

TRAVAUX PRATIQUES N°4

Détection de la Maladie de Parkinson par Apprentissage Automatique



Classification avec Naive Bayes Gaussien



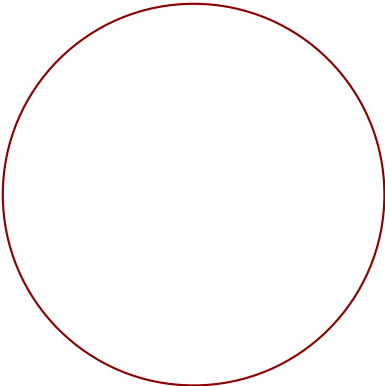
Centre Universitaire de Mila



Master 1 (STIC & I2A)



Apprentissage Automatique



Plan du Cours

1 Introduction à la Maladie de Parkinson

Contexte médical et approche par analyse de la voix

3 Les Attributs Vocaux

Description des 23 caractéristiques extraites

5 Validation Croisée

Évaluation robuste du modèle

2 Présentation du Dataset

Caractéristiques et structure des données

4 Théorie: Naive Bayes Gaussien

Principe et formulation mathématique

6 Implémentation Python

Code complet et étapes de traitement

I

Introduction à la Maladie de Parkinson

Contexte médical et approche par analyse de la voix

Contexte et Objectifs

La Maladie de Parkinson

Maladie neurodégénérative progressive affectant le système nerveux central. Caractérisée par la dégénérescence des neurones dopaminergiques dans la substance noire.

Symptômes principaux: tremblements, rigidité musculaire, bradykinésie, troubles de l'équilibre.

Analyse de la Voix

La maladie affecte le contrôle moteur des cordes vocales, entraînant des changements mesurables dans la fréquence fondamentale, le jitter, le shimmer et d'autres paramètres acoustiques.

Objectif Principal

Discriminer les personnes en bonne santé de celles atteintes de la maladie de Parkinson en utilisant des mesures biomédicales de la voix.



Personne saine



Maladie de Parkinson

Intérêt du Diagnostic Précoce

- ✓ Permet un traitement précoce et plus efficace
- ✓ Méthode non invasive et peu coûteuse
- ✓ Accessible à grande échelle via téléphonie

2

Présentation du Dataset

Caractéristiques et structure des données

Informations sur l'Ensemble de Données

Origine des Données

Dataset créé par **Max Little** de l'Université d'Oxford, en collaboration avec le **National Center for Voice and Speech** de Denver, Colorado.

Les signaux vocaux ont été enregistrés et analysés pour extraire des caractéristiques pertinentes pour les troubles de la voix.

31

Personnes

195

Enregistrements

24

Colonnes

Structure du Dataset

Types de données:

- 22 colonnes: float64
- 1 colonne: int64 (status)
- 1 colonne: object (name)

Taille mémoire:

- 195 entrées
- 36.7+ KB
- Pas de valeurs manquantes

Distribution des Classes

Maladie de Parkinson (1) 147



Sain (0) 48



⚠ Déséquilibre de Classes

Le dataset présente un déséquilibre significatif avec 75.4% de cas malades.

Solution: Utilisation de SMOTE pour suréchantillonner la classe minoritaire.

3

Les Attributs Vaux

Description des 23 caractéristiques extraites

Catégories de Mesures Vocales

I. Fréquence Fondamentale

MDVP:Fo(Hz)	Fréquence fondamentale moyenne
MDVP:Fhi(Hz)	Fréquence fondamentale maximale
MDVP:Flo(Hz)	Fréquence fondamentale minimale

2. Variation de Fréquence (Jitter)

MDVP:Jitter(%)	Variation relative de la période
MDVP:Jitter(Abs)	Variation absolue de la période
MDVP:RAP, PPQ, DDP	Mesures de variation de fréquence

3. Variation d’Amplitude (Shimmer)

MDVP:Shimmer r	Variation relative d’amplitude
MDVP:Shimmer(dB))	Variation en décibels
APQ3, APQ5, APQ, DDA	Mesures de variation d’amplitude

4. Rapport Bruit/Tonalité

NHR R	Noise-to-Harmonics Ratio
HNR R	Harmonics-to-Noise Ratio

Note: NHR élevé et HNR faible indiquent une voix plus bruitée.

5. Mesures Non Linéaires

RPDE Complexité dynamique	D2 Dimension fractale	DFA Scaling exponent	spread1, spread2 Variations non linéaires	PPE Periodicity entropy
-------------------------------------	---------------------------------	--------------------------------	---	-----------------------------------

4

Théorie: Naive Bayes Gaussien

Principe et formulation mathématique

Principe du Classificateur Naive Bayes

🔄 Théorème de Bayes

Le classificateur Naive Bayes est basé sur le **théorème de Bayes** appliqué à la classification probabiliste.

$$P(C|X) = P(X|C) \cdot P(C) / P(X)$$

$P(C|X)$: Probabilité a posteriori de la classe C

$P(X|C)$: Vraisemblance des données

$P(C)$: Probabilité a priori de la classe

$P(X)$: Probabilité marginale des données

🔄 Hypothèse d'Indépendance

L'hypothèse "naive" suppose que **toutes les caractéristiques sont conditionnellement indépendantes** étant donné la classe.

$$P(X|C) = P(x_1|C) \cdot P(x_2|C) \cdot \dots \cdot P(x_n|C)$$

📈 Naive Bayes Gaussien

Pour les données continues, on suppose que les caractéristiques suivent une **distribution normale (gaussienne)** conditionnellement à chaque classe.

$$P(x_i|C) = (1/\sqrt{2\pi\sigma^2}) \cdot \exp(-(x_i-\mu)^2/(2\sigma^2))$$

μ : Moyenne de la caractéristique pour la classe C

σ : Écart-type de la caractéristique pour la classe C

💡 Pourquoi GaussianNB?

- ✓ Adapté aux caractéristiques continues (mesures vocales)
- ✓ Très rapide à entraîner et prédire
- ✓ Fonctionne bien avec peu de données
- ✓ Peu de paramètres à régler

GaussianNB dans Scikit-learn

🔗 Importation

```
# Import de GaussianNB
from sklearn.naive_bayes import GaussianNB
# Instanciation
gnb = GaussianNB()
```

⚙️ Paramètres Principaux

priors

Probabilités a priori des classes. Si None, calculées à partir des données.

var_smoothing

Portion de la variance ajoutée pour la stabilité numérique. Défaut: 1e-9

🔗 Méthodes Principales

<code>fit(X, y)</code>	Entraîne le modèle sur les données
<code>predict(X)</code>	Prédit les classes
<code>predict_proba(X)</code>	Probabilités de chaque classe
<code>score(X, y)</code>	Retourne la précision moyenne
<code>partial_fit(X, y)</code>	Apprentissage incrémental

★ Avantages de GaussianNB

- ⚡ **Rapide:** Entraînement et prédiction très rapides
- ⚙️ **Simple:** Peu de paramètres à régler
- 📊 **Efficace:** Fonctionne bien avec des données continues
- 📄 **Peu de données:** Performant même avec peu d'exemples

5

Validation Croisée

Évaluation robuste du modèle

Principe de la Validation Croisée

✂ K-Fold Cross-Validation

Technique d'évaluation qui divise les données en **k sous-ensembles (folds)** de taille égale.

Procédure:

1. Diviser les données en k folds
2. Entraîner sur k-1 folds
3. Tester sur le fold restant
4. Répéter k fois (chaque fold utilisé pour le test)
5. Calculer la moyenne des performances

📊 Calcul des Performances

Moyenne: $\mu = (1/k) \cdot \sum(\text{score}_i)$

Écart-type: $\sigma = \sqrt{[\sum(\text{score}_i - \mu)^2 / k]}$

Intervalle de confiance: $\mu \pm 2\sigma$

🎲 Illustration: 5-Fold Cross-Validation

Fold 1:	TEST	Train	Train	Train	Train
Fold 2:	Train	TEST	Train	Train	Train
Fold 3:	Train	Train	TEST	Train	Train
Fold 4:	Train	Train	Train	TEST	Train
Fold 5:	Train	Train	Train	Train	TEST

✓ Avantages

- **Utilisation optimale:** Toutes les données utilisées pour entraîner et tester
- **Estimation fiable:** Réduit la variance de l'estimation
- **Détection surapprentissage:** Variance élevée = surapprentissage

6

Implémentation Python

Code complet et étapes de traitement

Étape 1: Chargement et Exploration

```
# Import des bibliothèques
import pandas as pd
import numpy as np
# Chargement des données
data = pd.read_csv('parkinsons.csv')
# Affichage des 5 premières lignes
print(data.head())
# Informations sur le dataset
print(data.info())
# Distribution de la cible
print(data['status'].value_counts())
```

📊 Résultat: data.head()

name	MDVP:Fo	MDVP:Fhi	MDVP:Flo	MDVP:Jit(%)	status
phon_R01_S01	119.992	157.302	74.997	0.00784	1
phon_R01_S01_2	122.400	148.650	113.819	0.00968	1

[5 rows x 24 columns]

📊 Résultat: data.info()

RangeIndex: 195 entries

Data columns: 24 columns

Dtypes: float64(22), int64(1), object(1)

Memory usage: 36.7+ KB

📊 Distribution des Classes

Maladie	147
Sain	(48.4%)
(o):	(24.6%)

Étape 2: Prétraitement des Données

```
# Suppression de la colonne 'name'
data.drop(['name'], axis=1, inplace=True)

# Séparation features et target
X = data.drop('status', axis=1).values
y = data['status'].values
```

Gestion du Déséquilibre: SMOTE

SMOTE (Synthetic Minority Over-sampling Technique) génère des exemples synthétiques pour équilibrer les classes.

```
from imblearn.over_sampling import SMOTE
SMOTE
oversample = SMOTE()
X, y = oversample.fit_resample(X, y)

print(pd.Series(y).value_counts())
```

Standardisation

La **standardisation** transforme les données pour avoir une moyenne de 0 et un écart-type de 1.

```
from sklearn.preprocessing import StandardScaler
StandardScaler
# Instanciation et transformation
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

Formule: $z = (x - \mu) / \sigma$

Résumé du Prétraitement

- ✓ Suppression colonne non informative
- ✓ Séparation X (features) et y (target)
- ✓ Standardisation des features
- ✓ Équilibrage avec SMOTE (147/147)

6.3 Code Étape 3: Entraînement du Modèle

Division Train/Test

```
# Import de train_test_split
from sklearn.model_selection import train_test_split

# Division 90% train / 10% test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.1, random_state=42)
```

`test_size=0.1`: 10% des données pour le test

Entraînement GaussianNB

```
# Import de GaussianNB
from sklearn.naive_bayes import GaussianNB

# Instanciation du modèle
gnb = GaussianNB()

# Entraînement
gnb.fit(X_train, y_train)
```

Prédiction

```
# Prédiction sur train
predict_train = gnb.predict(X_train)

# Prédiction sur test
predict_test = gnb.predict(X_test)
```

Pipeline Complet

- 1 Chargement des données
- 2 Exploration et nettoyage
- 3 Séparation X et y
- 4 Équilibrage avec SMOTE
- 5 Standardisation
- 6 Division train/test
- 7 Entraînement GaussianNB
- 8 Évaluation

Paramètres de GaussianNB

priors

None (calculé auto)

var_smoothing

1e-9 (défaut)

Étape 4: Évaluation des Performances

✓ Calcul de l'Accuracy

```
# Import de accuracy_score
from sklearn.metrics import accuracy_score

# Accuracy sur train
accuracy_train = accuracy_score(y_train, predict_train)
print(f'Accuracy train: {accuracy_train:.4f}')

# Accuracy sur test
accuracy_test = accuracy_score(y_test, predict_test)
print(f'Accuracy test: {accuracy_test:.4f}')
```

> Résultats Obtenus

Accuracy Train

80.7%

0.8068

Accuracy Test

76.7%

0.7667

Q Interprétation des Résultats

Bonne capacité de généralisation

L'accuracy test (76.7%) est proche de l'accuracy train (80.7%), indiquant que le modèle généralise bien.

Léger surapprentissage

La différence de 4% suggère un léger surapprentissage, mais reste acceptable.

Performance satisfaisante

76.7% de précision est raisonnable pour une classification médicale avec peu de données.

⚠ Améliorations Possibles

- Validation croisée pour estimation plus robuste
- Feature selection pour réduire la dimension
- Essayer d'autres classificateurs (SVM, RF)
- Ajustement des hyperparamètres

Code Complet - Solution

```
# DETECTION DE LA MALADIE DE PARKINSON
# Classificateur: Gaussian Naive Bayes
# =====

# 1. IMPORT DES BIBLIOTHEQUES
import pandas as pd
import numpy as np
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# 2. CHARGEMENT DES DONNEES
data = pd.read_csv('parkinsons.csv')
print("Dataset shape:", data.shape)
print("Class distribution:", data['status'].value_counts())

# 3. PRETRAITEMENT
data.drop(['name'], axis=1, inplace=True)
X = data.drop('status', axis=1).values
y = data['status'].values

# 4. EQUILIBRAGE AVEC SMOTE
oversample = SMOTE()
X, y = oversample.fit_resample(X, y)
print("After SMOTE:", pd.Series(y).value_counts())

# 5. STANDARDISATION
scaler = StandardScaler()
X = scaler.fit_transform(X)

# 6. DIVISION TRAIN/TEST
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

# 7. ENTRAINEMENT DU MODELE
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

Conclusion et Perspectives

✓ Récapitulatif

- ✓ Dataset: 195 enregistrements vocaux, 23 caractéristiques
- ✓ Prétraitement: SMOTE pour équilibrage, StandardScaler
- ✓ Modèle: Gaussian Naive Bayes
- ✓ Performance: 76.7% accuracy sur test

★ Points Clés

- Le Naive Bayes Gaussien offre une approche simple et efficace
- L'analyse vocale est pertinente pour le diagnostic de Parkinson
- Le prétraitement (SMOTE, standardisation) est crucial

🔧 Améliorations Futures

- > **Autres classificateurs:** SVM, Random Forest, XGBoost
- > **Feature selection:** Sélectionner les attributs les plus pertinents
- > **Hyperparameter tuning:** GridSearchCV pour optimiser les paramètres
- > **Cross-validation:** K-fold pour estimation plus robuste
- > **Métriques complètes:** Precision, Recall, F1-score, ROC-AUC

🏥 Applications Cliniques

Cette approche pourrait être développée en **outil d'aide au diagnostic précoce**, accessible via smartphone pour un dépistage à grande échelle.

🎓 Merci pour votre attention!