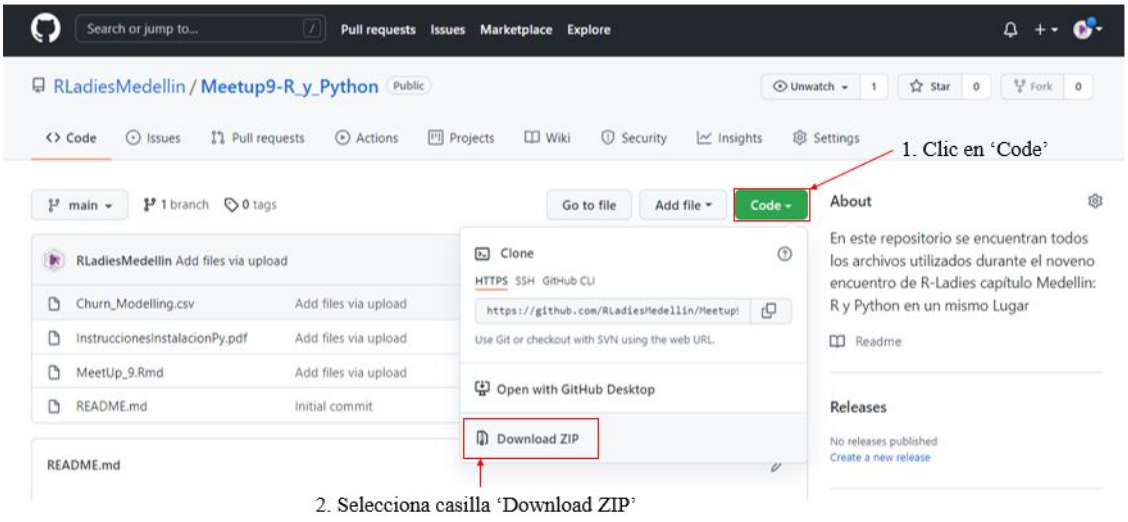
MeetUp9_Python_R

RLadies_Medellin

24/11/2021

Noveno Encuentro R Ladies Medellín: R y Python en el mismo lugar



Nuestra actividad práctica estará dividida en tres partes:

- **Parte 1:** Pre-procesamiento de datos.
- **Parte 2:** Construcción de la red neuronal artificial.
- **Parte 3:** Evaluación del modelo y cálculo de predicciones finales.
- **Sin embargo...** Antes necesitamos inicializar Python e importar librerías necesarias.

Inicializar Python

```
library(reticulate)
use_python('your_path')
py_install("scikit-learn")
py_install("keras")
py_install("tensorflow")
# reticulate::py_config()
```

Importar librerias

```
import numpy as np
import pandas as pd
import sklearn
```

Importar DataSet

```
dataset = pd.read_csv("C:/Users/valen/Downloads/Meetup 9/Churn_Modelling.csv")
```

Parte 1

Pre - procesamiento (generar la matriz de características)

```
X = dataset.iloc[:, 3:13].values
y = dataset.iloc[:, 13].values
```

Codificar datos categóricos

```
from sklearn.preprocessing import LabelEncoder

labelencoder_X_1 = LabelEncoder()
X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1])
labelencoder_X_2 = LabelEncoder()
X[:, 2] = labelencoder_X_2.fit_transform(X[:, 2])
```

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

transformer = ColumnTransformer(
    transformers=[
        ("Churn_Modelling",
         OneHotEncoder(categories='auto'),
         [1]
         )
    ], remainder='passthrough'
)

X = transformer.fit_transform(X)
X = X[:, 1:]
```

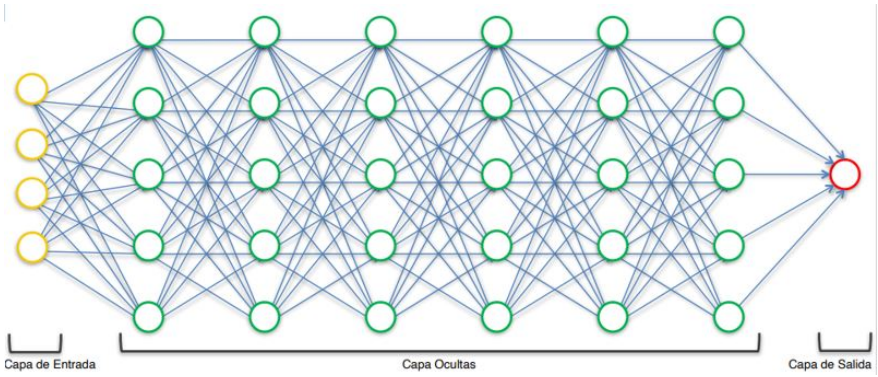
Dividir el dataset (entrenamiento y test)

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Normalizar las variables

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

Parte 2



Construir la RNA

```
import keras
```

```
## Using TensorFlow backend.
```

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
```

Inicializar la RNA

```
classifier = Sequential()
```

Diseñar la RNA

```
# Añadir las capas de entrada y primera capa oculta
classifier.add(Dense(units = 6, kernel_initializer = "uniform",  activation = "relu", input_dim = 11))
classifier.add(Dropout(rate = 0.1))

# Añadir la segunda capa oculta
classifier.add(Dense(units = 6, kernel_initializer = "uniform",  activation = "relu"))
classifier.add(Dropout(rate = 0.1))

# Añadir la capa de salida
classifier.add(Dense(units = 1, kernel_initializer = "uniform",  activation = "sigmoid"))
```

Compilar la RNA

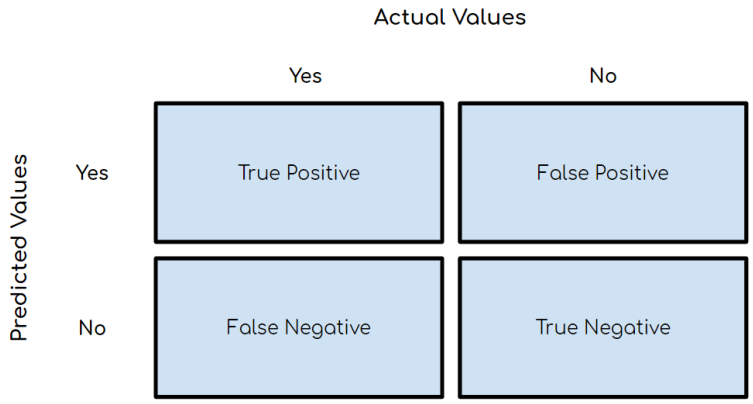
```
classifier.compile(optimizer = "adam", loss = "binary_crossentropy", metrics = ["accuracy"])
```

```
## WARNING:tensorflow:From C:\Users\valen\AppData\Local\R-MINI~1\envs\R-RETI~1\lib\site-packages\tensor
## Instructions for updating:
## Use tf.where in 2.0, which has the same broadcast rule as np.where
```

```
classifier.fit(X_train, y_train,  batch_size = 10, epochs = 50, verbose= 0)
```

```
## <keras.callbacks.callbacks.History object at 0x00000000422FEEF0>
##
## WARNING:tensorflow:From C:\Users\valen\AppData\Local\R-MINI~1\envs\R-RETI~1\lib\site-packages\keras\
```

Parte 3



Evaluar el modelo y calcular predicciones finales

```
y_pred  = classifier.predict(X_test)
y_pred = (y_pred>0.5)
```

Elaborar la matriz de confusión

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print((cm[0][0]+cm[1][1])/cm.sum())
```

```
## 0.8515
```

Ejercicio Final

- Utiliza nuestro modelo de RNA para predecir si el cliente con la siguiente información abandonará el banco:
- **Geografia:** Francia
- **Puntaje de crédito:** 600
- **Género:** Masculino
- **Edad:** 40 años de edad
- **Tenencia:** 3 años
- **Saldo:** $ 60000
- **Número de productos:** 2
- **¿Este cliente tiene una tarjeta de crédito?** Sí
- **¿Es este cliente un miembro activo?** Sí
- **Salario estimado:** $ 50000

```
new_prediction = classifier.predict(sc_X.transform(np.array(
                                    [[0,0,600, 1, 40, 3, 60000, 2, 1, 1, 50000]])))
print(new_prediction)
```

```
## [[0.09396186]]
```

```
print(new_prediction > 0.5)
```

```
## [[False]]
```