¿Comprar casa o Alquilar? Naive Bayes usando Python

Ejercicio práctico, intentando llevar los algoritmos de Machine Learning a ejemplos claros y de la vida real, repasaremos la teoría del Teorema de Bayes (video) de estadística para poder tomar una decisión muy importante: **¿me conviene comprar casa ó alquilar?**

Veamos si la Ciencia de Datos nos puede ayudar a resolver el misterio... ¿Si alquilo estoy tirando el dinero a la basura? ó ¿Es realmente conveniente pagar una hipoteca durante el <<resto de mi vida>>?

Si bien tocaremos el tema livianamente -sin meternos en detalles como intereses de hipotecas variable/fija, porcentajes, comisiones de bancos,etc- haremos un planteo genérico para obtener resultados y tomar la mejor decisión dada nuestra condición actual.

Aalgoritmos Supervisados del Aprendizaje Automático que nos dejan clasificar datos y/o obtener predicciones o asistencia a la toma de decisiones (árbol de decisión, regresión logística y lineal, red neuronal). Por lo general esos algoritmos intentan minimizar algún tipo de coste iterando las entradas y las salidas y ajustando internamente las “pendientes” ó “pesos” para hallar una salida. Esta vez, el algoritmo que usaremos se basa completamente en teoría de probabilidades y obteniendo resultados estadísticos. ¿Será suficiente el Teorema de Bayes para obtener buenas decisiones? Veamos!

¿Qué necesitaras para programar?

Para realizar este ejercicio, crearemos una Jupyter notebook con código Python y la librería SkLearn muy utilizada en Data Science. Recomendamos utilizar la suite para Python de Anaconda. Puedes leer este artículo donde muestro paso a paso como instalar el ambiente de desarrollo. Podrás descargar los archivos de entrada csv o visualizar la notebook online (al final de este artículo los enlaces).

¿Comprar casa o Alquilar? Naive Bayes usando Python

Nuestros Datos de Entrada:

Importemos las librerías que usaremos y visualicemos la información que tenemos de entrada:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import colors
import seaborn as sb

%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.naive_bayes import GaussianNB
from sklearn.feature_selection import SelectKBest
```

cargamos lo datos desde el dataset csv

```
dataframe = pd.read_csv(r"comprar_alquilar.csv")
dataframe.head(10)
```

click to expand output; double click to hide output

	ingresos	gastos_comunes	pago_coche	gastos_otros	ahorros	vivienda	estado_civil	hijos	trabajo	comprar
0	6000	1000	0	600	50000	400000	0	2	2	1
1	6745	944	123	429	43240	636897	1	3	6	0
2	6455	1033	98	795	57463	321779	2	1	8	1
3	7098	1278	15	254	54506	660933	0	0	3	0
4	6167	863	223	520	41512	348932	0	0	3	1
5	5692	911	11	325	50875	360863	1	4	5	1
6	6830	1298	345	309	46761	429812	1	1	5	1
7	6470	1035	39	782	57439	606291	0	0	1	0
8	6251	1250	209	571	50503	291010	0	0	3	1
9	6987	1258	252	245	40611	324098	2	1	7	1

¿Comprar casa o Alquilar? Naive Bayes usando Python

Las columnas que tenemos son:

- ingresos: los ingresos de la familia mensual
- gastos comunes: pagos de luz, agua, gas, etc mensual
- pago coche: si se está pagando cuota por uno o más coches, y los gastos en combustible, etc al mes.
- gastos_otros: compra en supermercado y lo necesario para vivir al mes
- ahorros: suma de ahorros dispuestos a usar para la compra de la casa.
- vivienda: precio de la vivienda que quiere comprar esa familia
- estado civil:
    - 0-soltero
    - 1-casados
    - 2-divorciados
- hijos: cantidad de hijos menores y que no trabajan.
- trabajo:
    - 0-sin empleo 1-autónomo (freelance)
    - 2-empleado
    - 3-empresario
    - 4-pareja: autónomos
    - 5-pareja: empleados
    - 6-pareja: autónomo y asalariado
    - 7-pareja:empresario y autónomo
    - 8-pareja: empresarios los dos o empresario y empleado
- comprar: 0-No comprar 1-Comprar (esta será nuestra columna de salida, para aprender)

Algunos supuestos para el problema formulado:

- Está pensado en Euros pero podría ser cualquier otra moneda
- No tiene en cuenta ubicación geográfica, cuando sabemos que dependerá mucho los precios de los inmuebles de distintas zonas
- Se supone una hipoteca fija a 30 años con interés de mercado “bajo”.

Con esta información, queremos que el algoritmo aprenda y que como resultado podamos consultar nueva información y nos dé una decisión sobre comprar (1) o alquilar (0) casa.

¿Comprar casa o Alquilar? Naive Bayes usando Python

El teorema de Bayes

El teorema de Bayes es una ecuación que describe la relación de probabilidades condicionales de cantidades estadísticas. En clasificación bayesiana estamos interesados en encontrar la probabilidad de que ocurra una “clase” dadas unas características observadas (datos). Lo podemos escribir como P(Clase | Datos). El teorema de Bayes nos dice cómo lo podemos expresar en términos de cantidades que podemos calcular directamente:

$$P(\text{Clase} \mid \text{Datos}) = \frac{P(\text{Datos}|\text{Clase}) * P(\text{Clase})}{P(\text{Datos})}$$

- **Clase** es una salida en particular, por ejemplo “comprar”
- **Datos** son nuestras características, en nuestro caso los ingresos, gastos, hijos, etc
- **P(Clase|Datos)** se llama posterior (y es el resultado que queremos hallar)
- **P(Datos|Clase)** se llama “verosimilitud” (en inglés likelihood)
- **P(Clase)** se llama anterior (pues es una probabilidad que ya tenemos)
- **P(Datos)** se llama probabilidad marginal

¿Comprar casa o Alquilar? Naive Bayes usando Python

Si estamos tratando de elegir entre dos clases como en nuestro caso “comprar” ó “alquilar”, entonces una manera de tomar la decisión es calcular la tasa de probabilidades a posterior:

$$\frac{P(\text{Comprar} \mid \text{Datos})}{P(\text{Alquilar} \mid \text{Datos})} = \frac{P(\text{Datos}|\text{Comprar}) * P(\text{Comprar})}{P(\text{Datos}|\text{Alquilar}) * P(\text{Alquilar})}$$

con esta maniobra, nos deshacemos del denominador de la ecuación anterior P(Datos) el llamado “probabilidad marginal”.

¿Comprar casa o Alquilar? Naive Bayes usando Python

Clasificador Gaussian Naive Bayes

Uno de los tipos de clasificadores más populares es el llamado en inglés Gaussian Naive Bayes Classifier. NOTA:Hay otros clasificadores Bayesianos que no veremos en este artículo. Veamos cómo es su fórmula para comprender este curioso nombre: aplicaremos 2 clases (comprar, alquilar) y tres características: ingresos, ahorros e hijos.

$$\text{posterior(comprar)} = \frac{P(\text{comprar})p(\text{ingresos}|\text{comprar})p(\text{ahorros}|\text{comprar})p(\text{hijos}|\text{comprar})}{\text{probabilidad marginal}}$$

$$\text{posterior(alquilar)} = \frac{P(\text{alquilar})p(\text{ingresos}|\text{alquilar})p(\text{ahorros}|\text{alquilar})p(\text{hijos}|\text{alquilar})}{\text{probabilidad marginal}}$$

Posterior de comprar es lo que queremos hallar: P(comprar|datos).

Explicaremos los demás:

- **P(comprar)** es la probabilidad que ya tenemos. Es sencillamente el número de veces que se selecciona comprar =1 en nuestro conjunto de datos, dividido el total de observaciones. En nuestro caso (luego lo veremos en Python) son 67/202

¿Comprar casa o Alquilar? Naive Bayes usando Python

- **p(ingresos|comprar)p(ahorros|comprar)p(hijos|comprar)** es la verosimilitud. Los nombres Gaussian y Naive (ingenuo) del algoritmo vienen de dos suposiciones:
  1. asumimos que las características de la verosimilitud no estan correlacionada entre ellas. Esto sería que los ingresos sean independientes a la cantidad de hijos y de los ahorros. Como no es siempre cierto y es una suposición ingenua es que aparece en el nombre “naive bayes”
  2. Asumimos que el valor de las características (ingresos, hijos, etc) tendrá una distribución normal (gaussiana). Esto nos permite calcular cada parte p(ingresos|comprar) usando la función de probabilidad de densidad normal.
- **probabilidad marginal** muchas veces es difícil de calcular, sin embargo, por la ecuación que vimos más arriba, no la necesitaremos para obtener nuestro valor a posterior. Esto simplifica los cálculos.

Bien!, Fin de teoría, sigamos con el ejercicio! Ahora toca visualizar nuestras entradas y programar un poquito.

Visualización de Datos

Veamos qué cantidad de muestras de comprar o alquilar tenemos:

```
print(dataframe.groupby('comprar').size())
```

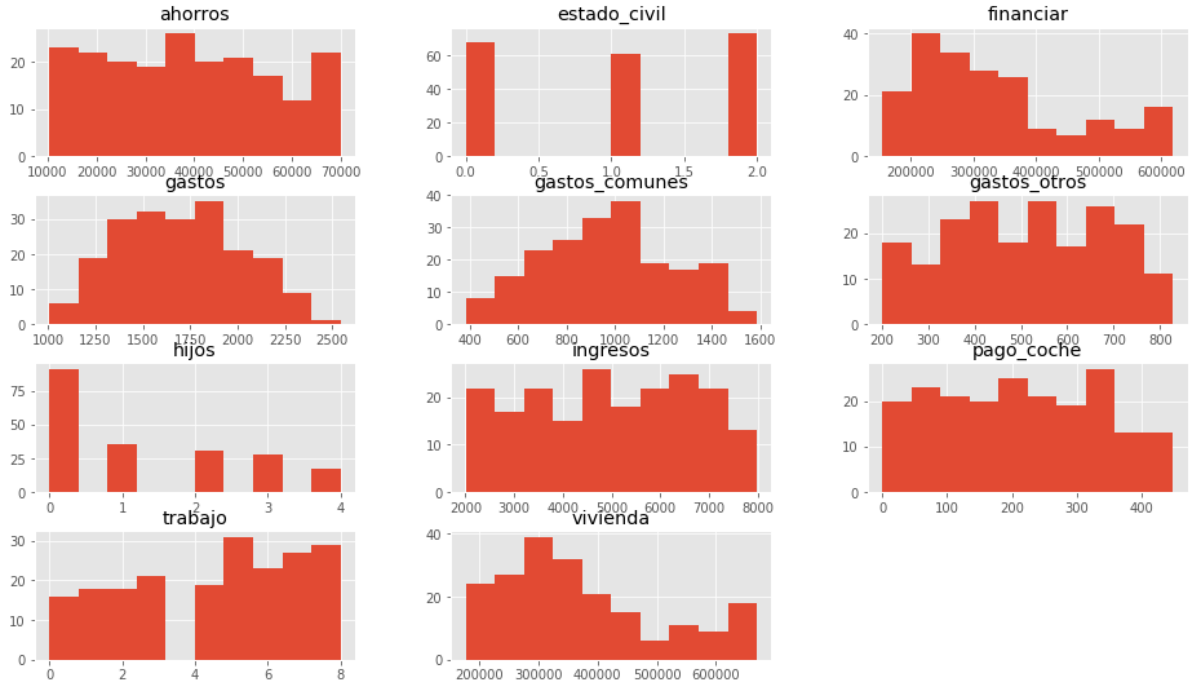
```
comprar
0    135
1     67
dtype: int64
```

Esto son 67 que entradas en las que se recomienda comprar y 135 en las que no.

¿Comprar casa o Alquilar? Naive Bayes usando Python

@mihifidem
creativity is intelligence having fun

Hagamos un histograma de las características quitando la columna de resultados (comprar):



Pareciera a grandes rasgos que la distribución de hijos e ingresos <<se parece>> un poco a una distribución normal.

¿Comprar casa o Alquilar? Naive Bayes usando Python

Preparar los datos de entrada

Vamos a hacer algo: procesemos algunas de estas columnas. Por ejemplo, podríamos agrupar los diversos gastos. También crearemos una columna llamada financiar que será la resta del precio de la vivienda con los ahorros de la familia.

```
dataframe['gastos']=(dataframe['gastos_comunes']+dataframe['gastos_otros']
+dataframe['pago_coche'])
dataframe['financiar']=dataframe['vivienda']-dataframe['ahorros']
dataframe.drop(['gastos_comunes','gastos_otros','pago_coche'], axis=1).head(10)
```

	ingresos	ahorros	vivienda	estado_civil	hijos	trabajo	comprar	gastos	financiar
0	6000	50000	400000	0	2	2	1	1600	350000
1	6745	43240	636897	1	3	6	0	1496	593657
2	6455	57463	321779	2	1	8	1	1926	264316
3	7098	54506	660933	0	0	3	0	1547	606427
4	6167	41512	348932	0	0	3	1	1606	307420
5	5692	50875	360863	1	4	5	1	1247	309988
6	6830	46761	429812	1	1	5	1	1952	383051
7	6470	57439	606291	0	0	1	0	1856	548852
8	6251	50503	291010	0	0	3	1	2030	240507
9	6987	40611	324098	2	1	7	1	1755	283487

Y ahora veamos un resumen estadístico que nos brinda la librería Pandas con describe():

```
reduced = dataframe.drop(['gastos_comunes','gastos_otros','pago_coche'], axis=1)
reduced.describe()
```

¿Comprar casa o Alquilar? Naive Bayes usando Python

	ingresos	ahorros	vivienda	estado_civil	hijos	trabajo	comprar	gastos	financiar
count	202.000000	202.000000	202.000000	202.000000	202.000000	202.000000	202.000000	202.000000	202.000000
mean	4958.995050	38749.668317	373349.638614	1.024752	1.232673	4.490099	0.331683	1698.752475	334599.970297
std	1682.862556	17365.231870	136371.525622	0.837184	1.367833	2.535794	0.471988	324.838005	126607.099497
min	2008.000000	10319.000000	176553.000000	0.000000	0.000000	0.000000	0.000000	1007.000000	154716.000000
25%	3513.750000	24964.250000	274810.000000	0.000000	0.000000	2.000000	0.000000	1430.500000	240410.250000
50%	4947.500000	38523.000000	340783.500000	1.000000	1.000000	5.000000	0.000000	1669.500000	301177.000000
75%	6374.500000	52150.750000	444482.000000	2.000000	2.000000	7.000000	1.000000	1928.000000	393413.000000
max	7984.000000	69934.000000	669540.000000	2.000000	4.000000	8.000000	1.000000	2543.000000	618621.000000

Feature Selection ó Selección de Características

En este ejercicio haremos Feature Selection para mejorar nuestros resultados con este algoritmo. En vez de utilizar las 11 columnas de datos de entrada que tenemos, vamos a utilizar una Clase de SkLearn llamada SelectKBest con la que seleccionaremos las 5 mejores características y usaremos sólo esas.

```
X=dataframe.drop(['comprar'], axis=1)
y=dataframe['comprar']

best=SelectKBest(k=5)
X_new = best.fit_transform(X, y)
X_new.shape
selected = best.get_support(indices=True)
print(X.columns[selected])
```

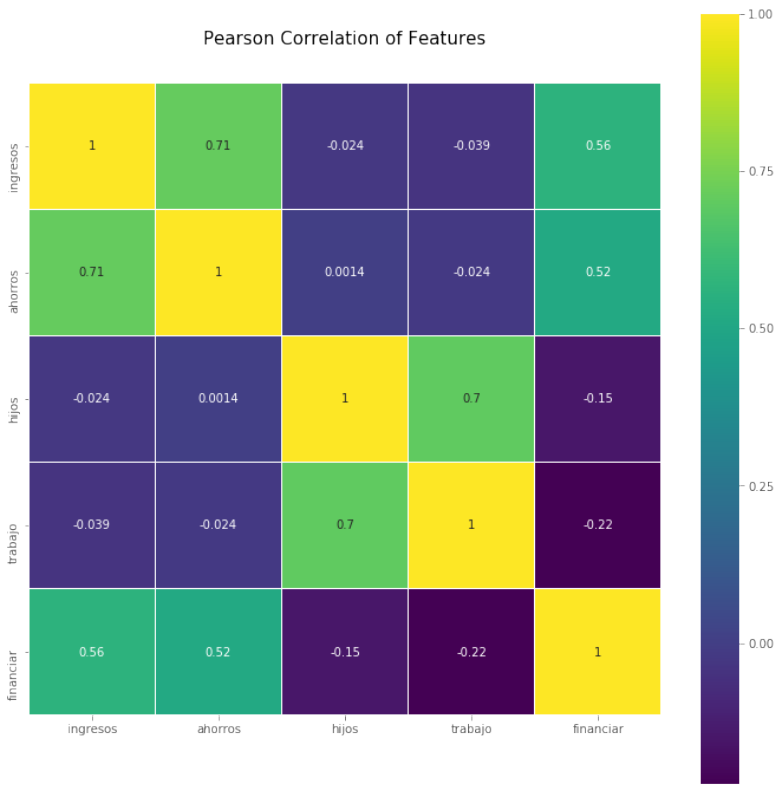
```
Index([‘ingresos’, ‘ahorros’, ‘hijos’, ‘trabajo’, ‘financiar’], dtype=’object’)
```

Bien, entonces usaremos 5 de las 11 características que teníamos. Las que “más aportan” al momento de clasificar. Veamos qué grado de correlación tienen:

```
used_features =X.columns[selected]

colormap = plt.cm.viridis
plt.figure(figsize=(12,12))
plt.title('Pearson Correlation of Features', y=1.05, size=15)
sb.heatmap(dataframe[used_features].astype(float).corr(),linewidths=0.1,vmax=1.0,
square=True, cmap=colormap, linecolor='white', annot=True)
```

¿Comprar casa o Alquilar? Naive Bayes usando Python



Con esto comprobamos que en general están poco correlacionadas, sin embargo también tenemos 2 valores de 0,7. Esperemos que el algoritmo sea lo suficientemente “naive” para dar buenos resultados

Otra alternativa para Feture Selection es utilizar Principal Component Analysis (PCA) y hacer reducción de Dimensión

¿Comprar casa o Alquilar? Naive Bayes usando Python

Crear el modelo Gaussian Naive Bayes con SKLearn

Primero vamos a dividir nuestros datos de entrada en entrenamiento y test.

```
# Split dataset in training and test datasets
X_train, X_test = train_test_split(dataframe, test_size=0.2, random_state=6)
y_train =X_train["comprar"]
y_test = X_test["comprar"]
```

Y creamos el modelo, lo ponemos a aprender con fit() y obtenemos predicciones sobre nuestro conjunto de test.

```
# Instantiate the classifier
gnb = GaussianNB()
# Train classifier
gnb.fit(
    X_train[used_features].values,
    y_train
)
y_pred = gnb.predict(X_test[used_features])

print('Precisión en el set de Entrenamiento: {:.2f}'
     .format(gnb.score(X_train[used_features], y_train)))
print('Precisión en el set de Test: {:.2f}'
     .format(gnb.score(X_test[used_features], y_test)))
```

```
Precisión en el set de Entrenamiento: 0.87
Precisión en el set de Test: 0.90
```

Pues hemos obtenido un bonito 90% de aciertos en el conjunto de Test con nuestro querido clasificador bayesiano.

¿Comprar casa o Alquilar? Naive Bayes usando Python

Probemos el modelo: ¿Comprar o Alquilar?

Ahora, hagamos 2 predicciones para probar nuestra máquina:

En un caso será una familia sin hijos con 2.000€ de ingresos que quiere comprar una casa de 200.000€ y tiene sólo 5.000€ ahorrados.

El otro será una familia con 2 hijos con ingresos por 6.000€ al mes, 34.000 en ahorros y consultan si comprar una casa de 320.000€.

```
#                ['ingresos', 'ahorros', 'hijos', 'trabajo', 'financiar']
print(gnb.predict([[2000,       5000,     0,       5,        200000],
                   [6000,       34000,    2,       5,        320000] ]))
#Resultado esperado 0-Alquilar, 1-Comprar casa
```

```
[0 1]
```

Los resultados son los esperados, en el primer caso, recomienda Alquilar (0) y en el segundo comprar la casa (1).

¿Comprar casa o Alquilar? Naive Bayes usando Python

Conclusiones

A lo largo del artículo repasamos el teorema de Bayes y vimos un ejemplo para aplicarlo en una toma de decisiones. Pero no olvidemos que en el proceso también hicimos pre procesamiento de los datos, visualizaciones y Selección de Características. Durante diversas charlas que tuve con profesionales del Data Science en mi camino de aprendizaje sale un mismo mensaje que dice: “No es tan importante el algoritmo a aplicar si no la obtención y pre procesamiento de los datos que se van a utilizar”. A tenerlo en cuenta!

Naive Bayes como clasificador se utiliza mucho en NLP (Natural Language Processing) tanto en el típico ejemplo de detectar “Spam” o no como en tareas más complejas como reconocer un idioma o detectar la categoría apropiada de un artículo de texto. También puede usarse para detección de intrusiones o anomalías en redes informáticas y para diagnósticos médicos dados unos síntomas observados. Por último veamos los pros y contras de utilizar Gaussian Naive Bayes:

Pros: Es rápido, simple de implementar, funciona bien con conjunto de datos pequeños, va bien con muchas dimensiones (features) y llega a dar buenos resultados aun siendo “ingenuo” sin que se cumplan todas las condiciones de distribución necesarias en los datos.

Contras: Requiere quitar las dimensiones con correlación y para buenos resultados las entradas deberían cumplir las 2 suposiciones de distribución normal e independencia entre sí (muy difícil que sea así ó deberíamos hacer transformaciones en lo datos de entrada).

¿Comprar casa o Alquilar? Naive Bayes usando Python