Modelos Bayesianos

Estadística bayesiana

Está basada en el teorema de Bayes y se diferencia de la estadística frecuentista básicamente en la incorporación de información externa al estudio que se esté realizando.

Estadística frecuentista

Se desarrolla a partir de los conceptos de probabilidad y que se centra en el cálculo de probabilidades y los contrastes de hipótesis. Tiene como objetivo determinar una conclusión en base a significación estadística o aceptación y rechazo de hipótesis siempre dentro del marco del estudio que se esté realizando.

Teorema de Bayes

Tiene una relación directa con la probabilidad de un determinado suceso a partir de todos los sucesos posibles. Lo que es lo mismo permite obtener la probabilidad de un evento en función de unos valores previos determinados.

$P(A|B) = P(B|A) * P(A) / P(B)$

Donde:

- P(Parametros|Datos): Lo que podemos calcular, probabilidad posterior.
- P(Datos|Parametros): Probabilidad de ver los datos que vimos en casos hipotéticos.
- P(Parámetros): Conocimiento a priori.

Distribución binomial

Describe el número de veces que se produce un éxito en un conjunto de ensayos. Hay N "experimentos" que resultan en un éxito (un 1) o un fracaso (un 0) con probabilidad p.

PyMC

PyMC es una librería de Python usando una sintaxis intuitiva para generar el modelado estadístico bayesiano y el aprendizaje automático probabilístico.

Ejemplo

En mayo del 2020 hace unos meses empezó la pandemia. El gobierno quiere saber cuánta gente se ha contagiado de covid en Santiago, Chile. Dependiendo de este número, el gobierno seguirá una estrategia de inmunidad de rebaño. Para esto usaremos test que detectan anticuerpos de SARS-CoV2.

Datos

Tomaremos una muestra aleatoria de 50 personas. 40 son negativas (el test no detecta anticuerpos) y 10 son positivas (el test detecta anticuerpos).

Generar modelos

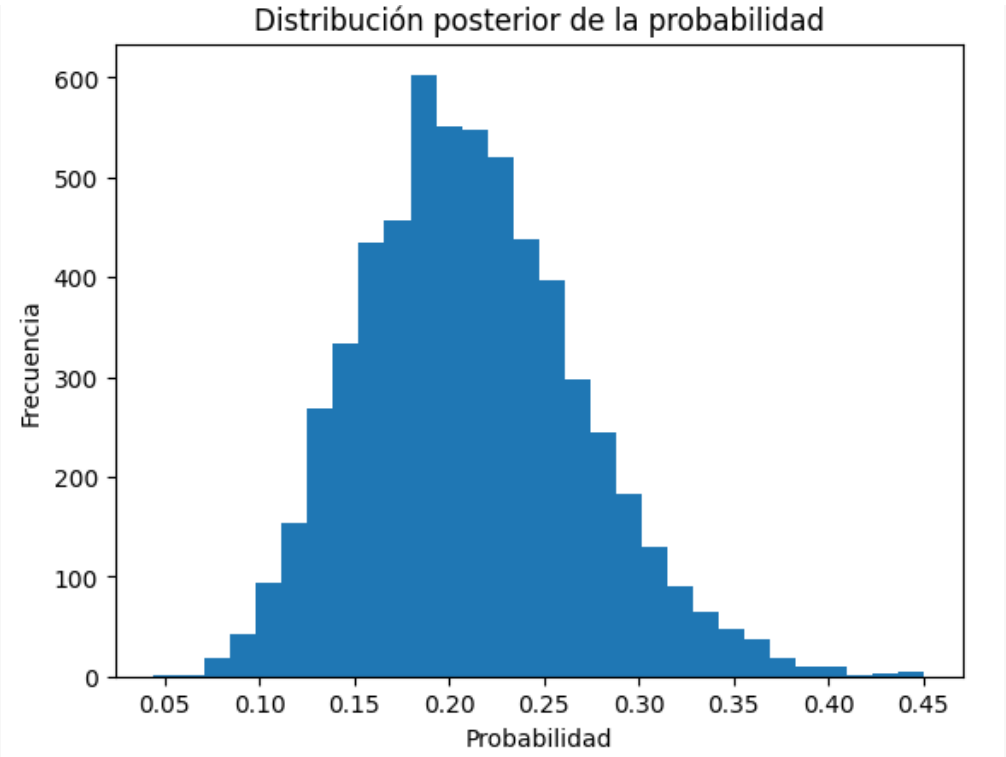
Modelo 1: Asume que el test es perfecto.

Modelo 2: El test a veces da falsos positivos.

Modelo 3: Incertidumbre sobre la tasa de falsos positivos.

Modelo 1

```
#Importación de librerías
import pymc as pm
import arviz as az
import numpy as np
import matplotlib.pyplot as plt

#Definición de las variables
tests_totales = 50
tests_positivos = 10

#Creación del modelo probabilístico
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)

    # Muestreo del modelo
    trace_test_perfecto = pm.sample(3000)

posterior = trace_test_perfecto.posterior

#Conversión de la distribución posterior a un arreglo NumPy
muestras_prop = np.array(posterior['prob'].stack(sample=('chain',
'draw')))

#Cálculo de la proporción de muestras mayores a 0.15
proporcion = len(muestras_prop[muestras_prop > 0.15]) /
len(muestras_prop)
print(proporcion)

# Visualización de la distribución posterior
plt.hist(muestras_prop, bins=30)
plt.xlabel('Probabilidad')
plt.ylabel('Frecuencia')
plt.title('Distribución posterior de la probabilidad')
plt.show()
```



Con esto podemos calcular la probabilidad de que sucedada x evento

```
# Calcula la proporción de muestras mayores a 0.15
proporcion = len(muestras_prop[muestras_prop > 0.15]) /
len(muestras_prop) print(proporcion)
```

Probabilidad de que al menos el 15% de la población haya sido infectoda?
0.1343333

Modelo 2
Considerar que la proporción de test positivos no es el mismo que la proporción de gente que covid.

Ajustamos el código

```
with pm.Model() as modelo_con_fp:
    prob_cov = pm.Uniform(name='prob_cov', lower=0, upper=1)

    prob_fp = 0.1
    prob_test_positivo = prob_cov + (1 - prob_cov) * prob_fp

    casos_positivos = pm.Binomial(
        name='casos_positivos',
        p=prob_test_positivo,
```

```
        n=tests_totales,
        observed=tests_positivos
    )

    modelo_con_fp = pm.sample(3000)
```

```
# Calcula la proporción de muestras mayores a 0.15
```

```
muestras_prop=modelo_con_fp.get_values(varname='prob_cov')
len(muestras_prop[muestras_prop<0.15])/len(muestras_prop)
```

Probabilidad de que al menos el 15% de la población haya sido infectoda?

0.67

Modelo 3

Modelar la tasa de falso positivo con una distribución binomial.

```
# Datos observados
lab_fp_observados = 10
lab_tests_hechos = 100
# Definición del modelo
with pm.Model() as modelo_con_incertidumbre:
     Modelo para estimar la tasa de falsos positivos
     prob_fp = pm.Uniform(name='prob_fp', lower=0, upper=1)
    # Distribución uniforme para la probabilidad de falsos positivos
entre 0 y 1
test_de_falsos_positivos=pm.Binomial(name='test_de_falsos_positivos',
                                     p=prob_fp,
                                     n=lab_tests_hechos,
                                     observed=lab_fp_observados)
    # Distribución binomial para modelar los falsos positivos
observados

    # Modelo para calcular la proporción de personas con COVID
    prob_cov = pm.Uniform(name='prob_cov', lower=0, upper=1)
    # Distribución uniforme para la proporción de personas con COVID
entre 0 y 1

    prob_test_positivo = prob_cov + (1 - prob_cov) * prob_fp
    # Cálculo de la probabilidad total de un test positivo,
considerando la tasa de falsos positivos

    casos_positivos = pm.Binomial(name='casos_positivos',
                                  p=prob_test_positivo,
                                  n=tests_totales,
```

```
                                    observed=tests_positivos)
    # Distribución binomial para modelar los casos positivos
observados

    trace_modelo_con_incertidumbre = pm.sample(3000)
    # Muestra del posterior usando 3000 iteraciones
```

Resultado de los tres métodos