

**UNIVERSIDAD ANDRÉS BELLO**

**FACULTAD DE INGENIERÍA**

**INGENIERÍA CIVIL INFORMÁTICA**

**Trabajo 3 – Aprendizaje No-Supervisado (K-Means)**

**ANDRÉS EDUARDO VALENZUELA GONZÁLEZ**

**SANTIAGO - CHILE**

**OCTUBRE, 2017**

Contenido

[**1.** **Introducción** 3](#_Toc493209607)

[**2.** **Vectorización** 3](#_Toc493209608)

[**3.** **Cross - Validation** 6](#_Toc493209609)

[**4.** **Entrenamiento** 8](#_Toc493209610)

[**5.** **Desarrollo** 9](#_Toc493209611)

[5.1. Cuando se tengan los diez archivos de resultados, se debe calcular el desempeño del clasificador SVM para predecir la ubicación geográfica. 9](#_Toc493209612)

[5.2. Calcule el MRR y la matriz de confusión. Entregue también los promedios de los diez splits. 10](#_Toc493209613)

[5.3. Comente acerca de los errores ¿Hay algún patrón?, ¿Cómo podría mejorar?, ¿Qué puede decir de los errores?, ¿Con respecto a las ubicaciones geográficas (strings) nota algún patrón en los errores? 12](#_Toc493209614)

[5.4. Calcule la curva ROC, Curva Lift, F(1)-score y el estimador AUC asumiendo cada una de las variables como positivas. 13](#_Toc493209615)

[5.4.1. F1 – Score 13](#_Toc493209616)

[5.4.2. Curva ROC 14](#_Toc493209617)

[5.4.3. Curva Lift 16](#_Toc493209618)

[5.4.4. Estimador AUC 19](#_Toc493209619)

[**6.** **Conclusión** 19](#_Toc493209620)

# **Introducción**

Para comprender el contexto del siguiente informe, se utilizaron los dos mil (2000) perfiles vectorizados en la tarea anterior para hacer uso del algoritmo *K-Means* provisto por el sitio [*http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/yakmo/*](http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/yakmo/) .

Antes de comenzar la ejecución, es necesario entender la estructura de los vectores recibidos por el programa (más detalles en <https://gitlab.com/Choapinus/SistemasInteligentes/blob/master/Tarea%202/enunciado_informe/Informe%20II.pdf> , punto 2):

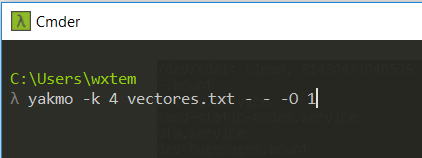
Para realizar el aprendizaje no-supervisado se debió modelar el problema mediante vectores.   
Si asumimos que cada una de las palabras es un atributo, entonces podríamos crear una representación vectorial que indique la presencia o ausencia de esta palabra en cada una de las descripciones del usuario etiquetadas en la primera tarea. Para la etiqueta se utilizará un valor desde 1 hasta 4:

1 = *Undetermined* 2 = *Non-USA* 3 = *World* 4 = *USA only*.

Cada etiqueta debe estar asociada con un único valor numérico. *Extracto del enunciado Tarea 2, Espacio Vectorial, 28 de agosto 2017 [consulta: 11 octubre 2017, 12:32 hrs]. Disponible en:* <https://gitlab.com/Choapinus/SistemasInteligentes/blob/master/Tarea%202/enunciado_informe/Tarea%202.pdf>

# **Ejecución**

Para explicar el proceso de aprendizaje, se considera usar el siguiente comando con el programa *yakmo*:



Donde el archivo *vectores.txt* contiene nuestros vectores creador con anterioridad.

## Errores

Este metodo condujo a errores ya que los datos dentro del archivo *vectores.txt* estaban de cierta forma ordenados. Tal orden afecto en gran medida la salida del programa *yakmo*, dando orignen a los primeros tres centroides con un solo dato y el ultimo con el resto de ellos como muestra la imagen.

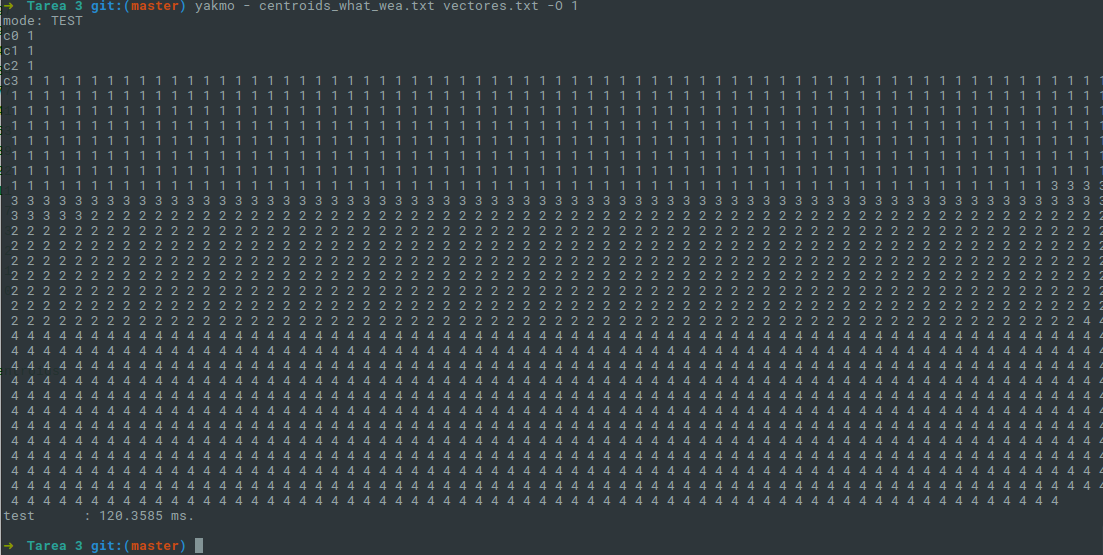


Ilustración 1 - Centroides

En la ilustración 1 se puede apreciar que se evalua el modelo generado por el yakmo para generar los centroides con base al archivo *vectores.txt.* De esto se puede inferir que la gran mayoria de los datos cayo dentro de la clasificacion del centroide 4 (c3) y los tres datos restantes fueron **outliers**, o bien la gran mayoria de datos fueron **outliers** que cayeron dentro del cluster 4 (c3) y los bien clasificados fueron los primeros tres datos (tiene mas sentido la primera conclusión).

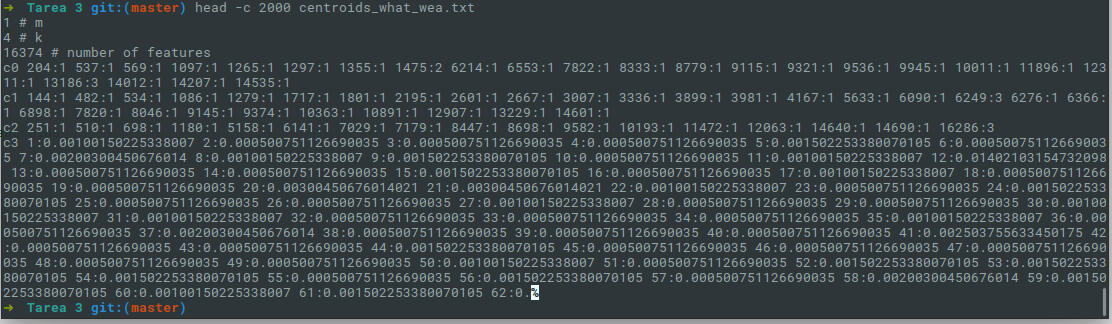


Ilustración 2 - Primer Modelo Generado

## Solucion propuesta

Como solución se propuso generar un *random input* a partir de los vectores originales. Para realizar esto, se creo un programa en *Python* el cual toma los vectores originales y randomiza su orden, ejecuta el programa yakmo

# **Salida**

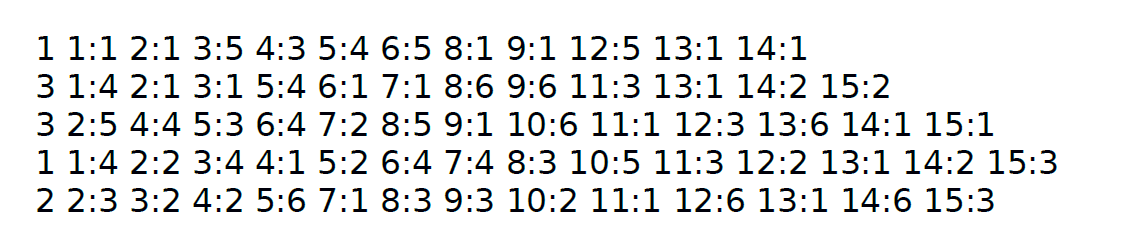
Se representaron los perfiles como vectores, indicando la frecuencia de cada palabra en la pregunta con su índice correspondiente respecto a la bolsa de palabras (Por bolsa de palabras entenderemos a las palabras ordenadas respecto al orden de llegada desde los perfiles, con las *stopwords* descontadas). Luego, se generaron los vectores con formato *SVM multiclass*, descrito a continuación:

<target> <feature>:<value> <feature>:<value> ... <feature>:<value>

En donde:

* **Target:** Valor de 1 a 4, representa la clase a la que pertenece.
* **Feature:** id de la palabra ubicada dentro de la bolsa de palabras.
* **Value:** Veces que se repite la palabra dentro del perfil vectorizado.

Aquellas palabras que no se encuentren presentes en el perfil se pueden omitir (si se quiere incluir, se iguala su atributo *value* = 0). Para el ejemplo descrito anteriormente, se tiene:

Cabe destacar que cada ID debe referirse a la misma palabra a través de todas las instancias de entrenamiento y que el formato de *SVM multiclass* exige que se escriban en orden ascendente (en términos de ID). *Extracto del enunciado Tarea 2, SVM Multiclass, 28 de agosto 2017 [consulta: 13 septiembre 2017, 20:12 hrs]. Disponible en:* <https://gitlab.com/Choapinus/SistemasInteligentes/blob/master/Tarea%202/enunciado_informe/Tarea%202.pdf>

Con base en el ejemplo anterior, el alumnado utilizo un programa de su autoría hecha en *Python* para contar las palabras, descontar las *stopwords* y generar los vectores con sus respectivas clases y sus pertenecientes palabras.

A continuación, un ejemplo de los vectores creados:

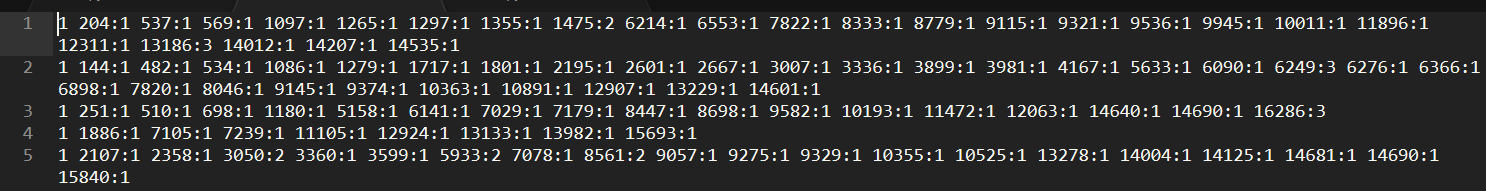


Ilustración 3.- Vectores

Se dispone acceso al repositorio en *gitlab* para disponer del ejemplo:

<https://gitlab.com/Choapinus/SistemasInteligentes/blob/master/Tarea%202/codigos/make_vectors/data/vectores.txt>

Y del código que genera los vectores con el csv del alumnado:

<https://gitlab.com/Choapinus/SistemasInteligentes/tree/master/Tarea%202/codigos/make_vectors>

Si se desea hacer uso del código, se ruega leer las instrucciones adjuntas.

# **Cross - Validation**

Una vez obtenidos los vectores, hay que dividir el set de entrenamiento en diez partes iguales (por ejemplo, si en total se tienen 2000 perfiles, cada *split* (sub-set) consta de 200 perfiles etiquetados). Luego, se debe repetir diez veces el proceso de asignar un *split* para evaluar y los restantes nueve como entrenamiento. *Extracto del enunciado Tarea 2, Cross-Validation, 28 de agosto 2017 [consulta: 13 septiembre 2017, 20:12 hrs]. Disponible en:* <https://gitlab.com/Choapinus/SistemasInteligentes/blob/master/Tarea%202/enunciado_informe/Tarea%202.pdf>

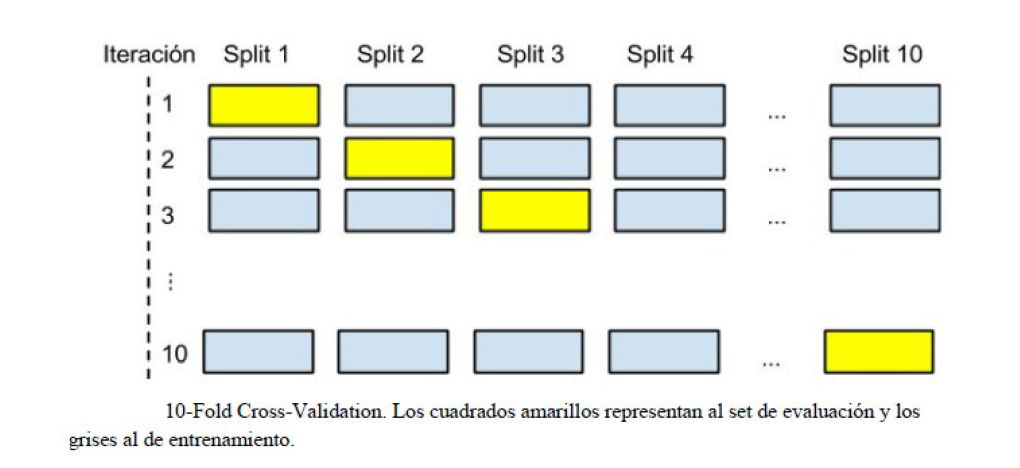


Ilustración 4.- 10-Fold Cross-Validation

Luego, se crearon los sets de entrenamientos de la siguiente manera:

E 1 = S2 + S3 + S4 + S5 + S6 + S7 + S8 + S9 + S10  
E 2 = S1 + S3 + S4 + S5 + S6 + S7 + S8 + S9 + S10  
E 3 = S1 + S2 + S4 + S5 + S6 + S7 + S8 + S9 + S10  
E 4 = S1 + S2 + S3 + S5 + S6 + S7 + S8 + S9 + S10  
E 5 = S1 + S2 + S3 + S4 + S6 + S7 + S8 + S9 + S10  
E 6 = S1 + S2 + S3 + S4 + S5 + S7 + S8 + S9 + S10  
E 7 = S1 + S2 + S3 + S4 + S5 + S6 + S8 + S9 + S10  
E 8 = S1 + S2 + S3 + S4 + S5 + S6 + S7 + S9 + S10  
E 9 = S1 + S2 + S3 + S4 + S5 + S6 + S7 + S8 + S10  
E 10 = S1 + S2 + S3 + S4 + S5 + S6 + S7 + S8 + S9

Es importante destacar que los datos de prueba **no deben ser usados para entrenar**.

Por recomendación del docente que imparte el ramo, se optó por realizar un “*random sampling Cross-Validation*” para cada split de datos con 200 muestras del archivo de vectores original. Esto debido a que permite que el entrenamiento se efectúe con una mayor diversidad de perfiles y con una mayor probabilidad de que existan todas las clases dentro del split (además de entretenido, se obtuvo un entrenamiento más preciso).

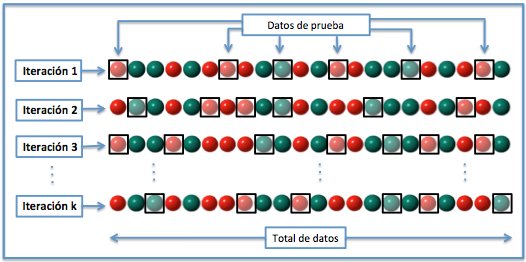


Ilustración 5.- Random Sampling Cross-Validation

Los datos de entrenamiento se construyeron de igual forma como se especifica en la formula descrita anterior al *RS Cross-Validation*.

E1 = S2 + S3 + S4 + S5 + S6 + S7 + S8 + S9 + S10  
E2 = S1 + S3 + S4 + S5 + S6 + S7 + S8 + S9 + S10  
…

Se dispone del código para su uso:

<https://gitlab.com/Choapinus/SistemasInteligentes/tree/master/Tarea%202/codigos/check_predictions>

Desde ya se advierte que por cada ejecución del código de creación de *splits*, se generaran *splits* totalmente distintos a sus antecesores.

# **Entrenamiento**

Una vez construidos los conjuntos de entrenamiento, se deben generar los modelos.   
Para esto, se debe utilizar el comando “*svm\_multiclass\_learn -c X Ei Mi*”, con lo que se genera un modelo (*Mi*) por cada uno de los *Ei*. *X* corresponde a un factor de penalización que utiliza el algoritmo. Para evaluar el modelo, debe ejecutarse sobre el split que no fue considerado en el respectivo *Ei*, es decir *Si*, mediante el comando “svm\_*multiclass*\_classify *Si* *Mi* *Ri*” donde *Ri* es el archivo en donde se almacenará el resultado. Por ejemplo:

**$ ./svm\_multiclass\_learn –c 5000 train.dat model.dat  
$ ./svm\_multiclass\_classify test.dat model.dat predictions.dat**

Luego, el resultado de ejecutar el comando de prueba queda en el archivo predictions.dat. *Extracto del enunciado Tarea 2, Cross-Validation, 28 de agosto 2017 [consulta: 13 septiembre 2017, 20:12 hrs]. Disponible en:*

<https://gitlab.com/Choapinus/SistemasInteligentes/blob/master/Tarea%202/enunciado_informe/Tarea%202.pdf>

Para fines demostrativos, se ejecutaron los siguientes comandos:

**$ svm\_multiclass\_learn.exe -c 5000 E\_train/E\_01.dat models/modelo\_01.dat  
$ svm\_multiclass\_learn.exe -c 5000 E\_train/E\_02.dat models/modelo\_02.dat  
$ svm\_multiclass\_learn.exe -c 5000 E\_train/E\_03.dat models/modelo\_03.dat**

**…**

Luego de obtener los modelos, se obtuvieron las predicciones con los siguientes comandos:

**$ svm\_multiclass\_classify.exe S\_01.dat modelo\_01.dat prediction\_01.dat  
$ svm\_multiclass\_classify.exe S\_02.dat modelo\_02.dat prediction\_02.dat  
$ svm\_multiclass\_classify.exe S\_03.dat modelo\_03.dat prediction\_03.dat**

**…**

Resultando en lo siguiente:

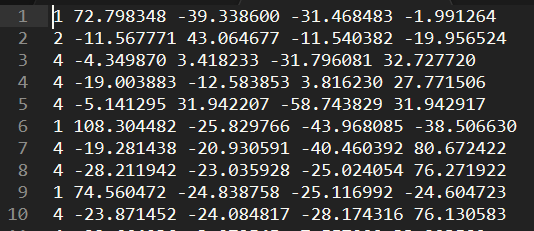


Ilustración 6.- prediction\_01.dat

# **Desarrollo**

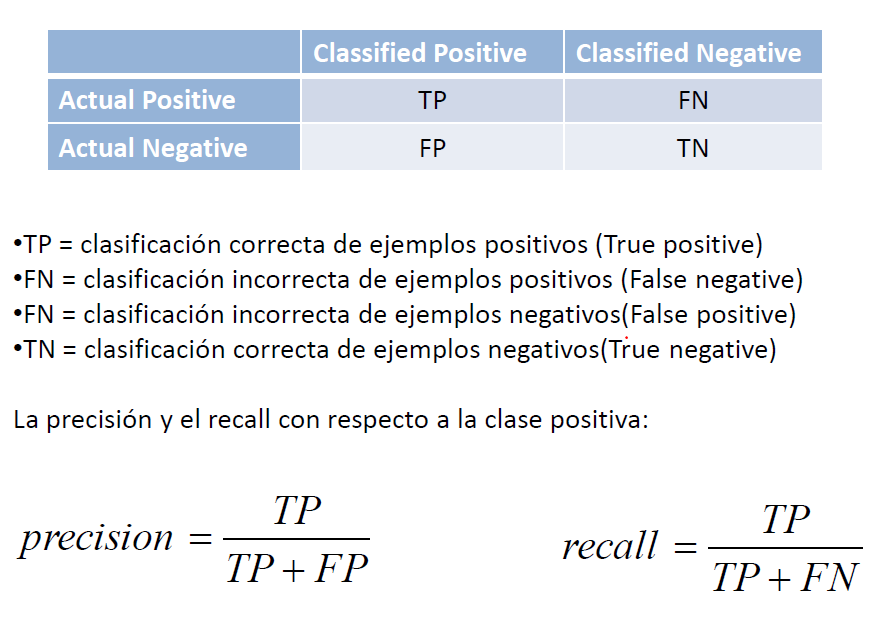
## Cuando se tengan los diez archivos de resultados, se debe calcular el desempeño del clasificador SVM para predecir la ubicación geográfica.

Luego de entrenar y clasificar, se obtuvieron los siguientes resultados:

|  |  |  |
| --- | --- | --- |
| ***Clasificador XX*** | ***Accuracy*** | ***Average Loss*** |
| Clasificador 01 | 86.5 % | 13.5 % |
| Clasificador 02 | 82 % | 18 % |
| Clasificador 03 | 81.5 % | 18.5 % |
| Clasificador 04 | 83.5 % | 16.5 % |
| Clasificador 05 | 81 % | 19 % |
| Clasificador 06 | 83.5 % | 16.5 % |
| Clasificador 07 | 84 % | 16 % |
| Clasificador 08 | 83 % | 17 % |
| Clasificador 09 | 82 % | 18 % |
| Clasificador 10 | 82 % | 18 % |

## Calcule el *MRR* y la matriz de confusión. Entregue también los promedios de los diez *splits*.

Matriz de confusión:

  
En la siguiente tabla se exponen los MRR de cada clasificador:

|  |  |
| --- | --- |
| *Clasificador XX* | *MRR* |
| Clasificador 01 | **0.9225** |
| Clasificador 02 | **0.890833333333** |
| Clasificador 03 | **0.895** |
| Clasificador 04 | **0.90125** |
| Clasificador 05 | **0.892083333333** |
| Clasificador 06 | **0.904166666667** |
| Clasificador 07 | **0.90875** |
| Clasificador 08 | **0.90625** |
| Clasificador 09 | **0.89375** |
| Clasificador 10 | **0.89875** |

*MRR* promedio: 0.901333333333

Luego, se dispone de las siguientes matrices de confusión:

|  |  |  |
| --- | --- | --- |
| Clase positiva: *Non-USA* | *Classified Positive* | *Classified Negative* |
| *Actual Positive* | 453 | 78 |
| *Actual Negative* | 264 | 1205 |

Precisión: 0.631799163  
Recall: 0.853107345

|  |  |  |
| --- | --- | --- |
| Clase positiva: *Undetermined* | *Classified Positive* | *Classified Negative* |
| *Actual Positive* | 372 | 65 |
| *Actual Negative* | 277 | 1286 |

Precisión: 0.573189522  
Recall: 0.851258581

|  |  |  |
| --- | --- | --- |
| Clase positiva: *World* | *Classified Positive* | *Classified Negative* |
| *Actual Positive* | 33 | 7 |
| *Actual Negative* | 335 | 1625 |

Precisión: 0.089673913  
Recall: 0.825

|  |  |  |
| --- | --- | --- |
| Clase positiva: *USA only* | *Classified Positive* | *Classified Negative* |
| *Actual Positive* | 800 | 192 |
| *Actual Negative* | 150 | 858 |

Precisión: 0.842105263  
Recall: 0.806451613

Promedio Precision: 0.53419196525  
Promedio Recall: 0.8339543847499999

## Comente acerca de los errores ¿Hay algún patrón?, ¿Cómo podría mejorar?, ¿Qué puede decir de los errores?, ¿Con respecto a las ubicaciones geográficas (strings) nota algún patrón en los errores?

Los errores que comete la *SVM* dependen bastante de la magnitud del entrenamiento que reciba y de la calidad de entrenamiento, es decir, si recibe un set de entrenamiento que contiene solo dos clases, las predicciones de su modelo serán inadecuadas para un set de prueba que contenga las cuatro clases. Además se sospecha de cercanías en cuanto a vectores con distintas clases asignadas (vectores similares pero catalogados con distintas clases), es decir, se estima la posibilidad de que existan vectores idénticos en cantidad de palabras y uso de las mismas palabras, por lo tanto, la SVM clasificara a ambos como iguales (siendo diferentes).

La clase *World* obtuvo la precision más baja desde la matriz de confusión. Esto podría deberse a la baja cantidad de perfiles existentes para esta clase comparados con el resto, por lo cual se entiende que su hiperplano tiende a ser más pequeño, entonces será poco probable que un vector caiga dentro de su área de probabilidad. Aun así, sus valores fueron 33 positivos y 7 negativos (17% de error si 40 fue el total de su clase), lo cual no es malo si es comparado con la clase *USA only* que más equivocaciones obtuvo. Sus valores fueron 800 positivos y 192 negativos (19% de error considerando que el total de etiquetas clasificadas bajo esta clase fueron 992). Se pone en evidencia que hubo un índice mayor de error (comparado a la clase *World*, la cual fue la segunda con mayor porcentaje de error) dado a su cantidad de etiquetas.  
Como dato anexo, las curvas *Lift* reflejan que la clase *World* fue la que mejor rendimiento obtuvo comparado con las demás clases.

Para finalizar con los errores, para el alumnado no es comprensible la ejecución de la *SVM* ni el posicionamiento de los vectores en el hiperplano. Por lo tanto, puede que se omitan algunos errores.

La *SVM* podría mejorar su índice de acierto si:

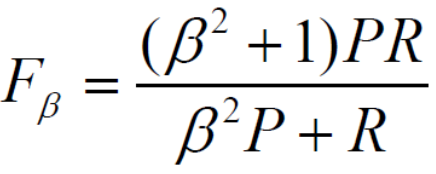
1. Se crea un ranking de las palabras más usadas y descartar las menos usadas.
2. Se descartan los perfiles con una cantidad de palabras inferior a 5 (factor propuesto al azar) y/o perfiles ambiguos.
3. Se entrena la *SVM* con un mayor espectro de datos de pruebas (1800 vectores de entrenamiento y 200 de prueba son poquísimos, algoritmos como *HyperGAN*, *CBLSTM* y *PixelCNN* son entrenados con miles de datos y sus resultados son prometedores, es decir, a mayor iteración de entrenamiento, mayor será el performance del modelo predictivo (aunque no se debería comparar redes neuronales con un clasificador *SVM*)). De todas maneras, un porcentaje de acierto del 86.5% es bastante aceptable.
4. Minimizar el error humano. Puede que la *SVM* este en lo cierto con sus predicciones …
5. Disminuir el espectro de clases (incompatible con nuestra problemática)
6. Apoyar a la *SVM* con algún otro clasificador (clasificador Bayesiano de textos)
7. Apoyar con generadores y ejecutores, los cuales vayan generando distintos sets de entrenamiento y los segundos aplicándolos a la *SVM* para luego ser comparados y verificar que set de datos obtuvo un mejor rendimiento.

## Calcule la curva ROC, Curva Lift, F(1)-score y el estimador AUC asumiendo cada una de las variables como positivas.

### F1 – Score

Es entendida como la media harmónica entre las medidas de *precision* y *recall.* Esta media tiende a estar cerca del menor de los valores (debido a la multiplicación de la formula).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *split* | *F-score* | *split* | *F-score* | *split* | *F-score* | *split* | *F-score* |
| 01 | 0.780487805 | **01** | 0.71061947 | **01** | 0.29032258 | **01** | 0.855614973 |
| 02 | 0.7875 | **02** | 0.64 | **02** | 0.142857143 | **02** | 0.821782178 |
| 03 | 0.725925926 | **03** | 0.647619048 | **03** | 0.177777778 | **03** | 0.804232804 |
| 04 | 0.74015748 | **04** | 0.673267327 | **04** | 0.29787234 | **04** | 0.827225131 |
| 05 | 0.693548387 | **05** | 0.672413793 | **05** | 0.136363636 | **05** | 0.802083333 |
| 06 | 0.736 | **06** | 0.67961165 | **06** | 0 | **06** | 0.83902439 |
| 07 | 0.709090909 | **07** | 0.728813559 | **07** | 0.2 | **07** | 0.836734694 |
| 08 | 0.679245283 | **08** | 0.685185185 | **08** | 0.19047619 | **08** | 0.839622642 |
| 09 | 0.714285714 | **09** | 0.625 | **09** | 0.666666667 | **09** | 0.828571429 |
| 10 | 0.75 | **10** | 0.714285714 | **10** | 0.181818182 | **10** | 0.772151899 |



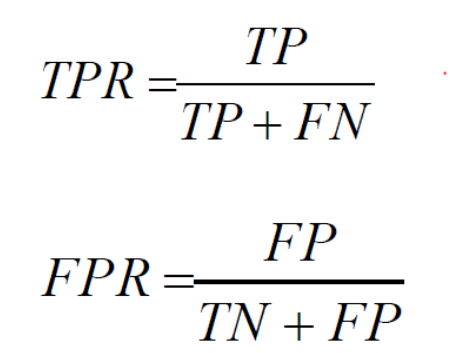
A continuación, se presentan tablas de cada split y para cada clase:

Non-USA Undetermined World USA only

### Curva ROC

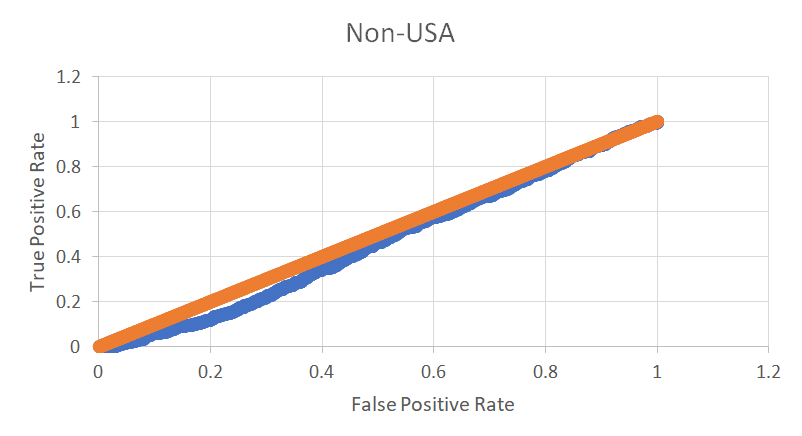
La curva *ROC* (*Receiver Operating Characteristic Curve*) es la gráfica del ratio de los verdaderos positivos vs el ratio de los falsos negativos.  
Se utiliza en la clasificación de dos clases, pero nuestra problemática consta de cuatro, por lo tanto se escoge una como positiva y las demás negativas.  
El clasificador necesita *rankear* los ejemplos de prueba de acuerdo a su probabilidad de pertenecer a la clase, dejando el más probable en el lugar más alto.

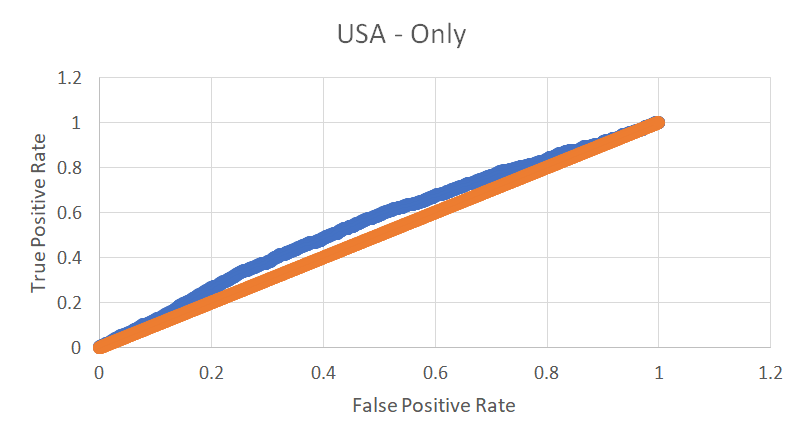
El *TPR* *(True Positive Rate)* es definido como la fracción de casos genuinos verdaderos que son correctamente clasificados.

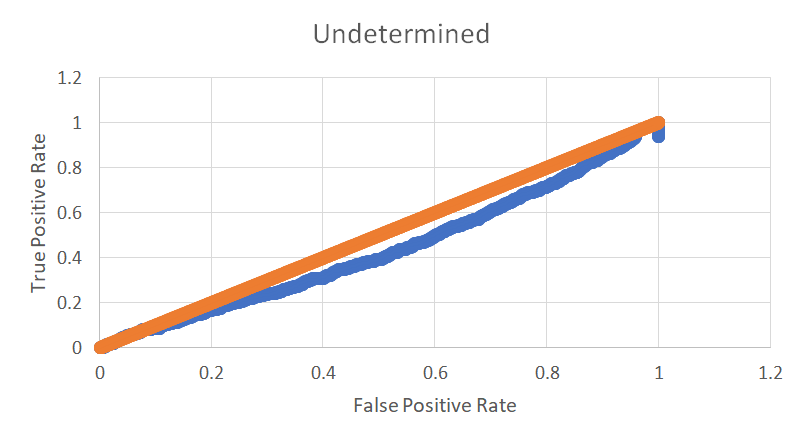


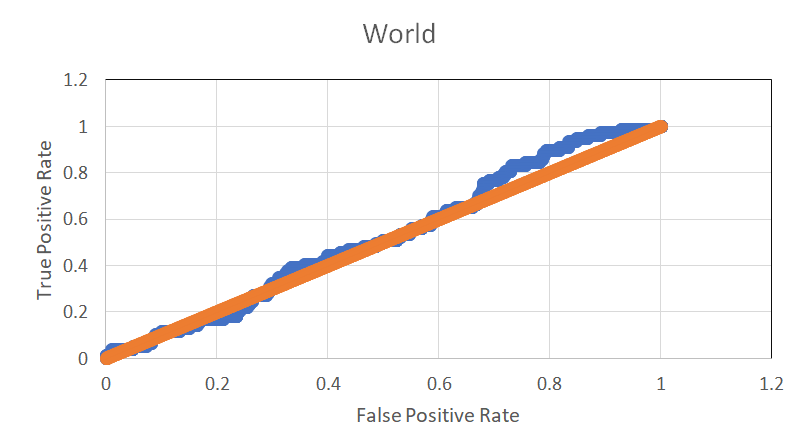
El TPR es básicamente el recall de la clase positiva y también es llamado “sensitividad”.

A continuación, se dispone de las curvas ROC de las distintas clases:





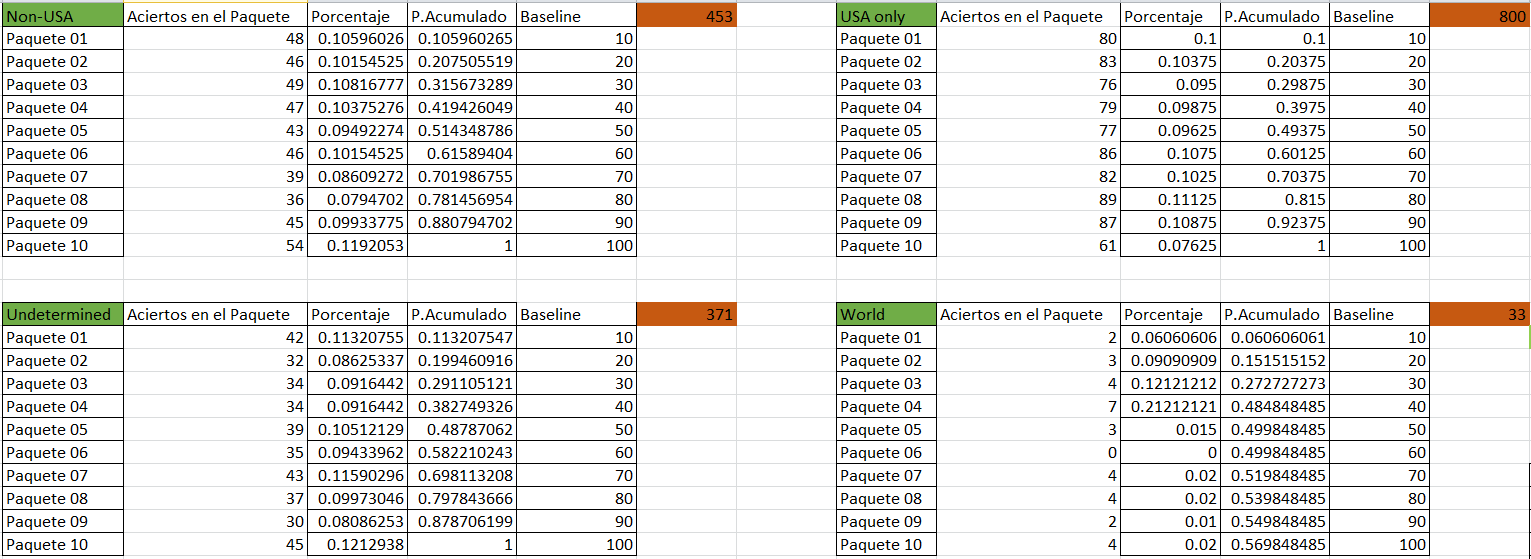




### Curva Lift

En minería de datos, *Lift* es la medida del performance de un modelo prediciendo o clasificando, medido contra un modelo de selección aleatoria (línea 0,0 – 1,1). Se considera que el modelo está realizando un buen trabajo si su predicción es mejor que el promedio del clasificador aleatorio (es decir, si la curva de rendimiento supera a la recta del clasificador de selección aleatoria, el modelo evaluado es mejor).

A continuación, se dispone de los datos para realizar las curvas *Lift*:



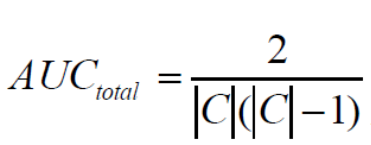
Luego, se presentan las curvas:

Pero mira esa curva papu

### Estimador AUC

El estimador *AUC(ci, cj)* representa el área bajo la curva ROC construida con las clases *ci* y *cj*.

Se presenta la fórmula del estimador:



Siendo | C | la cantidad de clases (4).

El AUC para la presente problemática equivale a 0.166666666.

# **Conclusión**

Para finalizar, se declara que la SVM Multiclass obtuvo un rendimiento aceptable en cuanto a clasificación de etiquetas y cometió una cantidad de errores relativamente cercano a lo que cometería una persona (342 etiquetas mal clasificadas).

La SVM debe ser entrenada con anterioridad y se estima que a mayor magnitud de entrenamiento, mejor rendimiento poseerán los modelos predictivos. Aun así, el entrenamiento debe ser algo vectorizado y se estima que deben existir mejores vectores que los creados por el alumnado. Además, se pone en evidencia que los vectores no siempre serán los mismos debido al orden de llegada de los perfiles, es decir, no será lo mismo vectorizar los perfiles desde el primero al último que desde el ultimo al primero (debido a los id’s asignados a las palabras en la bolsa de palabras), mayor se agrava la situación dependiendo de la cantidad de stopwords que se descuenten ya que esto puede desviar los vectores a sectores no deseados dentro del hiperplano.

Como conclusión personal, encuentro que fue una tarea bastante entretenida. Ver como algo que conoce absolutamente nada de una problemática clasifique de una manera tan aproximada a como yo, me asombro bastante. Aprender y tener en cuenta que los espacios vectoriales no son solo números, sino espacios verídicos en donde se puede catalogar algo también puede ser útil para varias problemáticas y clasificación de distintos problemas (casi como la teoría de conjuntos). Espero con ansias aprender sobre redes neuronales y algoritmos genéticos.