


# Una nueva implementación del algoritmo Bag of Features (BoF) en Python

Aarón Narváez

Mayo 2019

## 1. Introducción

 Una serie de tiempo es una sucesión de observaciones enlistadas de acuerdo al momento de captura. Son útiles en áreas como la agricultura, donde se lleva un registro de la producción y su precio en el mercado. En la meteorología, que registra la velocidad de los vientos, las temperaturas máximas y mínimas, la cantidad de lluvia anual, entre otras. En la Geofísica se observa continuamente la vibración de las placas tectónicas con el propósito de predecir posibles temblores o terremotos. Los Electroencefalogramas trazan las ondas cerebrales con el fin de detectar enfermedades cerebrales, al igual que los electrocardiogramas trazan las ondas cardíacas con el fin de detectar enfermedades cardíacas. La tasa anual de muertes y nacimientos, y las diferentes formas de actividades criminales [1]. Existen una gran cantidad de procesos que pueden ser expresados en forma de serie de tiempo.

Las razones para analizar y almacenar esta información son muchas, sin embargo, se destaca el deseo de entender los **mecanismos que generan estas observaciones**, la predicción de observaciones futuras, o el control óptimo de un sistema [1].

Baydogan, Runger y Tuv en [2], proponen un framework para la clasificación de series de tiempo basado en bolsa de características (***A Bag-of-features Framework***, TSBF). Este método selecciona múltiples subsecuencias de longitud y posición aleatorias de una serie de tiempo, con el fin de extraer patrones que aparecen con diferentes longitudes y en diferentes posiciones.

En el presente reporte se describe **la implementación** de este método en lenguaje Python, **buscando la compatibilidad con los clasificadores disponibles en la suite de *scikit-learn*, y con el fin de extender la disponibilidad del método**. Además se presentan las pruebas estadísticas aplicadas a este y a la implementación presentada por Baydogan *et al* en lenguaje R, para determinar si se comportan de manera similar. Por último, se compara el desempeño del algoritmo al utilizar *Extratrees* como clasificador en cada una de las etapas.

## 2. Descripción del Método



**El método inicia creando subsecciones de longitud ( $l_s$ ) aleatoria, mayor a una longitud mínima ( $l_{min}$ ) y con posición aleatoria.** Estas subsecciones se dividen en  $d$  intervalos. Por

cada subsección se crea una instancia o vector de características que incluye la media ( $\mu_{dn}$ ), la varianza ( $\sigma_{dn}^2$ ) y la pendiente ( $m_{dn}$ ) de cada uno de los intervalos de la subsección, además incluye la media ( $\mu_s$ ), la varianza ( $\sigma_s^2$ ), el punto inicial ( $x_i$ ), y el punto final ( $x_f$ ), de la de toda la subsección. Cada instancia pertenece a la clase de la serie de la que fue extraída. Para determinar la longitud mínima de la subsección se utiliza una proporción de la longitud total de la serie ( $z \times T$ ), para ( $0 < z \leq 1$ ). Para determinar el número de intervalos ( $d$ ) para cada subsección, se relaciona la longitud mínima ( $l_{min}$ ) con la longitud mínima de intervalo ( $w_{min}$ ), esta última definida por el usuario (5 en las pruebas). La cantidad de posibles intervalos en una serie de tiempo ( $r$ ), es la relación entre la longitud total de la serie de tiempo ( $T$ ) y la longitud mínima de intervalo. Por cada subsección analizada existen  $r-d$  intervalos sin analizar, por lo que se crean  $r-d$  subsecciones (e instancias) para cada serie.

Estas instancias alimentan un clasificador (*RandomForest* es el elegido por Baydogan *et al* en [2]) que entrega la probabilidad de pertenencia a cada una de las clases. Estas probabilidades son agrupadas en  $b$  conjuntos. El *codebook* está conformado por  $C \times b$  elementos (que corresponden al total de conjuntos formados), donde  $C$  es el número de clases. Por cada instancias se obtienen  $C$  palabras, por lo que por cada serie se obtienen  $(r-d) \times C$  palabras. Estas palabras son utilizadas para generar el histograma de la serie de tiempo.

Los histogramas de las series de tiempo son utilizados para alimentar el segundo clasificador (*RandomForest*), que regresa la predicción de clase. En la etapa de entrenamiento se entrenan los clasificadores, mientras que en la etapa de pruebas solo se utilizan para obtener las probabilidades utilizadas para crear los histogramas, y la predicción de clase.

En la Figura 1, se muestra un diagrama del método. Este se puede resumir en 4 etapas: generación de instancias y extracción de características, clasificación y obtención de probabilidades de clases, crear bolsa de palabras e histogramas, y clasificación final.

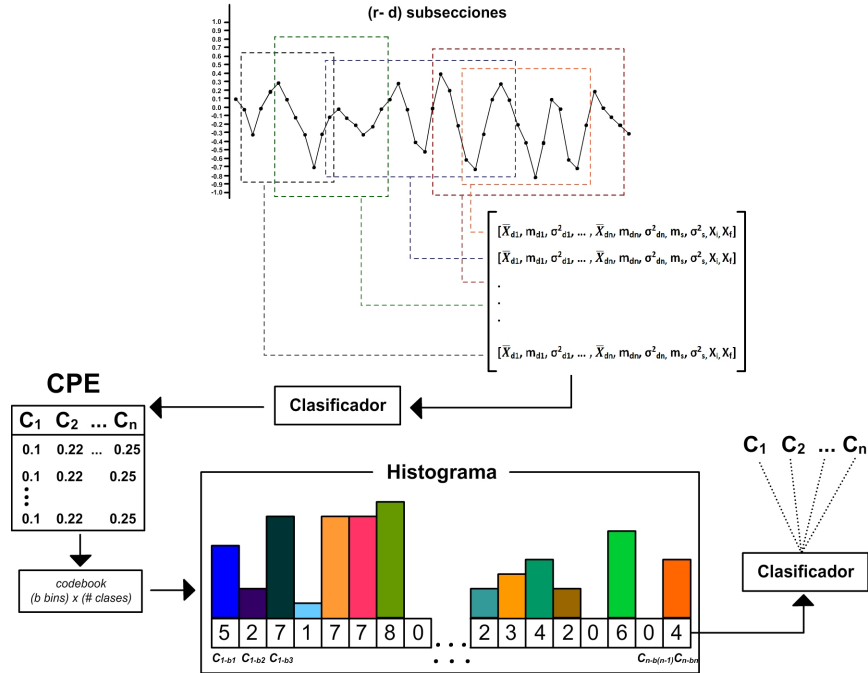


Figura 1: Diagrama de TSBF

## 2.1. Pseudocódigo

### Aprendizaje:

1. Dado un conjunto de series de tiempo para entrenamiento:

$$x^n = (x_1^n, x_2^n, \dots, x_T^n)$$

Donde  $T$  es longitud total de la serie  $x^n$ , y  $x_i^n$  la observación  $i$  para la serie de tiempo  $n$ . Cada serie de tiempo  $y^n$  para  $n = 1, 2, \dots, N$ , está asociada a una clase con  $y^n \in \{0, 1, 2, \dots, C - 1\}$ .

- 1.1: Para cada serie de tiempo crear  $r-d$  subsecciones de tamaño mayor a  $l_{min}$  y posición aleatoria, donde:

$$r = \frac{T}{w_{min}}, y \quad d = \frac{l_{min}}{w_{min}}$$

- 1.1.1: Para cada subsección crear  $d$  intervalos del mismo tamaño y mayores a  $w_{min}$ .

- 1.1.2: Para cada subsección crear un vector de características perteneciente a la clase de la serie de la que fue extraído, con los parámetros siguientes:

$$[\mu_{d1}, \sigma_{d1}^2, m_{d1}, \mu_{d2}, \sigma_{d2}^2, m_{d2}, \dots, \mu_{dn}, \sigma_{dn}^2, m_{dn}, \mu_s, \sigma_s^2, x_i, x_f]$$

Donde:

$\mu_{dn}$ , es la media del intervalo  $n$

$\sigma_{dn}^2$ , es la varianza del intervalo  $n$

$m_{dn}$ , es la pendiente del intervalo  $n$

$\mu_s$ , es la media de la subsección

$\sigma_s^2$ , es la varianza de la subsección

$x_i$ , es el punto inicial de la subsección

$x_f$ , es el punto final de la subsección

2. Entrenar un clasificador con el total de las instancias  $((r-d) \times N)$ , donde  $N$  es el número total de series de tiempo en el conjunto.

3. Obtener la probabilidad de pertenencia a cada de una de las clases para cada instancia

4. Agrupar las probabilidades en  $b$  conjuntos por cada clase.

5. Formar un histogramas para cada serie de tiempo con las palabras generadas.

El *codebook* se compone de  $b \times C$  palabras.

Cada instancia genera  $C$  palabras.

Cada serie de tiempo genera  $(r-d) \times C$  palabras.

6. Entrenar un clasificador con los histogramas generados.

#### Utilización:

1. Se sigue el mismo proceso que en el aprendizaje omitiendo el entrenamiento de los clasificadores

## 3. Pruebas

Para determinar si la implementación en Python se comporta de manera similar a la implementación en R, primero se inspeccionaron de manera visual los histogramas generados por ambos algoritmos. En la Figura 2, inciso (a) se muestran dos ejemplos de los histogramas generados en dos corridas diferentes para la misma serie de tiempo por cada uno de los algoritmos. Esta serie pertenece a la clase 2 del conjunto de datos “Beef”. De igual manera se presentan en el inciso (b) para una serie de tiempo diferente perteneciente al mismo conjunto de datos, y de la clase 3. Las diferencias se deben a la aleatoriedad del algoritmo.

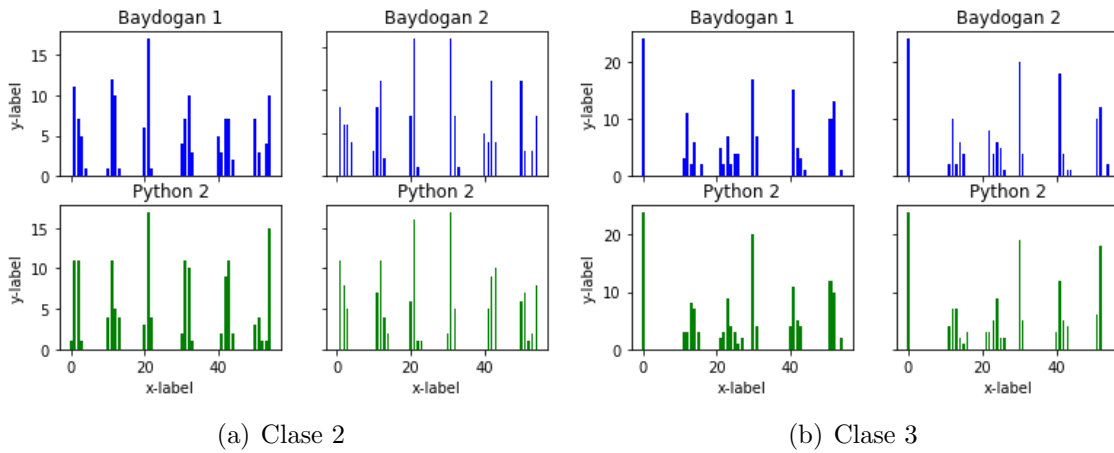


Figura 2: histogramas para el *dataset Beef*


## 4. Analisis Bayesiano

Benavoli *et al* en [3], presentan tres métodos de estimación Bayesiana que se pueden emplear para comparar el desempeño de dos algoritmos. Estos métodos, contrario a aquellos basados en la prueba de significación de la hipótesis nula (*Null Hypothesis Significance Testing*, NHST), son capaces de proveer la probabilidad de que un método sea mejor que otro a partir de un conjunto de datos.

Benavoli *et al* definen 3 pasos en el análisis bayesiano:

- 1.- Establecer un modelo matemático descriptivo de los datos. En el modelo paramétrico, este modelo matemático es la función de verosimilitud (*likelihood*), que proporciona la probabilidad del valor observado para cada uno de los parámetros  $p(Datos | \theta)$ .
- 2.- Establecer la credibilidad para cada uno de los parámetros antes de observar los datos, la distribución a priori  $p(\theta)$ .
- 3.- El tercer paso es usar la regla de Bayes para combinar la función de verosimilitud y la distribución a priori, para obtener la distribución a posteriori de los parámetros dados los datos  $p(\theta | Datos)$ .

Los tres métodos propuestos por Benavoli *et al* son:

- Prueba Bayesiana  $t$  de correlación
  - Prueba Bayesiana de rangos alienados
  - Prueba Bayesiana  $t$  de correlación jerárquica
- 

La prueba Bayesiana  $t$  de correlación es utilizada para medir el desempeño de dos algoritmos en un mismo dataset, mientras que las prueba Bayesiana de rangos alienados y la prueba Bayesiana  $t$  de correlación jerárquica, son utilizadas para medir el desempeño de dos algoritmos en multiples dataset.

Para comparar dos algoritmos en múltiples *datasets*, Benavoli *et al* recomiendan el uso de la prueba Bayesiana  $t$  de correlación jerárquica, ya que toma como entradas las  $m$  corridas de los  $k$ -folds de la validación cruzada de cada *dataset*, por lo que hace inferencias acerca de la diferencia media de precisión entre dos clasificadores para cada *dataset*, explotando toda la información disponible: la media de la muestra, la desviación estándar de la muestra, y la correlación debida a la superposición de los datos.

#### 4.1. Comparar el desempeño de las implementaciones en Python y R

Para compara el desempeño de la implementación de TSBF en lenguaje Python y la implementación presentada por Baydogan *et al* en lenguaje R, se seleccionaron la prueba Bayesiana  $t$  de correlación, para cada *dataset*, y la prueba Bayesiana  $t$  de correlación jerárquica para multiples *datasets*. Para las pruebas se utilizó la librería para lenguaje Python proporcionada por Benavoli *et al*.

En la tabla 1, se presentan los *datasets* utilizados en las pruebas. Cada uno de estos viene dividido *train* y *test*, sin embargo, para realizar las pruebas de validación cruzada (con 10 *folds*) se unieron *train* y *test* y se generaron los nuevos grupos de *train* y *test* con ayuda de la librería *StratifiedKFold* de *numpy*. Se hicieron 10 corridas para cada grupo, para los 4 valores de  $z$  : 0.1, 0.25, 0.5, y 0.75.

Cuadro 1: *Datasets* utilizados en las pruebas

<i>Dataset</i>	Train	Test	Total	Observaciones	clases	tipo
Coffe	28	28	56	286	2	SPECTRO
CBF	30	900	930	128	3	SIMULATED
ECGFiveDays	23	861	884	136	2	ECG
DiatomSizeReduction	16	306	322	345	4	IMAGE
Adiac	390	391	781	176	37	IMAGE
ECG200	100	100	200	96	2	ECG
Beef	30	30	60	470	5	SPECTRO
FaceFour	24	88	112	350	4	IMAGE
FaceAll	560	1690	2250	131	14	IMAGE
CricketY	390	390	780	300	12	MOTION
CricketX	390	390	780	300	12	MOTION
GunPoint	50	150	200	150	2	MOTION
FacesUCR	200	2050	2250	131	14	IMAGE
FiftyWords	450	455	905	270	50	IMAGE
CricketZ	390	390	780	300	12	MOTION
Lightning7	70	73	143	319	7	SENSOR
Fish	175	175	350	463	7	IMAGE
ChlorineConcentration	467	3840	4307	166	3	SIMULATED
MedicalImages	381	760	1141	99	10	IMAGE
MoteStrain	20	1252	1272	84	2	SENSOR
SonyAIBORobotSurface1	20	601	621	70	2	SPECTRO
SonyAIBORobotSurface2	27	953	980	65	2	SENSOR
SwedishLeaf	500	625	1125	128	15	IMAGE
OSULeaf	200	242	442	427	6	IMAGE
OliveOil	30	30	60	570	4	SPECTRO
SyntheticControl	300	300	600	60	6	SIMULATED
Trace	100	100	200	275	4	SENSOR
TwoLeadECG	23	1139	1162	82	2	ECG
Symbols	25	995	1020	398	6	IMAGE
Lightning2	60	61	121	637	2	SENSOR
WordSynonyms	267	638	905	270	25	IMAGE

En las tablas 2, 3, 4, y 5 del Anexo 1, se presentan los resultados de la prueba Bayesiana  $t$  de correlación para cada *dataset*, y para cada uno de los valores de  $z$ . Si ponemos un límite inferior 90 % para considerar que un algoritmos se desempeña mejor que el otro, o si ambos se comportan del mismo modo, podemos deducir lo siguiente:

Para  $z = 0.1$ : Los algoritmos se comportan de la misma manera en CBF, DiatomSizeReduction, FaceAll, FacesUCR, y Symbols. En los demás *datasets* las probabilidades no nos permiten tomar una decisión.

Para  $z = 0.25$ : Los algoritmos se comportan de la misma manera en CBF, DiatomSizeduction, FaceAll, FacesUCR, MoteStrain, Symbols, SyntheticControl y Trace. Baydogan se comporta mejor que Python en ChlorineConcentration. En los demás *datasets* las probabilidades no nos permiten tomar una decisión.

Para  $z = 0.5$ : Los algoritmos se comportan de la misma manera en CBF, Coffe, DiatomSizeReduction, ECGFiveDays, FaceAll, FacesUCR, MoteStrain, Symbols, SyntheticControl, y TwoLeadECG. Baydogan se comporta mejor que Python en ChlorineConcentration. En los demás *datasets* las probabilidades no nos permiten tomar una decisión.

Para  $z = 0.75$ : Los algoritmos se comportan de la misma manera en CBF, DiatomSizeReduction, FacesUCR, Symbols, Trace y TwoLeadECG. Baydogan se comporta mejor que Python en ChlorineConcentration. En los demás *datasets* las probabilidades no nos permiten tomar una decisión.

Por otra parte, la prueba Bayesiana  $t$  de correlación jerárquica, aplicada a todo el conjunto de *datasets*, arrojó una probabilidad del 100 % de que ambos algoritmos se desempeñan de la misma manera para cada valor de  $z$ . En la Figura 3, se presentan las gráficas generadas por la librería de Benavoli *et al.* En estas gráficas se aprecian las áreas de probabilidad y la posición que toma un conjunto de muestras generados aleatoriamente por la distribución a posteriori.

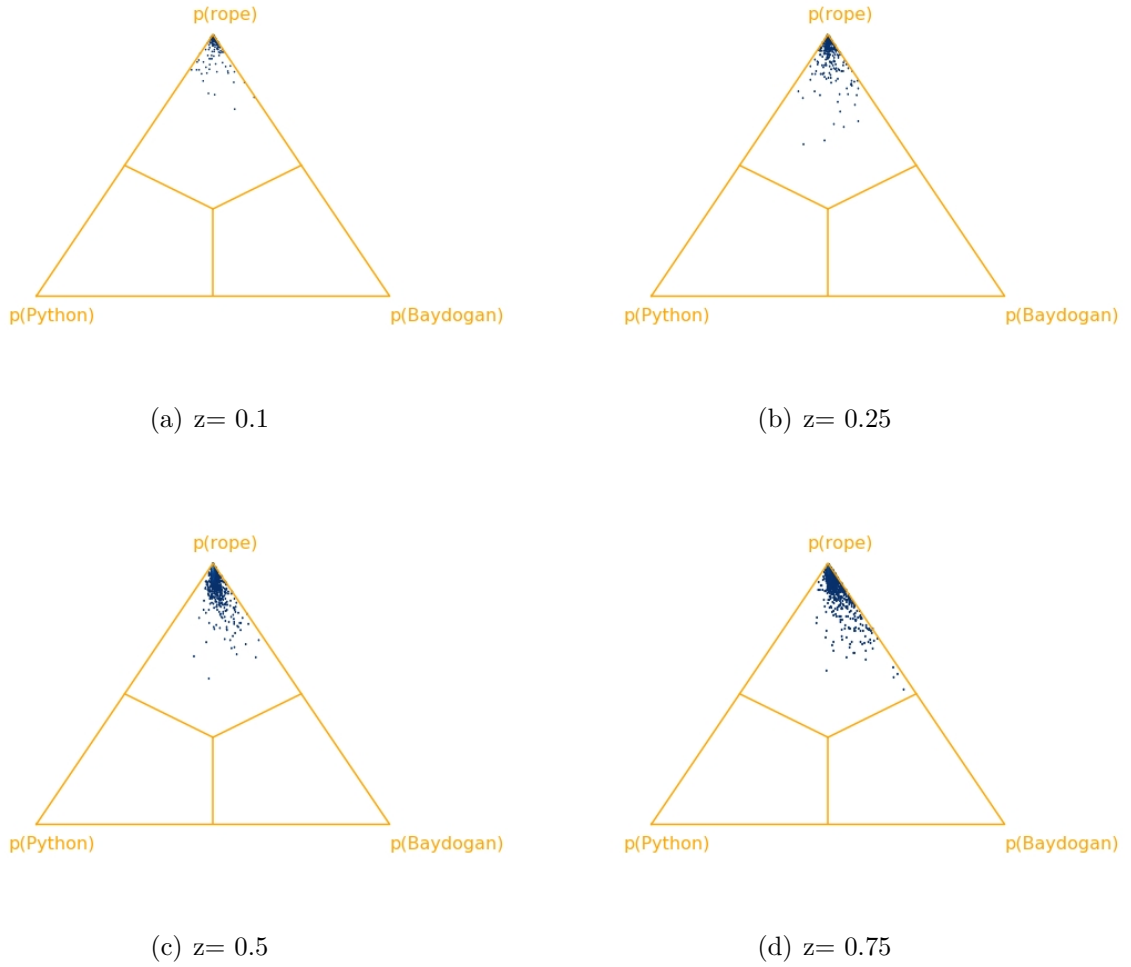


Figura 3: Resultados de la pruebas Bayesianas  $t$  de correlación jerárquica

#### 4.2. Comparar el desempeño de implementación en Python utilizando *ExtraTrees*

Para compara el desempeño de la implementación de TSBF en Python utilizando *ExtraTrees* como clasificador y la implementación presentada por Baydogan *et al* en lenguaje R, se seleccionaron la prueba Bayesiana  $t$  de correlación, para cada *dataset*, y la prueba Bayesiana  $t$  de correlación jerárquica para multiples *datasets*. Para las pruebas se utilizó la librería para lenguaje Python proporcionada por Benavoli *et al*.

Para comparar las implementaciones en Python utilizando *ExtraTrees* y R, se utilizaron los mismos *datasets* utilizados en al comparar el desempeño de las implementaciones en Python y R.

En las tablas 6, 7, 8, y 9 del Anexo 1, se presentan los resultados de la prueba Bayesiana



$t$  de correlación para cada *dataset*, y para cada uno de los valores de  $z$ . Si ponemos un límite inferior de 90 % para considerar que un algoritmo se desempeña mejor que el otro, o si ambos se comportan del mismo modo, podemos deducir lo siguiente:

Para  $z = 0.1$ : Los algoritmos tienen el mismo desempeño en CBF, DiatomSizeReduction, ECGFiveDays, y Symbols. Python con ExtraTrees se desempeña mejor en Fish. Baydogan se desempeña mejor en Adiac, y OliveOil. En los demás *datasets* las probabilidades no nos permiten tomar una decisión.

Para  $z = 0.25$ : Los algoritmos tienen el mismo desempeño en CBF, DiatomSizeduction, FacesUCR, Symbols, y Trace. Python con ExtraTrees se desempeña mejor en CricketY, Fish, y Lightning7. Baydogan se desempeña mejor en ChlorineConcentration. En los demás *datasets* las probabilidades no nos permiten tomar una decisión.

Para  $z = 0.5$ : Los algoritmos tienen el mismo desempeño en CBF, DiatomSizeReduction, Symbols, SyntheticControl, y TwoLeadECG. Python con ExtraTrees se desempeña mejor en Lightning7. Baydogan se comporta mejor en ChlorineConcentration, MoteStrain y SonyAIBORobotSurface2. En los demás *datasets* las probabilidades no nos permiten tomar una decisión.

Para  $z = 0.75$ : Los algoritmos tienen el mismo desempeño en CBF, DiatomSizeReduction, Symbols, y TwoLeadECG. Baydogan se comporta mejor en ChlorineConcentration, y SonyAIBORobotSurface2. En los demás *datasets* las probabilidades no nos permiten tomar una decisión.

Por otra parte, la prueba Bayesiana  $t$  de correlación jerárquica, aplicada a todo el conjunto de *datasets*, arrojó una probabilidad del 100 % de que ambos algoritmos se desempeñen de la misma manera para  $z = 0.1$ , 95.15 % para  $z = 0.25$ , 90 % para  $z = 0.5$ , y 99.7 % para  $z = 0.75$ . En la Figura 4, se presentan las gráficas generadas por la librería de Benavoli *et al.* En estas gráficas se aprecian las áreas de probabilidad y la posición que toma un conjunto de muestras generados aleatoriamente por la distribución a posteriori.

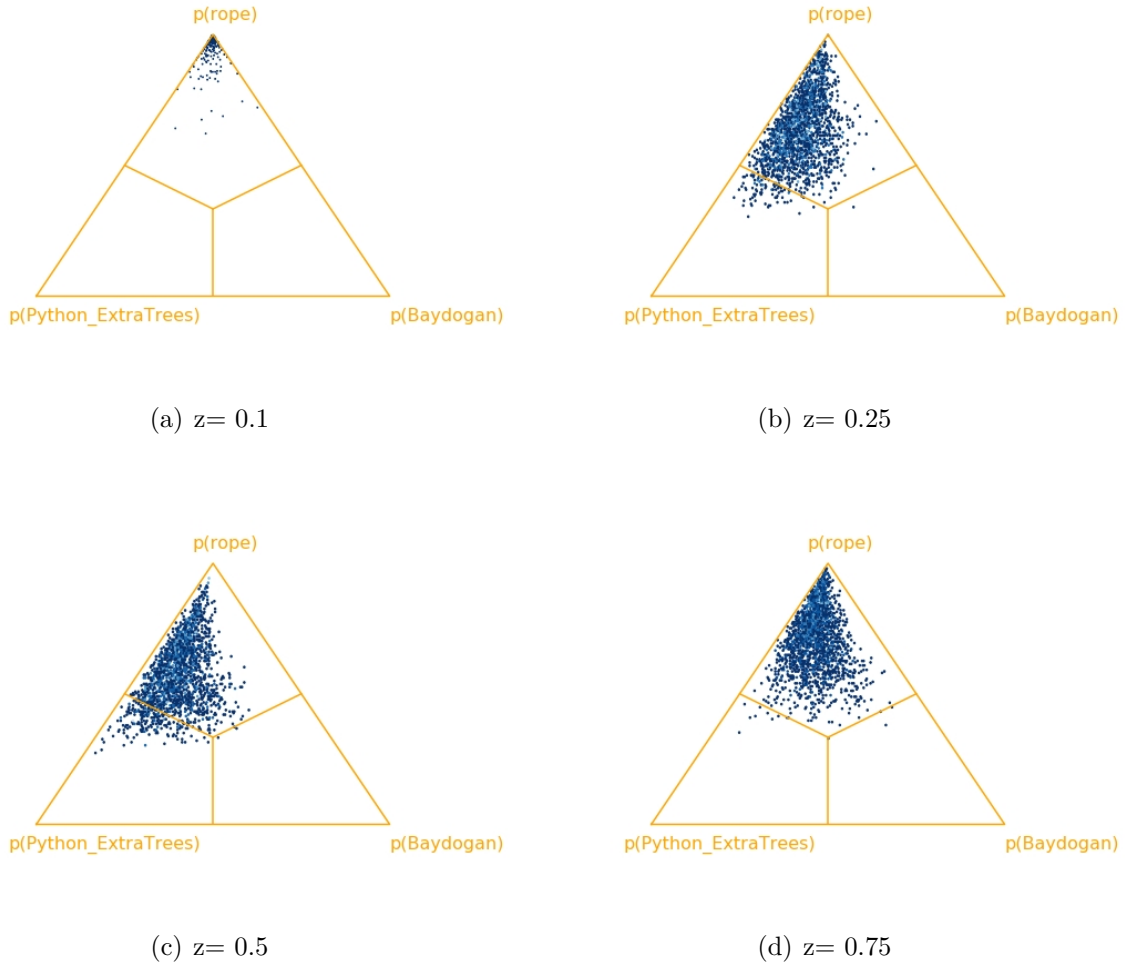


Figura 4: Resultados de la pruebas Bayesianas  $t$  de correlación jerárquica

## 5. Conclusiones

La mayoría de las pruebas Bayesianas  $t$  de correlación aplicadas al comparar las implementaciones en Python y en R, para los distintos *dataset*, no permitieron emitir un juicio. Sin embargo, en aquellos en los que si se puede emitir un juicio, la mayoría apuntó a que el comportamiento de ambos algoritmos es igual. Este mismo comportamiento se [presentó](#) en la comparación entre la implementación en Python utilizando *ExtraTrees* con R.

Por otra parte, el resultado presentado por la prueba Bayesiana  $t$  de correlación jerárquica, permitió determinar que el comportamiento de ambos algoritmos en base a los resultado de las pruebas de todos los dataset es el mismo, con una probabilidad del 100 % para todos los valores de  $z$  en la implementación en Python. De igual modo esta prueba nos permitió determinar que el comportamiento de la implementación en Python utilizando *ExtraTrees* es igual a la implementación en R; con una probabilidad del 100 % para  $z = 0.1$ , 95.15 % para

$z = 0.25$ , 90 % para  $z = 0.5$ , y 99.7 % para  $z = 0.75$ .

## 6. Trabajo Futuro

Partiendo de la implementación en Python de TSBF se pueden probar diferentes configuraciones utilizando los clasificadores disponibles en la *suite* de *scikit-learn*. Además, se puede mejorar el rendimiento de la implementación utilizando múltiples hilos. Actualmente, tanto en la implementación de Baydogan *et al* en R como en la implementación en Python; solo los clasificadores funcionan en múltiples hilos. Por último, aún no se han probado diferentes tamaños de conjuntos ( $b$ ). En las pruebas presentadas por Baydogan *etal*, y en las presentadas en este reporte se utilizó un valor de  $b = 10$ .

## Referencias

- [1] M. Falk, F. Marohn, R. Michel, D. Hofmann, M. Macke, B. Tewes, P. Dinges, and S. Englert, “A first course on time series analysis : Examples with sas,” <http://statistik.mathematik.uni-wuerzburg.de/timeseries/>, 04 2005.
- [2] M. G. Baydogan, G. Runger, and E. Tuv, “A bag-of-features framework to classify time series,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2796–2802, Nov 2013.
- [3] A. Benavoli, G. Corani, J. Demsar, and M. Zaffalon, “Time for a change: A tutorial for comparing multiple classifiers through bayesian analysis,” *Journal of Machine Learning Research*, vol. 18, 06 2016.

## Anexo 1

Cuadro 2: Resultados de la prueba Bayesiana  $t$  de correlación para  $z = 0.1$

<i>dataset</i>	Python > R	Python = R	R > Python
Adiac	0.396098	0.39052047	0.21338152
Beef	0.3620509	0.15359659	0.48435252
CBF	5.52408099e-14	1.00000000e+00	3.10862447e-15
ChlorineConcentration	0.03248057	0.6815569	0.28596253
Coffee	0.24404075	0.27573313	0.48022612
CricketX	0.09795619	0.53963231	0.36241149
CricketY	0.13171922	0.50672823	0.36155255
CricketZ	0.07810557	0.57393297	0.34796146
DiatomSizeReduction	9.36704327e-03	9.89890592e-01	7.42364764e-04
ECG200	0.33732759	0.26597639	0.39669602
ECGFiveDays	0.06736336	0.88132356	0.05131307
FaceAll	0.02228171	0.90880353	0.06891476
FaceFour	0.02909849	0.75720018	0.21370133
FacesUCR	0.01220821	0.91397865	0.07381314
FiftyWords	0.26193502	0.61716665	0.12089833
Fish	0.17769794	0.60423594	0.21806613
GunPoint	0.23499697	0.60057358	0.16442945
Lightning2	0.28582051	0.31806703	0.39611246
Lightning7	0.17881893	0.29182803	0.52935304
MedicalImages	0.1302978	0.46453746	0.40516474
MoteStrain	0.44113126	0.53110339	0.02776535
OSULeaf	0.26103839	0.54052386	0.19843775
OliveOil	0.27205948	0.14638731	0.58155321
SonyAIBORobotSurface1	0.21471721	0.54824327	0.23703952
SonyAIBORobotSurface2	0.14351545	0.51131274	0.34517181
SwedishLeaf	0.22669238	0.60905739	0.16425023
Symbols	2.66853706e-05	9.99863973e-01	1.09341413e-04
SyntheticControl	0.30749757	0.47494863	0.21755381
Trace	0.0648209	0.87035819	0.0648209
TwoLeadECG	0.50649815	0.46809731	0.02540454
WordSynonyms	0.21791905	0.65204729	0.13003366

Cuadro 3: Resultados de la prueba Bayesiana  $t$  de correlación para  $z = 0.25$

<i>dataset</i>	Python > R	Python = R	R > Python
Adiac	0.26146006	0.53451812	0.20402183
Beef	0.19333826	0.10355984	0.7031019
CBF	1.95100051e-19	1.00000000e+00	0.00000000e+00
ChlorineConcentration	1.59195184e-07	1.23138326e-03	9.98768458e-01
Coffee	0.20519111	0.66986445	0.12494444
CricketX	0.14886226	0.55427803	0.29685972
CricketY	0.17525434	0.55968058	0.26506508
CricketZ	0.12587706	0.59564856	0.27847438
DiatomSizeReduction	2.43857735e-03	9.97386128e-01	1.75294798e-04
ECG200	0.29215488	0.29781774	0.41002738
ECGFiveDays	0.46857275	0.53028539	0.00114187
FaceAll	6.24506549e-02	9.36720464e-01	8.28881225e-04
FaceFour	0.04821857	0.54229265	0.40948878
FacesUCR	0.00250245	0.9886196	0.00887795
FiftyWords	0.24381672	0.65580516	0.10037812
Fish	0.06409594	0.52520392	0.41070014
GunPoint	0.12663108	0.69867106	0.17469785
Lightning2	0.37327944	0.28629114	0.34042941
Lightning7	0.51109	0.29172452	0.19718548
MedicalImages	0.10170043	0.55889157	0.339408
MoteStrain	0.0739804	0.90024542	0.02577419
OSULeaf	0.20717132	0.64465551	0.14817317
OliveOil	0.52761181	0.1784513	0.29393689
SonyAIBORobotSurface1	0.15710329	0.70577947	0.13711724
SonyAIBORobotSurface2	0.05760484	0.56387328	0.37852188
SwedishLeaf	0.14890955	0.66096486	0.19012559
Symbols	3.78550856e-07	9.99998811e-01	8.10591268e-07
SyntheticControl	0.04701916	0.92066887	0.03231197
Trace	1.77175837e-07	9.99999809e-01	1.38986265e-08
TwoLeadECG	0.00531992	0.77386084	0.22081924
WordSynonyms	0.37316272	0.52586696	0.10097032

Cuadro 4: Resultados de la prueba Bayesiana  $t$  de correlación para  $z = 0.5$

<i>dataset</i>	Python > R	Python = R	R > Python
Adiac	0.09215519	0.53126751	0.3765773
Beef	0.08496373	0.05983979	0.85519648
CBF	9.02577392e-48	1.00000000e+00	0.00000000e+00
ChlorineConcentration	1.75334747e-12	9.57492653e-07	9.99999043e-01
Coffee	0	1	0
CricketX	0.02002252	0.38880093	0.59117656
CricketY	0.00920922	0.4030986	0.58769218
CricketZ	0.06059288	0.62910723	0.31029988
DiatomSizeReduction	0.00237944	0.99363317	0.00398739
ECG200	0.23084836	0.33718756	0.43196408
ECGFiveDays	0.00141687	0.90974291	0.08884022
FaceAll	0.00141359	0.96934833	0.02923807
FaceFour	0.02926575	0.49461347	0.47612077
FacesUCR	0.00265374	0.99387365	0.00347261
FiftyWords	0.15080876	0.69510542	0.15408582
Fish	0.17687858	0.71458547	0.10853595
GunPoint	0.10601104	0.6918037	0.20218526
Lightning2	0.29309608	0.3310332	0.37587072
Lightning7	0.49976217	0.35038493	0.1498529
MedicalImages	0.30264651	0.59338136	0.10397213
MoteStrain	0.04647764	0.93004308	0.02347928
OSULeaf	0.5582785	0.39633563	0.04538587
OliveOil	0.19936386	0.15063178	0.65000436
SonyAIBORobotSurface1	0.11195657	0.77538014	0.11266329
SonyAIBORobotSurface2	0.00935158	0.26105982	0.7295886
SwedishLeaf	0.10685169	0.77810573	0.11504258
Symbols	1.57428303e-04	9.99804257e-01	3.83149097e-05
SyntheticControl	0.02621573	0.95757798	0.01620629
Trace	0.10280507	0.89245999	0.00473494
TwoLeadECG	3.33651599e-04	9.84344734e-01	1.53216145e-02
WordSynonyms	0.2946319	0.63454097	0.07082714

Cuadro 5: Resultados de la prueba Bayesiana  $t$  de correlación para  $z = 0.75$

<i>dataset</i>	Python > R	Python = R	R > Python
Adiac	0.2331464	0.50893522	0.25791838
Beef	0.12907094	0.08899277	0.78193629
CBF	8.34401735e-36	1.00000000e+00	0.00000000e+00
ChlorineConcentration	9.89601704e-11	5.95645256e-05	9.99940435e-01
Coffee	0.11400755	0.7719849	0.11400755
CricketX	0.02980696	0.63346995	0.33672309
CricketY	0.06999215	0.55401224	0.3759956
CricketZ	0.05485093	0.68311518	0.26203389
DiatomSizeReduction	9.62627466e-04	9.97203874e-01	1.83349836e-03
ECG200	0.28867736	0.29345446	0.41786819
ECGFiveDays	3.38580608e-04	7.63990686e-01	2.35670733e-01
FaceAll	9.64043974e-05	8.47005727e-01	1.52897869e-01
FaceFour	0.13101914	0.7090594	0.15992146
FacesUCR	2.75392924e-05	9.25243106e-01	7.47293543e-02
FiftyWords	0.19525927	0.70472656	0.10001417
Fish	0.22210458	0.70584535	0.07205007
GunPoint	0.30674946	0.65122986	0.04202068
Lightning2	0.36098963	0.32133916	0.31767121
Lightning7	0.31834174	0.37359889	0.30805938
MedicalImages	0.14515025	0.59590085	0.25894891
MoteStrain	0.00704297	0.89148668	0.10147036
OSULeaf	0.41346112	0.45354226	0.13299662
OliveOil	0.12366161	0.14697396	0.72936444
SonyAIBORobotSurface1	0.0389231	0.85262057	0.10845633
SonyAIBORobotSurface2	0.00637977	0.33480527	0.65881496
SwedishLeaf	0.046074	0.77603425	0.17789175
Symbols	1.51917346e-03	9.98480665e-01	1.61287964e-07
SyntheticControl	0.20261335	0.77888683	0.01849982
Trace	1.38986265e-08	9.99999809e-01	1.77175837e-07
TwoLeadECG	5.75646817e-07	9.99900700e-01	9.87247405e-05
WordSynonyms	0.28659738	0.6621893	0.05121332



Cuadro 6: Resultados de la prueba Bayesiana  $t$  de correlación entre *ExtraTrees* y Baydogan para  $z = 0.1$

<i>dataset</i>	pExtraTrees > R	pExtraTrees = R	R > pExtraTrees
Adiac	6.46710743e-04	1.68719811e-02	9.82481308e-01
Beef	0.24822436	0.15633266	0.59544298
CBF	3.36606509e-12	1.00000000e+00	2.59237076e-13
ChlorineConcentration	0.02525937	0.83272716	0.14201347
Coffee	0.22542949	0.47960636	0.29496415
CricketX	0.73174182	0.2506055	0.01765268
CricketY	0.68118887	0.29355097	0.02526016
CricketZ	0.49693069	0.47520954	0.02785977
DiatomSizeReduction	0.03876648	0.95795292	0.00328059
ECG200	0.33703973	0.28823787	0.37472239
ECGFiveDays	0.02430877	0.94849373	0.02719749
FaceAll	0.00849142	0.87086091	0.12064767
FaceFour	0.15459606	0.78766784	0.0577361
FacesUCR	0.00694194	0.83245027	0.16060779
FiftyWords	0.0878548	0.67524683	0.23689837
Fish	0.94468187	0.05352602	0.00179211
GunPoint	0.48232144	0.47458951	0.04308905
Lightning2	0.27014319	0.2839228	0.44593401
Lightning7	0.81211906	0.14899943	0.0388815
MedicalImages	0.00451651	0.13108379	0.86439969
MoteStrain	0.51745177	0.46753865	0.01500958
OSULeaf	0.45855676	0.45687996	0.08456328
OliveOil	0.0504915	0.03785491	0.91165359
SonyAIBORobotSurface1	0.2288345	0.51629672	0.25486878
SonyAIBORobotSurface2	0.08079513	0.52706829	0.39213658
SwedishLeaf	0.1313242	0.57988555	0.28879026
Symbols	1.25107054e-04	9.99715984e-01	1.58908905e-04
SyntheticControl	0.49562482	0.40749039	0.09688479
Trace	0.04191652	0.80945961	0.14862388
TwoLeadECG	0.29074814	0.58498518	0.12426668
WordSynonyms	0.13248274	0.64348087	0.2240364

Cuadro 7: Resultados de la prueba Bayesiana  $t$  de correlación entre *ExtraTrees* y Baydogan para  $z = 0.25$

<i>dataset</i>	pExtraTrees > R	pExtraTrees = R	R > pExtraTrees
Adiac	0.01857578	0.21948665	0.76193757
Beef	0.28074775	0.11400528	0.60524697
CBF	5.8781518e-28	1.0000000e+00	0.0000000e+00
ChlorineConcentration	2.43889782e-09	4.63340199e-05	9.99953664e-01
Coffee	0.20519111	0.66986445	0.12494444
CricketX	0.81967442	0.1706181	0.00970748
CricketY	0.94047724	0.05851396	0.0010088
CricketZ	0.82877917	0.1655187	0.00570213
DiatomSizeReduction	4.03874768e-02	9.59048426e-01	5.64097080e-04
ECG200	0.33284719	0.346635	0.32051781
ECGFiveDays	7.40218205e-01	2.59747208e-01	3.45864871e-05
FaceAll	7.23313440e-01	2.76685574e-01	9.85770253e-07
FaceFour	0.26331568	0.63800617	0.09867815
FacesUCR	2.00767511e-02	9.79196259e-01	7.26989936e-04
FiftyWords	0.2829053	0.62604753	0.09104717
Fish	9.88873444e-01	1.10884739e-02	3.80819906e-05
GunPoint	0.38722328	0.58927793	0.02349879
Lightning2	0.47007899	0.26189571	0.26802529
Lightning7	0.95226556	0.0410862	0.00664823
MedicalImages	0.10718398	0.57575351	0.31706251
MoteStrain	2.31077348e-04	2.55539251e-01	7.44229672e-01
OSULeaf	0.56394392	0.40774946	0.02830661
OliveOil	0.82078114	0.08979733	0.08942153
SonyAIBORobotSurface1	0.30110758	0.62050442	0.078388
SonyAIBORobotSurface2	0.00371633	0.16434029	0.83194338
SwedishLeaf	0.26278354	0.65914663	0.07806983
Symbols	2.15463654e-03	9.97769387e-01	7.59768865e-05
SyntheticControl	0.15809137	0.81436689	0.02754174
Trace	1.92512955e-04	9.99797950e-01	9.53748993e-06
TwoLeadECG	0.00310353	0.78702756	0.20986891
WordSynonyms	0.5721368	0.39276501	0.03509819

Cuadro 8: Resultados de la prueba Bayesiana  $t$  de correlación entre *ExtraTrees* y Baydogan para  $z = 0.5$

<i>dataset</i>	pExtraTrees > R	pExtraTrees = R	R > pExtraTrees
Adiac	0.14610689	0.58442775	0.26946536
Beef	0.42343168	0.12779249	0.44877583
CBF	5.8703429e-47	1.0000000e+00	0.0000000e+00
ChlorineConcentration	1.37629968e-15	2.11859785e-09	9.99999998e-01
Coffee	0.02349879	0.89952879	0.07697243
CricketX	0.8077774	0.18690178	0.00532082
CricketY	0.86878557	0.13005597	0.00115846
CricketZ	0.68120954	0.30523578	0.01355468
DiatomSizeReduction	2.55212264e-03	9.97435836e-01	1.20411526e-05
ECG200	0.32917462	0.32103808	0.34978729
ECGFiveDays	6.57653864e-01	3.42346081e-01	5.42742595e-08
FaceAll	8.52041247e-01	1.47958662e-01	9.03398781e-08
FaceFour	0.33952513	0.52784956	0.13262531
FacesUCR	5.87988173e-01	4.12011737e-01	8.99786197e-08
FiftyWords	0.07980746	0.66526498	0.25492757
Fish	0.87866022	0.11692128	0.00441851
GunPoint	0.30579808	0.57548779	0.11871414
Lightning2	0.53534328	0.30219277	0.16246395
Lightning7	0.97159944	0.02485892	0.00354163
MedicalImages	0.73711962	0.25064357	0.01223681
MoteStrain	2.65532183e-05	7.19956612e-02	9.27977786e-01
OSULeaf	0.88032835	0.11672678	0.00294487
OliveOil	0.89290594	0.0639084	0.04318566
SonyAIBORobotSurface1	0.35260085	0.62716195	0.0202372
SonyAIBORobotSurface2	9.33530023e-06	7.04829216e-03	9.92942373e-01
SwedishLeaf	0.55439983	0.43922988	0.00637028
Symbols	8.21545873e-05	9.99913490e-01	4.35550387e-06
SyntheticControl	0.01803021	0.9751514	0.00681839
Trace	0.24158744	0.75547318	0.00293938
TwoLeadECG	1.08235522e-04	9.99316019e-01	5.75745160e-04
WordSynonyms	0.29500835	0.63251962	0.07247202

Cuadro 9: Resultados de la prueba Bayesiana  $t$  de correlación entre *ExtraTrees* y Baydogan para  $z = 0.75$

<i>dataset</i>	pExtraTrees > R	pExtraTrees = R	R > pExtraTrees
Adiac	0.26143326	0.56935432	0.16921242
Beef	0.22661319	0.10904163	0.66434518
CBF	0	1	0
ChlorineConcentration	2.09168490e-14	2.06935390e-08	9.99999979e-01
Coffee	0.12416538	0.52607248	0.34976214
CricketX	0.30507216	0.61126145	0.08366639
CricketY	0.4728749	0.50384523	0.02327987
CricketZ	0.09780625	0.74066755	0.1615262
DiatomSizeReduction	6.58571023e-03	9.93326454e-01	8.78352963e-05
ECG200	0.3941328	0.30763584	0.29823136
ECGFiveDays	4.08371447e-01	5.91628138e-01	4.15367483e-07
FaceAll	6.50879702e-01	3.49120279e-01	1.88678633e-08
FaceFour	0.33428664	0.58045212	0.08526125
FacesUCR	3.19944685e-01	6.80055109e-01	2.05548951e-07
FiftyWords	0.00424299	0.36521922	0.63053779
Fish	0.83476153	0.16103783	0.00420063
GunPoint	0.2428085	0.72319644	0.03399506
Lightning2	0.24407948	0.28436902	0.4715515
Lightning7	0.59804501	0.2490308	0.1529242
MedicalImages	0.29445385	0.58792641	0.11761974
MoteStrain	2.18834608e-05	1.30136221e-01	8.69841896e-01
OSULeaf	0.55416663	0.36611917	0.0797142
OliveOil	0.80262646	0.11273741	0.08463612
SonyAIBORobotSurface1	0.34182201	0.64860939	0.0095686
SonyAIBORobotSurface2	1.95340593e-04	5.67242010e-02	9.43080458e-01
SwedishLeaf	0.2469643	0.70751751	0.04551819
Symbols	1.04770524e-05	9.99980190e-01	9.33316374e-06
SyntheticControl	0.20839311	0.78559728	0.00600961
Trace	0.41381459	0.56006995	0.02611546
TwoLeadECG	2.64040106e-05	9.99916464e-01	5.71318434e-05
WordSynonyms	0.0457645	0.65775291	0.29648259