

Nom & prenom : Brahim samba Lawol Toure

Num: C13682

Rapport : Classification de Patients Atteints de Cancer avec Random Forest

Introduction à Random Forest Définition

Random Forest est un algorithme d'apprentissage automatique supervisé qui crée une "forêt" d'arbres de décision et fusionne leurs prédictions pour obtenir une prédiction plus précise et stable.

Fonctionnement

1. **Bootstrap Aggregating (Bagging)** : Création de multiples échantillons du dataset original
2. **Construction des arbres** : Pour chaque échantillon, création d'un arbre de décision
3. **Random Feature Selection** : À chaque nœud, sélection aléatoire d'un sous-ensemble de caractéristiques
4. **Agrégation** : Combinaison des prédictions par vote majoritaire (classification) ou moyenne (régression)

Avantages

- Robuste au surapprentissage
- Gère bien les données manquantes et les valeurs aberrantes
- Fournit des mesures d'importance des caractéristiques
- Parallélisable

Inconvénients

- Complexité computationnelle élevée
- Moins interprétable qu'un seul arbre de décision
- Nécessite plus de mémoire

Cas d'Utilisation : Prédiction de Survie des Patients Atteints de Cancer Dataset

Données synthétiques de patients atteints de cancer en Chine, comprenant :

- Caractéristiques démographiques (âge, genre, province)
- Informations médicales (type de tumeur, stade, traitement)
- Variables comportementales (tabagisme, alcool)
- Statut de survie (variable cible)

Implémentation

Le notebook contient :

1. Chargement et prétraitement des données
2. Encodage des variables catégorielles
3. Entraînement du modèle Random Forest
4. Évaluation des performances
5. Visualisation des résultats

Code Principal

```
# Prétraitement df_encoded =
pd.get_dummies(df[cat_columns]) df_final =
pd.concat([df_encoded, df[num_columns]], axis=1)

# Modélisation X = df_final y =
(df['SurvivalStatus'] == 'Deceased').astype(int)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Entraînement rf_model = RandomForestClassifier(n_estimators=100,
random_state=42)
rf_model.fit(X_train, y_train)
```

```
# Évaluation y_pred =  
rf_model.predict(X_test)
```

Résultats

- Matrices de confusion visualisant les prédictions correctes/incorrectes
- Top 20 des caractéristiques les plus importantes pour la prédiction
- Métriques de performance (accuracy, precision, recall, F1-score)

Conclusion

Random Forest s'avère efficace pour la prédiction du statut de survie des patients, offrant :

- Une bonne performance prédictive
- Des insights sur les facteurs les plus importants
- Une base solide pour la prise de décision médicale

Analyse des Algorithmes Utilisés dans le Projet Cancer

Algorithmes de Classification Implémentés

1. Régression Logistique

- Modèle de classification binaire pour prédire les résultats du cancer
- Implémentation en Python avec scikit-learn :

```
from sklearn.linear_model import LogisticRegression  
model = LogisticRegression()
```

2. Random Forest (Forêt Aléatoire)

- Méthode d'apprentissage d'ensemble utilisant plusieurs arbres de décision

- Analyse d'importance des caractéristiques :

```
from sklearn.ensemble import RandomForestClassifier  
rf_model = RandomForestClassifier(n_estimators=100)
```

3. SVM (Machines à Vecteurs de Support)

- Création de frontières de décision pour la classification du cancer
- Efficace avec les données médicales :

```
from sklearn.svm import SVC  
svm_model = SVC(kernel='rbf')
```

Analyse Complémentaire

ACP (Analyse en Composantes Principales)

- Réduction de dimensionnalité
- Visualisation des caractéristiques :

```
from sklearn.decomposition import PCA  
pca = PCA(n_components=2)
```

Raisons du Choix

- Adapté aux problèmes de classification binaire (cancer/non-cancer)
- Bonne interprétabilité des résultats
- Performance optimale pour la taille du jeu de données
- Capacité à gérer efficacement les données médicales

Annexes

