

Crie um modelo de Deep Learning. Comece criando uma massa de dados de exemplo com pelo menos 3 variáveis preditoras e uma variável alvo do tipo categórica. Explore rapidamente os dados, faça o pré-processamento, crie uma arquitetura multi-layer perceptron, treine e avalie o modelo usando pelo menos duas métricas. Use Linguagem Python.

Claro, vou criar um exemplo simples de um modelo de Deep Learning em Python usando a biblioteca TensorFlow e Keras. Vou criar uma massa de dados fictícia com três variáveis preditoras e uma variável alvo categórica. Em seguida, criarei uma arquitetura de Multi-Layer Perceptron (MLP), treinarei o modelo e avaliarei usando métricas de precisão e matriz de confusão.

Certifique-se de ter as bibliotecas TensorFlow e outras bibliotecas necessárias instaladas. Você pode instalá-las usando pip:

```
pip install tensorflow scikit-learn numpy
```

Aqui está o código:

```
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix

# Criar dados de exemplo
np.random.seed(0)
num_samples = 1000
feature1 = np.random.rand(num_samples) # Variável preditora 1
feature2 = np.random.rand(num_samples) # Variável preditora 2
feature3 = np.random.rand(num_samples) # Variável preditora 3
target = np.random.choice(['A', 'B', 'C'], num_samples) # Variável alvo categ

# Pré-processamento de dados
encoder = LabelEncoder()
target_encoded = encoder.fit_transform(target)
X = np.column_stack((feature1, feature2, feature3))
y = tf.keras.utils.to_categorical(target_encoded, num_classes=3)

# Dividir o conjunto de dados em treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Criar a arquitetura do modelo MLP
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(3,)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax') # Camada de saída com 3 classes
])

# Compilar o modelo
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
# Treinar o modelo
model.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2)

# Avaliar o modelo usando métricas
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_test_classes = np.argmax(y_test, axis=1)

accuracy = accuracy_score(y_test_classes, y_pred_classes)
confusion = confusion_matrix(y_test_classes, y_pred_classes)

print("Acurácia do modelo:", accuracy)
print("Matriz de Confusão:")
print(confusion)
```

Este é um exemplo básico de como criar um modelo de Deep Learning em Python usando TensorFlow e Keras. Certifique-se de ajustar a arquitetura do modelo, hiperparâmetros e tamanho dos conjuntos de dados de acordo com seus requisitos específicos. Este modelo usa três variáveis preditoras e uma variável alvo categórica, mas você pode expandi-lo para um caso de uso real com mais dados e recursos.

Não deveria dividir os dados em treino e teste antes de aplicar o pré-processamento?

Você está correto, é uma prática recomendada dividir os dados em conjuntos de treinamento e teste antes de aplicar qualquer pré-processamento. Isso evita qualquer vazamento de informações do conjunto de teste para o conjunto de treinamento durante o pré-processamento. Portanto, você deve dividir os dados antes de fazer qualquer codificação ou transformação.

Aqui está o código atualizado com a divisão de dados antes do pré-processamento:

```
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix

# Criar dados de exemplo
np.random.seed(0)
num_samples = 1000
feature1 = np.random.rand(num_samples) # Variável preditora 1
feature2 = np.random.rand(num_samples) # Variável preditora 2
feature3 = np.random.rand(num_samples) # Variável preditora 3
target = np.random.choice(['A', 'B', 'C'], num_samples) # Variável alvo categ

# Dividir o conjunto de dados em treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(
    np.column_stack((feature1, feature2, feature3)),
    target,
    test_size=0.2,
    random_state=42
)
```

```

# Pré-processamento de dados
encoder = LabelEncoder()
y_train_encoded = encoder.fit_transform(y_train)
y_test_encoded = encoder.transform(y_test)

y_train_one_hot = tf.keras.utils.to_categorical(y_train_encoded, num_classes=3)
y_test_one_hot = tf.keras.utils.to_categorical(y_test_encoded, num_classes=3)

# Criar a arquitetura do modelo MLP
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(3,)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax') # Camada de saída com 3 classes
])

# Compilar o modelo
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Treinar o modelo
model.fit(X_train, y_train_one_hot, epochs=20, batch_size=32, validation_split=0.1)

# Avaliar o modelo usando métricas
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)

accuracy = accuracy_score(y_test_encoded, y_pred_classes)
confusion = confusion_matrix(y_test_encoded, y_pred_classes)

print("Acurácia do modelo:", accuracy)
print("Matriz de Confusão:")
print(confusion)

```

Agora, os dados são divididos em treinamento e teste antes de qualquer pré-processamento, garantindo que o processo seja realizado de maneira adequada.

Is this conversation helpful so far?

