Université Abdelmalek Essaâdi
Faculté des Sciences et Techniques d'Al-Hoceima
Département de Mathématiques et Informatique

Filière : LST- IDLL
Module : Intelligence artificielle
A .U : 2023/2024
Pr : ZANNOU Abderrahim

Version 1

TP N°8

Énoncé (ANN-Problème de régression)

L'objectif de ce TP est d'appliquer :

- ✓ Réseaux de neurones artificiels (ANN) sur un problème de régression

Tâche 1 : Prévoir le profit d'une entreprise

- ✓ **Jeux de données (Dataset) : profit.csv**
- ✓ **La variable cible** : *Profit*
- ✓ **Le fichier associé :** TP8_1_Regression_ANN_Profit
- ✓ **Les étapes essentielles :**
  - Importer du jeu de données
  - Répartir les données sur X et Y :
    X=df.iloc[:,:-1].values
    Y=df.iloc[:,-1].values
  - Encoder l'attribut catégorielle *State* en utilisant OneHotEncoder :
    ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='passthrough')
    X = np.array(ct.fit_transform(X))
  - Normaliser l'ensemble X en utilisant RobustScaler
    scaler = RobustScaler()
    X= scaler.fit_transform(X)
  - Diviser l'ensemble de données en ensemble d'entraînement et ensemble de test
  - Initialiser l'ANN
    ann = tf.keras.models.Sequential()
  - Ajouter de la couche d'entrée et les couches cachées
    ann.add(tf.keras.layers.Dense(units=4, activation='relu'))
    ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
  - Ajouter de la couche de sortie
    ann.add(tf.keras.layers.Dense(units=1, activation=None))
  - Compiler l'ANN
    # Spécifier le taux d'apprentissage
    learning_rate = 0.2
    # Création de l'optimiseur Adam avec le taux d'apprentissage spécifié
    opt = tf.keras.optimizers.Adam(learning_rate=learning_rate)
    # Compilation du modèle avec l'optimiseur spécifié et la fonction de perte
    ann.compile(optimizer=opt, loss='mse')
  - Entraîner l'ANN sur l'ensemble d'entraînement
    ann.fit(X_train.astype('float32'), Y_train, batch_size = 4, epochs = 100,verbose=1)

- Prédire les profits de l'ensemble de tests
  Y_pred = ann.predict(X_test.astype('float32'))
- Déterminer les différentes métriques d'évaluation du modèle (Erreurs)
- Prédire le profits pour E1 (142007, 91321, 366268, California) et E2(165349, 136897, 471784, 'California')

***Tâche 2 : Refaire la tâche 1* pour obtenir des résultats reproductibles**

- ✓ **Le fichier associé :** TP8_2_Regression_ANN_Fixe_Profit
- ✓ **Code à ajouter :**

```
# Fixer la graine aléatoire pour la reproductibilité
tf.random.set_seed(42)
# Définir un initialisateur de poids avec une graine aléatoire fixe
initializer = tf.keras.initializers.GlorotUniform(seed=42)
```

- ✓ **Code à modifier**

```
ann.add(tf.keras.layers.Dense(units=4, activation='relu' ,kernel_initializer=initializer))
ann.add(tf.keras.layers.Dense(units=6, activation='relu' ,kernel_initializer=initializer))
ann.add(tf.keras.layers.Dense(units=1, activation=None ,kernel_initializer=initializer))
```

***Tâche 3 : Refaire la tâche 2* en subdivisant les données sur training, validation et test**

- ✓ **Le fichier associé :** TP8_3_Regression_ANN_Fixe_Validation_Profit
- ✓ **Code à modifier :**

```
X_train, X_temp, Y_train, Y_temp = train_test_split(X, Y, test_size=0.3, random_state=42)
X_val, X_test, Y_val, Y_test = train_test_split(X_temp, Y_temp, test_size=0.5, random_state=42)
ann.fit(X_train.astype('float32'), Y_train, batch_size=32, epochs=300,
validation_data=(X_val.astype('float32'), Y_val), verbose=1)
```