exercise-lecture 2-3.ipynb
Archivo
Editar
Ver
Insertar
Entorno de ejecución
Herramientas
Ayuda
No se pueden guardar cambios
Compartir
+ Código
+ Texto
Copiar en Drive
RAM
Disco
1803

Homework 2.3.

TASK: T1. - Reservoir

Suppose you need to gather a sample from sales at the online market place during the Christmas season. Write the code to the *reservoir algorithm* in order to gather the sample. Each sale uses an integer as id. See here

I. sign in grader

```
!pip install graphviz

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_graphviz
import graphviz

# Configura la semilla aleatoria para reproducibilidad
np.random.seed(42)

# Genera un conjunto de datos ficticio para juegos de estrategia con más escenarios
num_samples = 50000  # Aumenta el número de muestras
features = {
    'numero_de_unidades': np.random.randint(1, 100, num_samples),
    'recursos_disponibles': np.random.uniform(0, 1000, num_samples),
    'nivel_de_dificultad': np.random.randint(1, 5, num_samples),
    'tiempo_de_juego': np.random.uniform(0, 50, num_samples),
    'estrategia_utilizada': np.random.choice(['Defensiva', 'Ofensiva', 'Mixta'], num_samples),
}

# Agrega ruido a las características numéricas
numeric_features = ['numero_de_unidades', 'recursos_disponibles', 'nivel_de_dificultad', 'tiempo_de_juego']
for feature in numeric_features:
    features[feature] = features[feature].astype(float) + np.random.normal(0, 5, num_samples)

# Codifica la característica categórica 'estrategia_utilizada' usando one-hot encoding
df = pd.DataFrame(features)
df = pd.get_dummies(df, columns=['estrategia_utilizada'], prefix='estrategia')

# Define la variable objetivo (etiqueta)
# Supongamos que la etiqueta representa si el jugador ganó o perdió el juego (1 si ganó, 0 si perdió)
df['ganador'] = (df['numero_de_unidades'] + 2 * df['recursos_disponibles'] -
          0.5 * df['nivel_de_dificultad'] + 0.2 * df['tiempo_de_juego'] > 150).astype(int)

# Divide los datos en conjuntos de entrenamiento y prueba
X = df.drop('ganador', axis=1)
y = df['ganador']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Crea y entrena el modelo de Árbol de Decisión
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# Realiza predicciones en el conjunto de prueba
y_pred = model.predict(X_test)

# Evalúa la precisión del modelo
accuracy = np.mean(y_pred == y_test)
print(f'Precisión del modelo de Árbol de Decisión: {accuracy:.2f}')

# Exporta el árbol a un archivo .dot
dot_data = export_graphviz(model, out_file=None,
                           feature_names=X.columns,
                           class_names=['Perdió', 'Ganó'],
                           filled=True, rounded=True, special_characters=True)

# Visualiza el árbol
graph = graphviz.Source(dot_data)
graph.render(filename='decision_tree_strategy_game', format='png', cleanup=True)
graph
```

```
[ ] # Please DON'T MODIFY THIS CODE
    # init.course_id = "LOGICA_3.v1"
    init.course_id = "bigdata.v2"
    import inspect
    from local.lib.rlxmoocapi import session
    session.LoginSequence(endpoint=init.endpoint, course_id=init.course_id, lab_id="L02.03", varname="student");
```

```
logging in as freddy.duitama@gmail.com... please wait
using course session bigdata-2023-1::group02
success!! you are logged in
```

II. Add your solution

```
[ ] # function to generate a random stream
    def generate_stream(size_stream):       # size_stream is the stream size.
      import random
      return random.sample(range(size_stream*10), size_stream)
```

```
[ ]
    def reservoir(stream,size_sample):       # stream is a list, its elements will be generated by the function generate_stream
      #RESERVOIR
      import numpy as np
      import random
      ### START CODE HERE ###
      sample =
      list_items = []                        # list to store random values assigned to j



      ### END CODE HERE ###
      return sample, list_items
```

III. Test your solution

```
[ ] size_stream = int(input('Ingrese el numero de elementos del stream  de entrada '))
    size_sample = int(input('Ingrese el numero de elementos de la muestra '))
    stream_aleatorio = generate_stream(size_stream)
    sample , lista_j = reservoir(stream_aleatorio,size_sample)
    sample
```

IV. Submit your solution to the grader

```
[ ] ## Student
    student.submit_task(globals(), task_id="T1");
```

TASK: T2. - Coding the algorithm 3.

Write the code to the **algorithm 3**. It gathers a sample to study the typical behavior of the individuals in the stream. The tuple < *id, value* > identifies each element in the stream.

Figure 1:
Hash function to pick a fraction $\frac{a}{b}$ of users with respect to the population
If $h(key) \leq a$ **then** add individual in the sample.

Algorithm 3.

```
buckets := self._table_size(capacity)
h(key) := getFunctions_(self)
sample_size := m
threshold := buckets - 1
for each element in stream ( key, {value} )   do :
    if ( h(key) < threshold)   then   add element to sample
                               else   discard element

    while ( users in sample > sample_size)
        remove elements in sample with h(user) = threshold
        threshold := threshold - 1
```

Let suppose:

- Population size = 10.000 users.
- Stream length = 15.000 user requests.
- sample size = 278

```
[ ] !pip install primesieve==2.3.0
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting primesieve==2.3.0
  Downloading primesieve-2.3.0-cp38-cp38-manylinux2010_x86_64.whl (2.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.4/2.4 MB 22.1 MB/s eta 0:00:00
Installing collected packages: primesieve
Successfully installed primesieve-2.3.0
```

```
[ ] # Define hash family class
    # Define  an ADT and its methods
    def my_solution_t2(capacity):

      # define  an ADT and its methods
      import numpy as np
      from primesieve import n_primes
      import hashlib
      class HashFamily:

        def __init__(self, capacity):
            self._buckets = n_primes(1, capacity+1)[0]                   # Bucket number.
            self._table = [None] * buckets
            self._a = np.random.randint(1, buckets-1)                    # variable a
            self._b = np.random.randint(0, buckets-1)                    # variable b
            self._n = 0                                                  # input size
```

✓ 11 s completado a las 15:08