{desafío}
latam_

Naive Bayes _

Sesión Presencial 1

Itinerario

Activación de conceptos

Desarrollo Desafío

Panel de discusión

{desafío}
latam_

Activación de conceptos

¿Qué problema resuelve GAM?

- El trueque entre sesgo y varianza existente en un modelo mediante inclusión de términos polinomiales.
- El trueque entre sesgo y varianza existente en un modelo mediante la inclusión de funciones de identidad.
- El problema de parámetros estimados muy grandes.

¿Cómo se obtienen las funciones de identidad en GAM?

- Se definen por el criterio de investigador.
- Mediante backfitting.
- Mediante la inclusión de información a priori en los parámetros.

¿Efecto de λ ?

- Regula el ancho de banda de la función de identidad.
- Penaliza la reducción del error cuadrático promedio.
- Ninguna de las anteriores.

¿Qué genera el gráfico de dependencia parcial?

- Evalúa la función de identidad para cada uno de los atributos.
- El efecto del parámetro en las predicciones.
- La varianza explicada por cada atributo.

Teorema de Bayes

Sabemos Pr(A|B), pero queremos Pr(B|A).
El Teorema de Bayes resuelve:

$$\Pr(\text{A posteriori}) = \frac{\Pr(\text{Verosimilitud}) \times \Pr(\text{A priori})}{\Pr(\text{Evidencia})}$$

Mantra: La probabilidad a posteriori es proporcional a la verosimilitud por la probabilidad a priori, ajustada por la evidencia.

Ejemplo: Confundiendo sonrisas

¿Qué pasa cuando una persona te sonríe cada vez que te ve? Esta es una de las incógnitas más grandes de la humanidad.

- Asumamos que sabemos que hay una probabilidad conjunta del 95% que cuando una persona le guste a alguien, sonría.
- También sabemos que existe un 10% de probabilidad que le sonría a un extraño y un 1% que le guste alguien al azar.

Identifiquemos los elementos:

1. A posteriori: ¿Le gusto a alguien?
2. Verosimilitud: ¿Me sonríe porque le gusto?
3. A priori: ¿Cual es la probabilidad que le guste alguien al azar?
4. Evidencia: ¿Sonríe al azar?

Implementación básica con Python

```
In [1]: def bayes_solver(likelihood = .95, prior = 0.01, evidence = 0.1):
            return round(likelihood * prior / evidence, 3)

In [2]: bayes_solver()

Out[2]: 0.095
```

¿Qué sucede si sabemos que sonríe mucho?

```
In [3]: bayes_solver(evidence=.20)

Out[3]: 0.047
```

¿Qué pasa si sabemos que se enamora fácilmente?

```
In [4]: bayes_solver(prior=0.2, evidence=.2)
Out[4]: 0.95
```

Naïve Bayes (Bayes Ingenuo)

Es un algoritmo generativo $\Pr(y|x) \rightsquigarrow \Pr(x, y)$ dado $x \in \mathbf{X}^{\mathbb{R}}$, e $y \in \mathbb{Y}$.

Objetivo:
Identificar la pertenencia de una observación a una clase específica.

Buscamos resolver:

$$\hat{y}_{\text{map}} = \underset{y \in \mathbb{Y}}{\text{argmax}}\ \hat{\Pr}(y|X) = \underset{y \in \mathbb{Y}}{\text{argmax}}\ \hat{\Pr}(y) \prod_{1 \leq k \leq n_d} \Pr(X_k|y)$$

Algoritmo

L ecuación anterior es difícil de trabajar. Generalmente trabajamos la suma del logaritmo:

$$\hat{y}_{\text{map}} = \underset{y \in Y}{\text{argmax}} \left[\log \hat{\Pr}(y) + \sum_{1 \leq k \leq n_d} \log \hat{\Pr}(X_k | y) \right]$$

Al ingresar una observación específica, evaluamos su verosimilitud en cada clase y asignamos aquella con una mayor probabilidad de ocurrencia

Implementación con sklearn

- Implementamos `sklearn.naive_bayes.BernoulliNB` → Atributos binarios
- `sklearn.preprocessing.LabelEncoder` nos permite recodificar atributos a numéricos.
- `sklearn.model_selection.train_test_split` divide la muestra.

Preprocesamiento

In [6]:
```
from sklearn.naive_bayes import BernoulliNB
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

# encoding numérico
df_deagg['Gender'] = LabelEncoder().fit_transform(df_deagg['Gender'])
df_deagg['Admit'] = LabelEncoder().fit_transform(df_deagg['Admit'])

# generamos una serie de dummies
df_deagg = pd.concat([df_deagg, pd.get_dummies(df_deagg['Dept'], prefix='dept')], axis=1)
.drop(columns='Dept')
```

Montando el modelo

In [7]:

```
X_train_mat, X_test_mat, y_train_vec, y_test_vec = train_test_split(df_deagg.loc[:, 'Gend
er':'dept_F'],df_deagg['Admit'],test_size=.30, random_state=11238)

# Hacemos Fit
nb_classifier = BernoulliNB().fit(X_train_mat, y_train_vec)
# Generamos predicciones en log(pr), pr y class
nb_log_prob_pred = nb_classifier.predict_log_proba(X_test_mat)
nb_prob_pred = nb_classifier.predict_proba(X_test_mat)
nb_class_pred = nb_classifier.predict(X_test_mat)
```

¿Qué son estas predicciones?

```
In [8]:
# log(Pr)
nb_log_prob_pred[:5].T
```

```
Out[8]:
array([[-0.46562497, -1.38216456, -1.38216456, -1.38216456, -0.2710045
6],
       [-0.9881699 , -0.28906247, -0.28906247, -0.28906247, -1.4380636
5]])
```

```
In [9]:
# Pr
nb_prob_pred[:5].T
```

```
Out[9]:
array([[0.62774266, 0.25103458, 0.25103458, 0.25103458, 0.76261302],
       [0.37225734, 0.74896542, 0.74896542, 0.74896542, 0.23738698]])
```

```
In [10]:
# argmax(Pr)
nb_class_pred[:5].T
```

```
Out[10]:
array([0, 1, 1, 1, 0])
```

Métricas: Matriz de confusión

```
In [12]: from sklearn.metrics import confusion_matrix; plt.figure(figsize=(10, 5));cnf_mat()
```

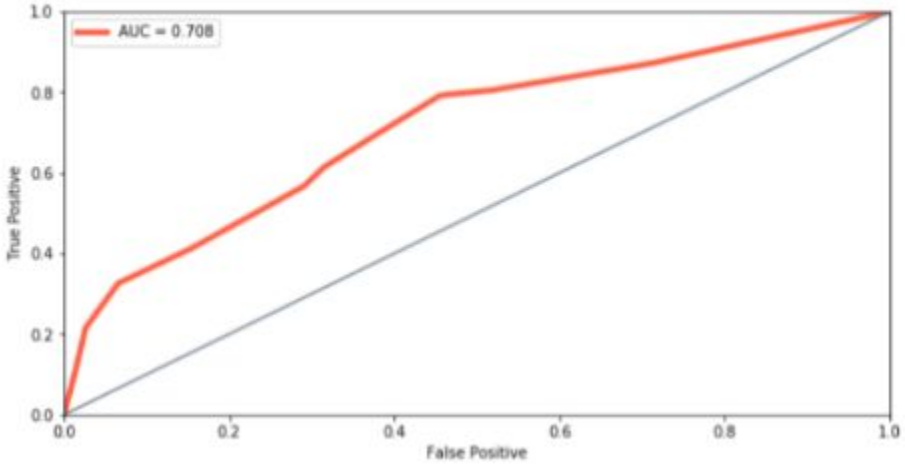
Métricas: Classification Report

In [13]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test_vec, nb_class_pred))
```

	precision	recall	f1-score	support
0	0.63	0.55	0.58	533
1	0.73	0.79	0.76	825
avg / total	0.69	0.69	0.69	1358

Métricas: ROC-AUC

```
In [15]: from sklearn.metrics import roc_curve, roc_auc_score; plt.figure(figsize=(10,5));roc_auc_
plot()
```



Efecto de la información a priori en nuestro a posteriori

In [16]:
```
# uninformative
gfx.compare_priors(X_train=X_train_mat,X_test=X_test_mat,
                   y_train=y_train_vec,y_test=y_test_vec,
                   prior=[0.39, 0.61])
```

```
A priori: [0.39, 0.61]
Accuracy: 0.694
Recall: 0.79
Precision: 0.729
F1: 0.759
AUC: 0.708
```

In [17]:
```
# berkeley 2015
gfx.compare_priors(X_train=X_train_mat,X_test=X_test_mat,
                   y_train=y_train_vec,y_test=y_test_vec,
                   prior=[.17, .83])
```

```
A priori: [0.17, 0.83]
Accuracy: 0.643
Recall: 0.872
Precision: 0.655
F1: 0.748
AUC: 0.708
```

/* Desafío */

Panel de discusión

{desafío} latam_

Academia de talentos digitales

www.desafiolatam.com