Introducción a los Modelos Bayesianos con Python

En esta charla se exploran los conceptos clave de los modelos bayesianos y cómo implementarlos utilizando la librería Pymc3 de Python. Aprenderemos a entender la diferencia entre la estadística frecuentista y la estadística bayesiana, y cómo los modelos bayesianos pueden generar soluciones más intuitivas y prácticas.

Estadística Frecuentista vs. Estadística Bayesiana

Durante la charla el expositor remarca las siguientes diferencias entre la estadística frecuentista y la bayesiana.

1. La estadística frecuentista ve la probabilidad como la frecuencia de un evento.
2. La estadística bayesiana ve la probabilidad como una medida de incertidumbre.
3. Los tests de hipótesis frecuentistas pueden ser poco intuitivos, mientras que los modelos bayesianos ofrecen soluciones más prácticas.

Antes de seguir adelante, desmenucemos un poco el termino estadística frecuentista.

Estadística Frecuentista.

La estadística frecuentista es un enfoque que se basa en la observación empírica de eventos repetidos y su frecuencia relativa para determinar la probabilidad de que ocurran ciertos eventos. Se desarrolla a partir de los conceptos de probabilidad y se centra en el cálculo de probabilidades y los contrastes de hipótesis. En otras palabras, es la estadística "básica" que se aprende en cualquier curso de estadística.

Teorema de Bayes.

El teorema de Bayes es la base de los modelos bayesianos.

Relaciona la probabilidad del parámetro dado los datos (probabilidad posterior) con la probabilidad de los datos dado el parámetro (verosimilitud) y la probabilidad a priori del parámetro, en palabras mas sencillas, describe cómo actualizar la probabilidad de un evento a la luz de nueva evidencia.

¿Qué se puede hacer con Bayes?

1. El teorema de Bayes permite calcular la distribución de probabilidad del parámetro de interés.
2. El modelado bayesiano nos permite definir modelos utilizando información a priori y los datos observados.

Se plantea un ejemplo para usar la librería.

El gobierno quiere estimar la tasa de infección por COVID-19 en Santiago para decidir si implementar una estrategia de inmunidad de rebaño. Se utilizarán pruebas de detección de anticuerpos para este análisis. En este ejemplo se utilizará la distribución Binomial.

Distribución binomial.

La distribución binomial es muy utilizada en estadística, ya que esta describe la probabilidad de obtener éxitos (1) o fracasos (0) en un experimento con un número fijo de ensayos.

Por ejemplo: Lanzar una moneda 10 veces y obtener 3 caras.

Modelado de los datos de las pruebas.

Los resultados de las pruebas de detección de anticuerpos pueden modelarse utilizando la distribución binomial. Un resultado positivo se considera un éxito, y un resultado negativo, un fracaso.

Existen tres modelos para estimar la tasa de infección.

- Modelo 1: Prueba perfecta.
  Asume que la prueba siempre dice la verdad. Con 40 resultados negativos y 10 positivos, el mejor estimado es que el 20% de la población ha sido infectada.
- Modelo 2: Prueba con falsos positivos.
  Considera la posibilidad de que algunos de los 10 resultados positivos sean falsos positivos. Requiere estimar la tasa de falsos positivos.
- Modelo 3: Incertidumbre en la tasa de infección.
  Reconoce que no se conoce con certeza la tasa de infección. Utiliza una distribución de probabilidad para modelar la incertidumbre en la tasa de infección.

Dicho lo anterior, procederemos a programar los modelos con PyMC3.

Antes de pasar a esto, se dará una breve introducción sobre esta librería.

PyMC3.

Es un paquete de Python para el modelado estadístico bayesiano y el aprendizaje automático probabilístico. Ofrece varios algoritmos de inferencia principalmente métodos Monte Carlo vía cadena de Markov e Inferencia Variacional.

Dicho lo anterior pasamos a la explicación del código.

Se explicará grosso modo el código del primer modelo. Modelo de prueba perfecta.

```python
import pymc3 as pm
import arviz as az
```

1

```python
tests_totales = 50
tests_positivos = 10
```

2

```python
with pm.Model() as modelo_test_perfecto:
    prob = pm.Uniform(name='prob',
                      lower=0,
                      upper=1)
    casos_positivos = pm.Binomial(name='casos_positivos',
                                  p=prob,
                                  n=tests_totales,
                                  observed=tests_positivos)
    trace_test_perfecto = pm.sample(3000)
```

3

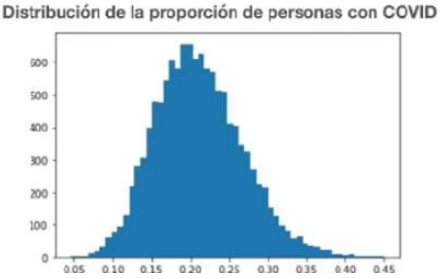
1. Se importa la librería pymc3 y arviz
2. Definimos los datos
3. Definimos la variable (prob) que queremos estimar y le damos una probabilidad a priori. Además se define la distribución de los datos (casos_positivos), que habíamos dicho seria binomial.
   La librería se manda llamar en pm.sample(3000), aquí se aplica toda la matemática, sacando tres mil muestras

Corremos el programa y obtenemos lo siguiente:

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [prop]
100.00% [16000/16000 00:02<00:00 Sampling 4 chains, 0 divergences]
```

```python
trace_test_perfecto.get_values(varname='prop')
```

```
array([0.15966427, 0.15410507, 0.14834236, ..., 0.23318682, 0.2177276 ,
       0.18031892])
```



Obtenido esto podemos obtener respuesta a algunas de nuestras preguntas como, por ejemplo: ¿Cuál es la probabilidad de que menos del 15% de la población se haya infectado?

Para responder esta pregunta corremos la siguiente línea de código:
muestras_prop = trace_test_perfecto.get_values(varname='prob')
len(muestras_prop[muestras_prop<0.15])/len(muestras_prop)

Dando como resultado: *0.134333*

Al final se pueden concluir dos cosas muy importantes:

1. Con PyCM3 se pueden hacer modelos muy robustos sin matemática compleja.
2. Se puede incorporar toda la incertidumbre que tenemos.