Naïve Bayes



Coneixement, Raonament i Incertesa

Enginyeria Informàtica 2021-2022

Martí Caixal Joaniquet - 1563587

Índex

[Introducció 3](#_Toc90738951)

[Dades 3](#_Toc90738952)

[Disseny i implementació 3](#_Toc90738953)

[Funció “Fit” 4](#_Toc90738954)

[Funció “Predict” 4](#_Toc90738955)

[Execució 5](#_Toc90738956)

[Mètrica i anàlisi 6](#_Toc90738957)

[Diferents mides de Train i diccionaris 7](#_Toc90738958)

[Laplace Smoothing 8](#_Toc90738959)

# Introducció

La pràctica tracta de dissenyar, programar i testejar un aprenentatge bayesià. Més concreatament, s’ha d’utilitzar l’algorisme “Naïve Bayes” per poder classificar un tweet entre positiu o negatius. Està dividida en 3 aparts:

1. Disseny i implementació de l’algorisme.
2. Modificar mides de diccionari i conjunt d’aprenentatge.
3. Efecte del Laplace Smoothing

# Dades

Les dades amb les que es treballa són un conjunt de tweets emmagatzemats a un arxiu csv. Es tracta de més d’un milió de mostres, cada una d’elles en el següent format:

*16; I fell in love again; 02/12/2015; 1*

El 16 és un identificador, la segona columna és el tweet en qüestió, la tercera la data de publicació i per últim l’etiqueta classificadora (0 si és negatiu, 1 si és positiu).

Estem davant d’un conjunt de dades balancejat, significant que es té el mateix nombre de mostres d’una classe que de l’altra.

# Disseny i implementació

Per poder implementar l’algorisme es fa ús d’un seguit de variables:

* wc (diccionari): Per cada classe, emmagatzema el número de vegades que una paraula apareix. Es tracta, doncs, de diccionaris anidats. A continuació es mostra un exemple del format

1. wc = {'positive': { 'hello' : 4,
2. 'car' : 7,
3. 'pencil' : 5
4. },
5. 'negative': { 'window': 2,
6. 'paper' : 10
7. }
8. }

* dictionary (set): emmagatzema cada paraula que es troba durant l’entrenament. Al tractar-se d’un set, ens assegurem que totes les paraules són úniques i no es repeteixen.
* log\_prior\_probability (diccionari): Guarda la probabilitat a priori que un tweet sigui positiu o negatiu.
* tweet\_num (diccionari): Guarda el nombre de tweets que hi ha positius i negatius.
* laplace\_smoothing (float): Conté el valor del Laplace Smoothing que s’utilitza durant la fase de predicció.

Tot i que moltes variables podrien ser simples llistes, el fet d’utilitzar diccionaris va bé per poder accedir directament al valor que es necessita.

L’algorisme està implementat dins d’una classe anomenada “NaiveBayes”. Les principals funcions que conté són “Fit” i “Predict”. La classe hereda de “sklearn.base.BaseEstimator” per tal de poder utilitzar el “cross\_validate” de “Sklearn”.

### Funció “Fit”

És la funció utilitzada durant l’entrenament del model. Primer omple les variables “tweet\_num” i “log\_prior\_probability”.

Seguidament fa els següents passos per cada tweet:

1. Separar tweet per paraules.
2. Mirar la classe a la que pertany el tweet.
3. Per cada paraula:
   1. Afegir-la al set “dictionary”.
   2. Incrementar el nombre d’aparicions per classe al diccionari “wc”.

### Funció “Predict”

És la funció utilitzada durant les prediccions de les mostres. Per cada mostra fa els següents passos:

1. Separar tweet per paraules.
2. Posar variables positive\_count i negative\_count a 0.
3. Per cada paraula:
4. Si no apareix al diccionari, continuar a la següent.
5. Calcular p(wi | ‘positive’) i p(wi | ‘negative’).
6. Sumar aquests valors a positive\_count i negative\_count
7. Afegir la probabilitat apriori de cada classe a positive\_count i negative\_count
8. Guardar com a resultat el la classe amb el valor més gran entre positive\_count i negative\_count

# Execució

El programa es pot executar amb Python des d’una línia de comandes. Abans, però, caldrà instal·lar totes les dependències necessàries:

python3 -m pip install -r requirements.txt

Seguidament es pot executar el programa:

python3 ./main.py

Addicionalment, també es pot executar amb arguments:

python3 ./main.py --smooth 0.1 --n\_rows 10000 --n\_splits 4

Es pot trobar una descripció de cada argument:

python3 main.py -h

# Mètrica i anàlisi

Al tractar-se d’un conjunt de mostres balancejat, les mètriques es simplifiquen bastant. Per exemple, el valor que s’obté d’accuracy és fiable. En unes dades no balancejades, el valor podria ser molt alt però les prediccions dolentes, i per tant s’hauria d’utilitzar F1-Score. El model que s’està creant només es vol per saber si un tweet és positiu o negatiu, sense donar més importància a un o l’alte. L’accuracy ens permet saber exactament això, el percentatge de prediccions ben fetes.

Si, per altra banda es volgués filtrar els tweets tal que només es mostrin els que són positius, s’hauria d’utilitzar una altra mètrica. La precission ens seria una bona mètrica, doncs calcula el percentatge de prediccions positives correctes d’entre el total de prediccions positives. Un precission alt permetria assegurar-nos que tots els tweets que passen el filtre són positius.

D’altre banda, si l’objectiu sigués permetre tants positius com sigui possible, sense importar els negatius que es colessin, s’hauria de mirar el Recall.

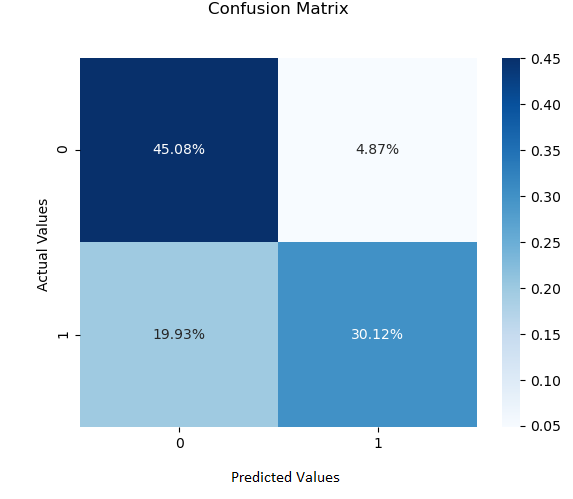
De totes formes, per aquesta pràctica ambdues classes són igual d’importants i es mirarà l’accuracy.

Les proves efectuades a la pràctica s’han realitzat amb un “K-Fold Cross Validation” de 10 folds. Per aconseguir la mètrica simplement es fa una mitjana de les 10 mètriques obtingudes. Al haver-hi més d’una predicció, no és suficient sabent la mitjana. Cal comprovar també que els resultats siguin consistents. Ajudant-se de la desviació estàndard de totes les mètriques es pot estar segur que són consistents.

Els resultats obtinguts són una Accuracy de 75% amb desviació estàndard de 0.001. Si bé no són excepcionalment bons, almenys les prediccions són consistents entre elles.

La matriu de confusió inferior mostra com estan distribuïdes les classificacions. Sumant els TP i TN 🡪 45+30 = 75 es pot veure que dona l’accuracy prèviament mencionada.

També es pot veure que és bastant més capaç de classificar correctament els tweets negatius que els positius.



*Cal anotar que la matriu està normalitzada tant per files com per columnes, sumant 1 entre totes les cel·les.*

# Diferents mides de Train i diccionaris

# Laplace Smoothing