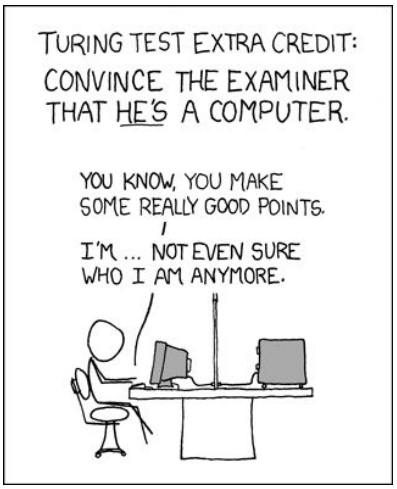
jampp

Data Science and Machine Learning Exam



Tiene **1 día** para completar el examen.

Nos interesa conocer la forma en que llegaste a la solución que proponés además de la solución en sí misma.

Cuando más nos puedas explicar la forma en que lo pensaste, mejor.

A menos que se aclare, las soluciones deben estar presentadas en Python 3. La prolijidad será tenida en cuenta.

¡Esperamos que te diviertas!

jampp

Linux Shell

1. Conteste al menos **dos** de las siguientes preguntas

    a. ¿Con que comando contaría la cantidad de veces que aparece la palabra *Unix* en un archivo?

    b. ¿Para qué sirve el comando chmod? ¿Qué se entiende por la línea r-- -w- --x? ¿Numéricamente a qué valores corresponde?

    c. Se necesita liberar espacio en el disco donde está montado el directorio *home* del usuario *ec2-user*. ¿Qué comandos, métodos o lógica utilizaría para ver qué archivos conviene borrar?

    d. ¿Cómo reemplazaría todas las ocurrencias de la palabra *Unix* por *UNIX* en el archivo unix_test_file?

    a. ¿Qué comando o método utilizaría para ver constantemente las últimas 100 líneas con la palabra ERROR que se van apendeando a un archivo de log?

jampp

Coding Exercises

Resolver al menos 2 de los siguientes ejercicios. **Indicar** qué se eligió.

1. Tenemos una matriz de elementos, donde:
   - Cada fila está ordenada de menor a mayor.
   - El último elemento de cada fila es menor que el primero de la siguiente.

   Implemente una función que devuelva si un número pertenece a la matriz o no.

   Ejemplo:

```
matriz = [[1, 3, 5],
          [7, 9, 11],
          [13, 15, 17]]

> funcion(matriz, 7)
True

> funcion(matriz, 4)
False
```

2. Se sospecha que la siguiente función contiene un error. Validé si lo tiene y en caso de que lo tenga, corríjalo.
   La función devuelve valores aleatoriamente de una lista, de acuerdo a su peso.
   Por ejemplo *weighted_random([1, 2, 3], [0.5, 0.3, 0.2])*, debería devolver *1* con un *50%* de probabilidades, *2* con un *30%* y *3* con un *20%*.

```
import random

def weighted_random(values, weights):
    total_weight = sum(weights)
    acum_weights = [w / total_weight for w in weights[:]]
    for i in range(len(weights)):
        acum_weights[i] += acum_weights[i]
    rand = random.random()
    for value, weight in zip(values, acum_weights):
        if weight > rand:
            return value
```

3. Matthew tiene que escribir muchas queries que extraen valores de jsons anidados con la siguiente sintaxis y quiere una función para escribir las queries más rápido.
   Quiere pasar de un json como el siguiente:

```
{'products': ['', 'price', {'description': ['name', 'id']}]}
```

   A imprimir algo asi:

```
json_extract(msg, '$.product'),
json_extract(msg, '$.product.price'),
json_extract(msg, '$.product.description.id'),
json_extract(msg, '$.product.description.name')
```

   Donde debe haber una fila por valor final no anidado.

SQL

1. Dada la siguiente base de datos relacional, de cuentas y transacciones, escribir una query SQL que calcule para cada cuenta, el promedio de volumen de transacciones por día, incluyendo sólo los días con actividad. Por ejemplo, si en el primer día extrajo $8, e introdujo $12, el segundo día no hizo nada, y el tercero extrajo $4, el volumen del primer día es $20 y el del tercero $4, resultando en un promedio de $12.

accounts		**transactions**
id integer PK	1 N	account_id integer
balance decimal		value decimal (>0)
		tx_date date
		operation enum('EXTRACT', 'DEPOSIT')
		id integer PK

2. Se cuenta con la siguiente tabla que tiene información agregada por día.

Daily_data

Columna	Tipo	Descripción
date	Datetime	Día (%Y-%m-%d)
impresiones	Integer	Cantidad de impresiones en el día
clicks	Integer	Cantidad de clicks en el día
precio_promedio	Float	Precio promedio de las impresiones del día
prediccion_ctr_promedio	Float	Predicción promedio del ctr de las impresiones del día

Escriba una consulta que devuelva la siguiente información agregada por mes:

- Mes.
- Cantidad de impresiones del mes.
- Cantidad de clicks del mes.
- CTR del mes.
- Precio promedio de las impresiones del mes.
- Predicción de CTR promedio del mes.

Donde el **CTR** se define como **clicks / impresiones**.

jampp

Machine Learning

En la industria de la publicidad online, se llama **impresión** a cuando se muestra un anuncio a un usuario, y **click**, a cuando el usuario clickea el mismo, y en los casos en los que se hace click, ambos eventos son guardados compartiendo un identificador correspondiente a la transacción.
A menudo se considera una tarea importante la capacidad de las empresas para predecir que usuarios van a hacer click en un anuncio antes de mostrarlo, ya que puede haber más de un anuncio disponible y mostrarlo incurre un costo.

Se cuenta con una base de datos con el siguiente esquema:

Impressions
Id de la transacción
Timestamp
4 variables numéricas
20 variables categóricas

<---------->

Clicks
Id de la transacción
Timestamp
1 variables categórica

Aquí (descargar) se cuenta con una base de datos sqlite3 con datos de ejemplo, para trabajar.
La misma cuenta con una muestra de tres días de datos, en la que las variables categóricas han sido hasheadas para mantener los nombres anónimos, esto puede considerarse irrelevante, las mismas corresponden a datos como el país donde ocurrió la impresión o a que cliente y proveedor corresponde.

1. Utilizar la base de datos para entrenar un modelo para estimar si una impresión va a ser clickeada.
   Nota: se considera que una impresión fue clickeada, si hay un click con el mismo id cuyo timestamp es lo sumo 2 horas mayor al de la impresión
   Se deben utilizar los primeros dos días de datos como datos de entrenamiento y el último día como test para evaluar su modelo.
2. Describa el modelo, algoritmo y metodología utilizada.
3. Presente brevemente los resultados obtenidos. Como los evaluó? Por qué?
4. Bonus: Cambiaría algo a su solución si la base de datos fuese 10 veces más grande que la provista? Que?

jampp

Outliers detection and APIs

Documentación de referencia: https://www.alphavantage.co/documentation/

El ejercicio consiste en trabajar con un dataset y está dividido en tres partes: obtener el dataset, visualizarlo y procesarlo.

1. Use Python para programaticamente descargar la daily_data del endpoint de *DIGITAL_CURRENCY_DAILY* y guardarla en un archivo local en el formato que le parezca más apropiado.
   La solución debería poder recibir la moneda y la ventana de tiempo como argumentos.
   Nota: la API no recibe fechas de inicio y fin, así que el filtro debe ser aplicado posteriormente.
   Endpoint de ejemplo:
   https://www.alphavantage.co/query?function=DIGITAL_CURRENCY_DAILY&symbol=BTC&market=CNY&apikey=demo
2. Lea los archivo(s) de salida del punto anterior y grafique los precios de apertura y cierre de los últimos 30 días.
   También imprima la media y la mediana de la diferencia entre precio de apertura y cierre. Explique cual considera más importante de las dos y por que.
3. Finalmente, detecte anomalías en la data. Podemos definir a un punto como una anomalía si es muy “atípico” o diferente respecto al resto de los puntos.
   Las anomalías deberían ser printeadas, graficadas o presentadas de alguna otra forma que considere adecuada.