Groupe : Yanis Hamdaoui, Wassil Fahem

TP Machine Learning - Activité de laboratoire 7 – Réseau de neurones artificiels

Voici notre code provenant de notre notebook, que vous trouverez dans notre github.

```
[75] import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import LabelEncoder, StandardScaler
     from keras.models import Sequential
     from keras.layers import Dense

     # Charger l'ensemble de données
     df = pd.read_csv('bank_customers.csv')

     # Inspecter et nettoyer les données si nécessaire
     df.isnull().sum()

     # Supprimer les fonctionnalités indésirables
     df = df.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1)

     # Encoder les caractéristiques catégorielles en numérique
     le = LabelEncoder()
     df['Geography'] = le.fit_transform(df['Geography'])
     df['Gender'] = le.fit_transform(df['Gender'])

     # Diviser les données en ensembles d'entraînement et de test
     X = df.iloc[:, 1:-1].values
```

```
[75] y = df.iloc[:, -1].values
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size

     # Mettre à l'échelle les données
     sc = StandardScaler()
     X_train = sc.fit_transform(X_train)
     X_test = sc.transform(X_test)

     # Initialiser le modèle ANN
     classifier = Sequential()

     # Ajouter la couche d'entrée avec l'argument input_shape
     classifier.add(Dense(units=6, kernel_initializer='uniform', activat

     # Ajouter la deuxième couche cachée
     classifier.add(Dense(units=6, kernel_initializer='uniform', activat

     # Ajouter la couche de sortie
     classifier.add(Dense(units=1, kernel_initializer='uniform', activat

     # Configurer le processus d'apprentissage
     classifier.compile(optimizer='adam', loss='binary_crossentropy', me

     # Former le modèle ANN
```

```
# Former le modèle ANN
classifier.fit(X_train, y_train, batch_size=10, epochs=100)

# Faire des prédictions sur l'ensemble de test
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)  # seuillage pour convertir les probabilités en valeurs binaires

# Évaluer les performances du modèle sur l'ensemble de test
from sklearn.metrics import confusion_matrix, accuracy_score

cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)

print("Matrice de confusion :")
print(cm)
print("\nPrécision : {:.2f}%".format(accuracy * 100))
```

Ce code fourni implémente un modèle d'apprentissage automatique utilisant un réseau de neurones artificiels (ANN) pour prédire la résiliation de clients bancaires.

Dans un premier temps, les chargements des données : Les données sont chargées à partir d'un fichier CSV nommé "bank_customers.csv" en utilisant la bibliothèque pandas.

Ensuite l'inspection et nettoyage des données : Une vérification rapide est effectuée pour détecter les valeurs manquantes. Aucune valeur manquante n'est trouvée, donc aucune opération de nettoyage n'est nécessaire.

Puis, la suppression des fonctionnalités indésirables : Trois fonctionnalités indésirables ('RowNumber', 'CustomerId', 'Surname') sont supprimées de l'ensemble de données car elles ne sont pas pertinentes pour la prédiction.

On encode les caractéristiques catégorielles : Les caractéristiques catégorielles ('Geography' et 'Gender') sont encodées en numérique à l'aide de LabelEncoder de sklearn.

On divise des données en ensembles d'entraînement et de test : Les données sont divisées en ensembles d'entraînement et de test à l'aide de train_test_split de sklearn. L'ensemble de test représente 20% des données.

Nous mettons la mise à l'échelle des données : Les données sont mises à l'échelle à l'aide de StandardScaler de sklearn pour normaliser les caractéristiques.

On initialise le modèle ANN : Un modèle séquentiel est initialisé à l'aide de Sequential de Keras.

Ajout des couches du réseau de neurones : Une couche d'entrée avec 6 neurones, une initialisation uniforme des poids et une fonction d'activation ReLU. Une deuxième couche cachée avec 6 neurones, une initialisation uniforme des poids et une fonction d'activation ReLU. Une couche de sortie avec 1 neurone, une initialisation uniforme des poids et une fonction d'activation sigmoïde.

On configure le processus d'apprentissage : Le modèle est compilé avec l'optimiseur 'adam', la perte binaire 'binary_crossentropy' et la métrique 'accuracy'.

Nous formons le modèle : Le modèle est entraîné sur l'ensemble de données d'entraînement avec un batch_size de 10 et 100 epochs.

Prédictions sur l'ensemble de test : Des prédictions sont faites sur l'ensemble de test et les valeurs sont seuillées à 0,5 pour convertir les probabilités en valeurs binaires.

Évaluation des performances du modèle : La matrice de confusion et la précision sont calculées à l'aide de confusion_matrix et accuracy_score de sklearn.

Nous obtenons le résultat suivant :

```
Epoch 100/100
800/800 [==============================
63/63 [==============================
Matrice de confusion :
[[1540   55]
 [ 261  144]]

Précision : 84.20%
```

Catégorie vrais négatifs (VN) : Il y a 1540 clients prédits comme non-résiliés et qui sont réellement non-résiliés.

Catégorie faux positifs (FP) : Il y a 55 clients prédits comme résiliés mais qui sont réellement non-résiliés.

Catégorie faux négatifs (FN) : Il y a 261 clients prédits comme non-résiliés mais qui sont réellement résiliés.

Catégorie vrais positifs (VP) : Il y a 144 clients prédits comme résiliés et qui sont réellement résiliés.

La précision du modèle est calculée en divisant le nombre de prédictions correctes (VN + VP) par le nombre total d'échantillons. Dans ce cas, la précision est de 84,20%, ce qui indique que le modèle prédit correctement la résiliation des clients dans environ 84,20% des cas.

Pour la suite :

```python
# Nouvelles données à prédire
new_data = pd.DataFrame({
    'Geography': ['France'],
    'CreditScore': [600],
    'Gender': ['Male'],
    'Age': [40],
    'Tenure': [3],
    'Balance': [60000],
    'NumOfProducts': [2],
    'HasCrCard': ['Yes'],
    'IsActiveMember': ['Yes'],
    'EstimatedSalary': [50000]
})

# Charger l'ensemble de données original pour obtenir les labels existants
df_original = pd.read_csv('bank_customers.csv')

# Encoder les caractéristiques catégorielles en numérique
le_geography = LabelEncoder()
le_gender = LabelEncoder()
le_has_cr_card = LabelEncoder()
le_is_active_member = LabelEncoder()
```

```
[74] # Mettre à jour les LabelEncoders avec les labels existants
     le_geography.fit(df_original['Geography'])
     le_gender.fit(df_original['Gender'])

     # Transformation spécifique pour 'HasCrCard' et 'IsActiveMember'
     le_has_cr_card.fit(['No', 'Yes'])  # 'No' is encoded as 0, 'Yes' as 1
     le_is_active_member.fit(['No', 'Yes'])  # 'No' is encoded as 0, 'Yes' as 1

     # Appliquer l'encodage sur les nouvelles données
     new_data['Geography'] = le_geography.transform(new_data['Geography'])
     new_data['Gender'] = le_gender.transform(new_data['Gender'])
     new_data['HasCrCard'] = le_has_cr_card.transform(new_data['HasCrCard'])
     new_data['IsActiveMember'] = le_is_active_member.transform(new_data['IsActiveMember'])

     # Mettre à l'échelle les données
     new_data_scaled = sc.transform(new_data.values[:, 1:])

     # Faire la prédiction
     prediction = classifier.predict(new_data_scaled)
     prediction_result = (prediction > 0.5)

     # Afficher le résultat de la prédiction
     print("La prédiction pour le client est :", "Quitte la banque" if prediction_result else "Ne quitte pas la b
```

Voici le résultat :

```
[74] # Appliquer l'encodage sur les nouvelles données
     new_data['Geography'] = le_geography.transform(new_data['Geography'])
     new_data['Gender'] = le_gender.transform(new_data['Gender'])
     new_data['HasCrCard'] = le_has_cr_card.transform(new_data['HasCrCard'])
     new_data['IsActiveMember'] = le_is_active_member.transform(new_data['IsActiveMember'])

     # Mettre à l'échelle les données
     new_data_scaled = sc.transform(new_data.values[:, 1:])

     # Faire la prédiction
     prediction = classifier.predict(new_data_scaled)
     prediction_result = (prediction > 0.5)

     # Afficher le résultat de la prédiction
     print("La prédiction pour le client est :", "Quitte la banque" if prediction_result else "Ne quitte pas la ban

     1/1 [==============================] - 0s 53ms/step
     La prédiction pour le client est : Ne quitte pas la banque
```

Le code ci-dessus signifie :

Que l'on va créer de nouvelles données à prédire :

Les caractéristiques du client, telles que la géographie, le score de crédit, le sexe, l'âge, ... sont fournies dans un DataFrame appelé new_data.

On récupère le chargement des données d'origine : Les données originales du fichier CSV 'bank_customers.csv' sont chargées dans le DataFrame df_original. Cela est fait pour obtenir les labels existants nécessaires pour l'encodage.

On encode des caractéristiques catégorielles : Les caractéristiques catégorielles telles que 'Geography', 'Gender', 'HasCrCard' et 'IsActiveMember' sont encodées en numérique à l'aide de l'objet LabelEncoder de la bibliothèque scikit-learn. Les LabelEncoders sont ajustés aux labels existants dans les données originales.

Transformation spécifique pour certaines caractéristiques : Pour les caractéristiques 'HasCrCard' et 'IsActiveMember', des LabelEncoders spécifiques sont créés en définissant manuellement les valeurs 'No' et 'Yes' pour l'encodage.

Application de l'encodage sur les nouvelles données : Les nouvelles données sont transformées en utilisant les LabelEncoders ajustés pour convertir les valeurs catégorielles en numériques.

Mise à l'échelle des données : Les données nouvellement encodées sont mises à l'échelle en utilisant le même objet StandardScaler utilisé précédemment pour mettre à l'échelle les données d'entraînement et de test.

Prédiction : Les données mises à l'échelle sont utilisées pour prédire si le client va quitter la banque ou non, en utilisant le modèle de classification neuronal entraîné précédemment.

Affichage du résultat de la prédiction : Le résultat de la prédiction est affiché, indiquant si le client est prédit comme quittant ou restant dans la banque, en fonction du seuil de probabilité de 0,5.

Ce code permet de prédire si un nouveau client quittera ou non la banque en se basant sur les caractéristiques fournies.