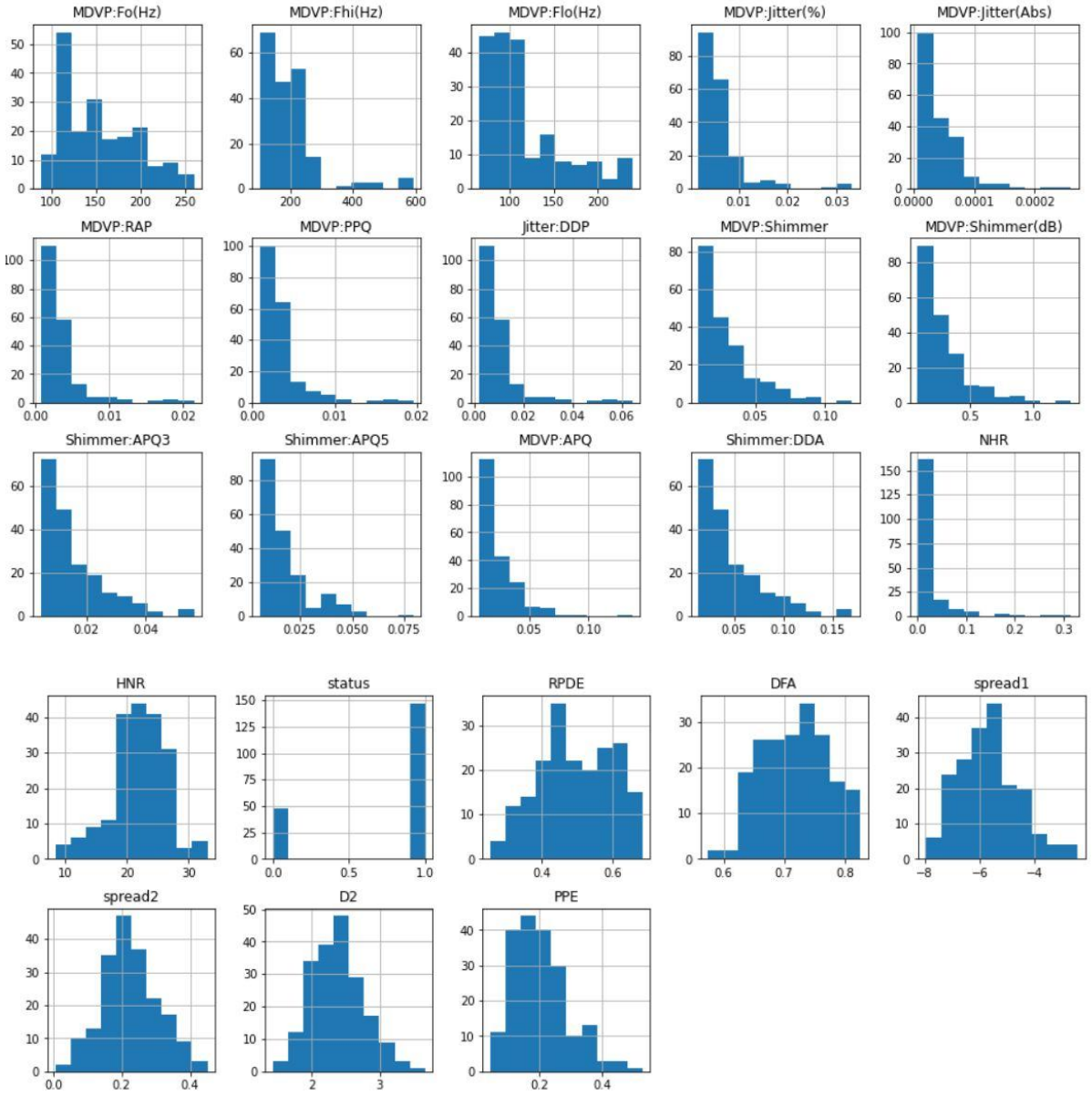
```
# Check NA values
data.isnull().sum().sum()
```

```
0
```

2-1-1. Visualisation des données

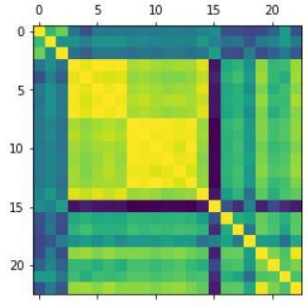
Visualisations univariées (histogrammes des attributs) :

```
data.hist(figsize=(16,16))
```



Visualisations multivariées (corrélation) :

```
# Correlation
plt.matshow(data.corr())
plt.show()
```



De l'analyse des données exploratoires ci-dessus, nous pouvons conclure que :

1. Le statut est la variable dépendante, qui est de la classe entière.

2. Toutes les variables indépendantes ont une variance acceptable

3. Plusieurs variables indépendantes sont fortement corrélées entre elles.

4. Il n'y a pas de valeurs nulles dans l'ensemble de données.

2-2. Préparation des données

Nous savons qu'il n'y a pas de valeurs manquantes dans notre jeu de données.

```
# Verification des attributs manquants
data.isna().sum()
```

```
name              0
MDVP:Fo(Hz)       0
MDVP:Fhi(Hz)      0
MDVP:Flo(Hz)      0
MDVP:Jitter(%)    0
MDVP:Jitter(Abs)  0
MDVP:RAP          0
MDVP:PPQ          0
Jitter:DDP        0
MDVP:Shimmer      0
MDVP:Shimmer(dB)  0
Shimmer:APQ3      0
Shimmer:APQ5      0
MDVP:APQ          0
Shimmer:DDA       0
NHR               0
HNR               0
status            0
RPDE              0
DFA               0
spread1           0
spread2           0
D2                0
PPE               0
dtype: int64
```

Nous allons vérifier les variables à très faible variance et envisager de les supprimer.

- Sélection des caractéristiques

```
#Avec la fonction suivante, nous pouvons sélectionner des caractéristiques hautement corrélées
#il supprimera la première caractéristique qui est corrélée avec toute autre caractéristique

def correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr
```

- Trouver des variables indépendantes fortement corrélées

```
corr_features
```

```
{'Jitter:DDP',
 'MDVP:APQ',
 'MDVP:Jitter(Abs)',
 'MDVP:PPQ',
 'MDVP:RAP',
 'MDVP:Shimmer(dB)',
 'NHR',
 'PPE',
 'Shimmer:APQ3',
 'Shimmer:APQ5',
 'Shimmer:DDA'}
```

- Supprimer les variables indépendantes fortement corrélées

```
features = features.drop(corr_features, axis=1)
```

- Découpage du dataset en trainset et testset

```
X_train, X_test, Y_train, Y_test = train_test_split(features, target, test_size=0.2, random_state=2)
```

```
print(X_train.shape, X_test.shape)
```

```
(156, 11) (39, 11)
```

3. Modélisation

A partir de l'analyse de la problème on se rencontre qu'on est face à un problème de classification. Qui est de savoir si un patient est atteint ou non de la maladie.

Ainsi nous allons nous intéresser à quelques algorithmes de classification et trouver lequel qui nous donnera un meilleure score.

3-1. XGBoost

XGBoost (pour contraction de eXtreme Gradient Boosting) est un algorithme d'apprentissage supervisé dont le principe est de combiner les résultats d'un ensemble de modèles plus simples et plus faibles afin de fournir une meilleure prédiction. Il est issu de la famille des algorithmes de **Ensemble Learning**. Il travaille de manière séquentielle en améliorant le modèle par rapport aux exécutions précédentes ce qui permet de corriger les erreurs en affectant des poids aux erreurs. *Son fonctionnement est le suivant* : Il commence donc par construire un premier modèle qu'il va évaluer. A partir de cette première évaluation il affecte des poids en fonction de la performance du modèle.

Pour notre cas, l'utilisation de cet algorithme nous a donné un score de **84,62%.**

3.2. Arbre de décision

Arbre de décision étant aussi de la famille de **Ensemble Learning**, il se base sur une méthode ensembliste parallèle : c'est -à -dire il fait la moyenne des différents modèles ce qui n'est pas le cas avec **XGBoost** qui est itératif.

Pour notre cas, le modèle a prédit un score de **79,49%.**

3.3. Régression logistique

La régression logistique est une méthode très utilisée car elle permet de modéliser des variables binomiales, ce qui cadre avec notre problème de prédiction. Le score du modèle est **76,92%.**

3.4. K Plus proche voisin

Le k plus proche voisin (ou *k-Nearest Neighbors)* un algorithme classique de classification ayant fait ses preuves. Pour notre cas, l'utilisation de cet algorithme nous a donné un score de **82,05%.**

Tableau récapitulatif des scores obtenus

Modèle	Précision	
	Découpage du dataset dont 80% train et 20% test	**Validation croisée avec K Fold=5**
XGBoost	84,62%	[0.84375 , 0.87096774, 0.93548387, 0.87096774, 0.93548387]
Arbre de décision	79,49%	[0.8125 , 0.87096774, 0.96774194, 0.80645161, 0.87096774]
Régression logistique	76,92%	[0.8125 , 0.83870968, 0.87096774, 0.80645161, 0.83870968]
K plus proche voisin	82,05%	[0.875 , 0.96774194, 0.93548387, 0.93548387, 0.87096774]

4. Conclusion

Ce travail a pour objectif de pouvoir prédire si un patient est atteint de la maladie de parkinson. Pour cela nous avons appliqué les différents modèles de machine learning, et nous avons conclu que les modèles de classification donnent un score supérieur.

Ce mémoire nous a permis d'acquérir une expérience formidable, il nous a été l'occasion d'améliorer nos connaissances théorique et pratique. comme il nous a été l'occasion de se familiariser d'avantages au travail en groupe.