UNIVERSIDAD ANDRÉS BELLO

FACULTAD DE INGENIERÍA

INGENIERÍA CIVIL INFORMÁTICA

Trabajo 2 – Aprendizaje Supervisado (SVM)

ANDRÉS EDUARDO VALENZUELA GONZÁLEZ

SANTIAGO - CHILE

SEPTIEMBRE, 2017

Contenido

1. Introducción

Para comprender el desarrollo de este informe, se utilizaron las dos mil etiquetas junto con sus perfiles de la tarea anterior para ser vectorizadas y luego utilizar los algoritmos de *SVM* (y otros creados por el alumnado) para lograr un aprendizaje supervisado.

Junto con las etiquetas proporcionadas (en formato *csv*) por el servidor de la *Universidad Nacional Andrés Bello*, se proporcionó además el conjunto de perfiles que fueron etiquetados por el alumnado (en formato *dat*), los cuales fueron unidos para luego generar una bolsa de palabras, de las cuales, además, se descontaron las *stopwords*.

Para realizar el aprendizaje 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: 13 septiembre 2017, 20:12 hrs]. Disponible en:* https://gitlab.com/Choapinus/SistemasInteligentes/blob/master/Tarea%202/enunciado_informe/Tarea%202.pdf

2. Vectorización

Para explicar el proceso de vectorización, se considerara el siguiente ejemplo que trabaja con preguntas *cQA* (cada perfil es una pregunta):

```
Título y Cuerpo (Unidos) Etiqueta (E)
Pregunta 1 (P1) What type of ham is your favorite? My favorite...1
Pregunta 2 (P2) What website will let you ask questions? ...     3
Pregunta 3 (P3) Test post please ignore? LOL...                  3
Pregunta 4 (P4) Spam, baloney, or ramon noodles? Which is ...    1
Pregunta 5 (P5) Which restaurant has the best tasting onion ...  2
Pregunta 6 (P6) Is Argentina a first world coutnry?...           1
Pregunta 7 (P7) What rugby position am I?...                     3
```

El primer paso sería construir una lista de palabras (Bolsa de palabras) para toda la colección. Para este ejemplo (case-insensitive), tendríamos;

ID	Palabra	ID	Palabra	ID	Palabra
1	WHAT	6	FAVORITE	11	ASK
2	TYPE	7	MY	12	QUESTIONS
3	OF	8	WILL	13	TEST
4	HAM	9	LET	14	POST
5	YOUR	10	YOU	15	PLEASE

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:

```
1 1:1 2:1 3:5 4:3 5:4 6:5 8:1 9:1 12:5 13:1 14:1
3 1:4 2:1 3:1 5:4 6:1 7:1 8:6 9:6 11:3 13:1 14:2 15:2
3 2:5 4:4 5:3 6:4 7:2 8:5 9:1 10:6 11:1 12:3 13:6 14:1 15:1
1 1:4 2:2 3:4 4:1 5:2 6:4 7:4 8:3 10:5 11:3 12:2 13:1 14:2 15:3
2 2:3 3:2 4:2 5:6 7:1 8:3 9:3 10:2 11:1 12:6 13:1 14:6 15:3
```

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:

```
1 204:1 537:1 569:1 1097:1 1265:1 1297:1 1355:1 1475:2 6214:1 6553:1 7822:1 8333:1 8779:1 9115:1 9321:1 9536:1 9945:1 10011:1 11896:1
12311:1 13186:3 14012:1 14207:1 14535:1
1 144:1 482:1 534:1 1086:1 1279:1 1717:1 1801:1 2195:1 2601:1 2667:1 3007:1 3336:1 3899:1 3981:1 4167:1 5633:1 6090:1 6249:3 6276:1 6366:1
6898:1 7820:1 8046:1 9145:1 9374:1 10363:1 10891:1 12907:1 13229:1 14601:1
1 251:1 510:1 698:1 1180:1 5158:1 6141:1 7029:1 7179:1 8447:1 8698:1 9582:1 10193:1 11472:1 12063:1 14640:1 14690:1 16286:3
1 1886:1 7105:1 7239:1 11105:1 12924:1 13133:1 13982:1 15693:1
1 2107:1 2358:1 3050:2 3360:1 3599:1 5933:2 7078:1 8561:2 9057:1 9275:1 9329:1 10355:1 10525:1 13278:1 14004:1 14125:1 14681:1 14690:1
15840:1
```

Ilustración 1.- 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

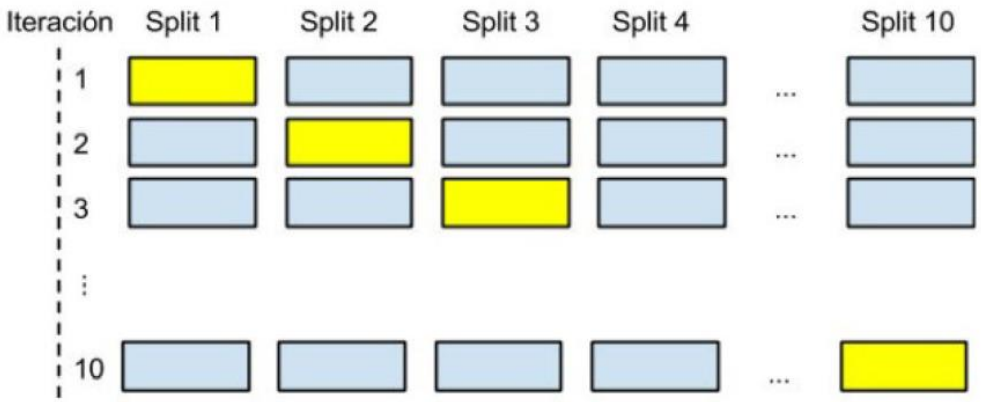
Y del código que genera los vectores con el csv del alumnado:

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.

3. 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



10-Fold Cross-Validation. Los cuadrados amarillos representan al set de evaluación y los grises al de entrenamiento.

Ilustración 2.- 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).

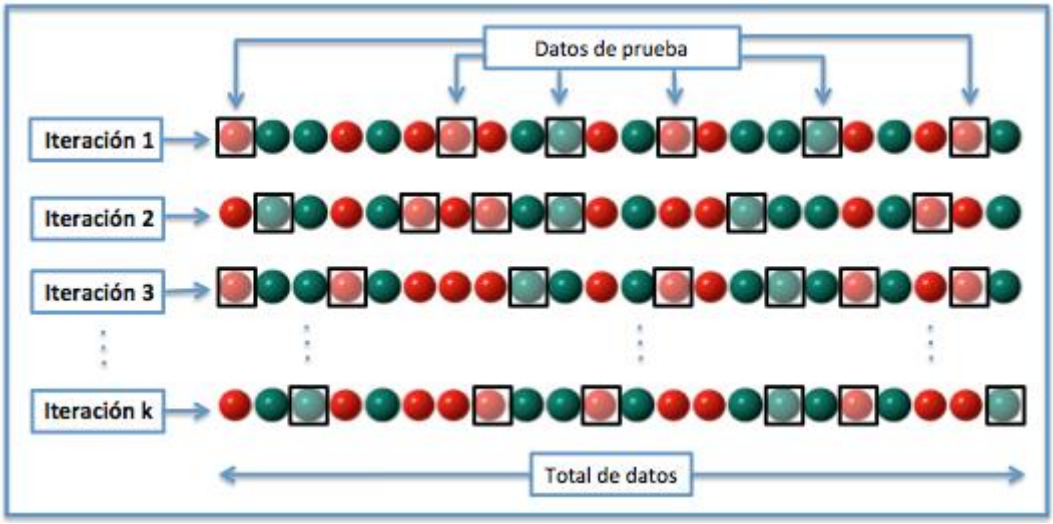


Ilustración 3.- Random Sampling Cross-Validation

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.

4. 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:

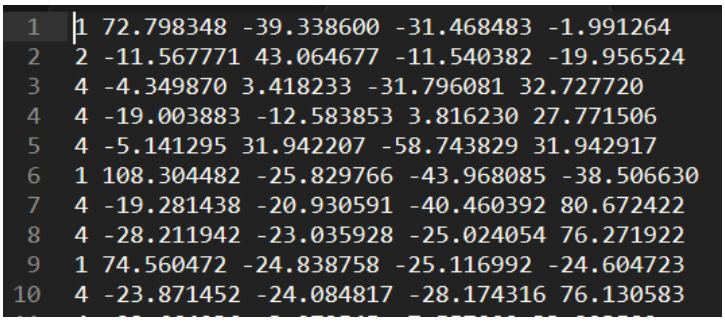


Ilustración 4.- prediction_01.dat

5. Desarrollo

5.1. 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 %

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

Matriz de confusión:

	Classified Positive	Classified Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

- TP = clasificación correcta de ejemplos positivos (True positive)
- FN = clasificación incorrecta de ejemplos positivos (False negative)
- FN = clasificación incorrecta de ejemplos negativos(False positive)
- TN = clasificación correcta de ejemplos negativos(True negative)

La precisión y el recall con respecto a la clase positiva:

$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN}$$

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

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?

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.

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

5.4.1. 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).

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

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

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
Non-USA		Undetermined		World		USA only	

5.4.2. 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.

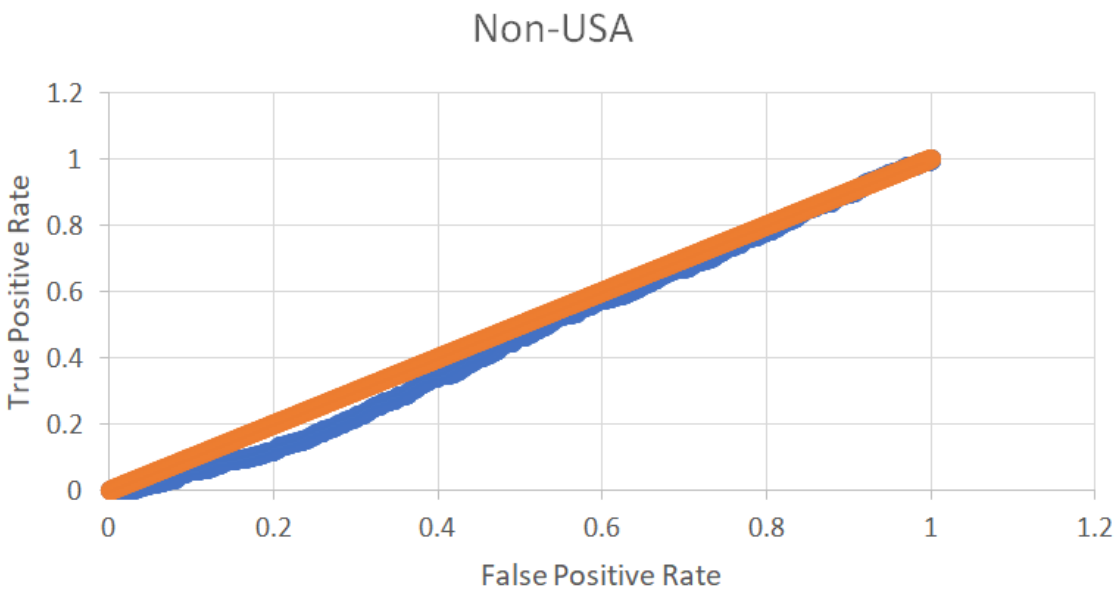
El *TPR (True Positive Rate)* es definido como la fracción de casos genuinos verdaderos que son correctamente clasificados.

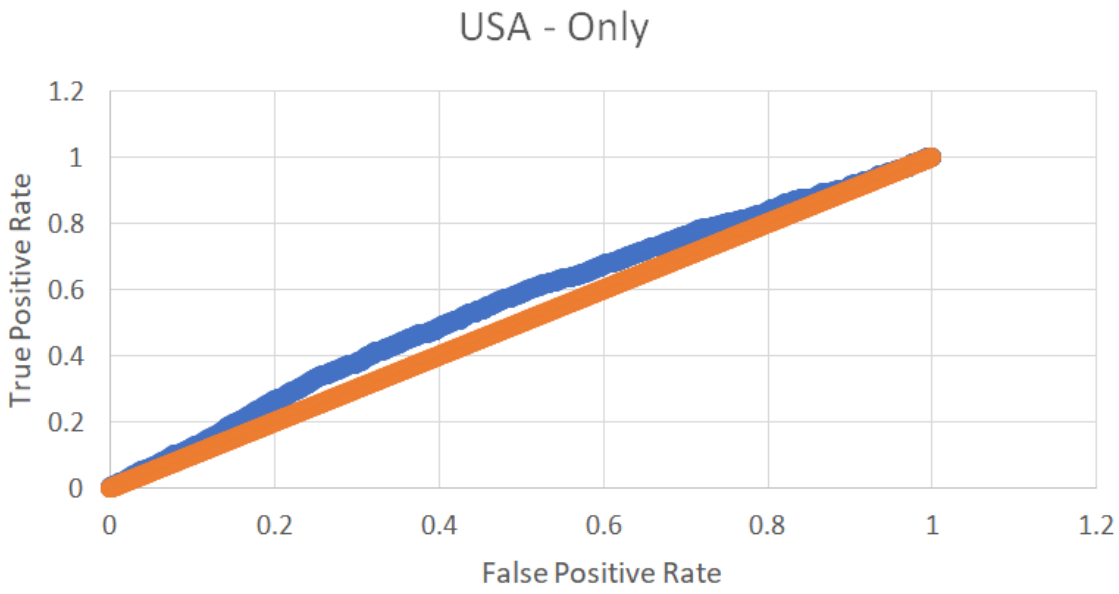
$$TPR = \frac{TP}{TP + FN}$$

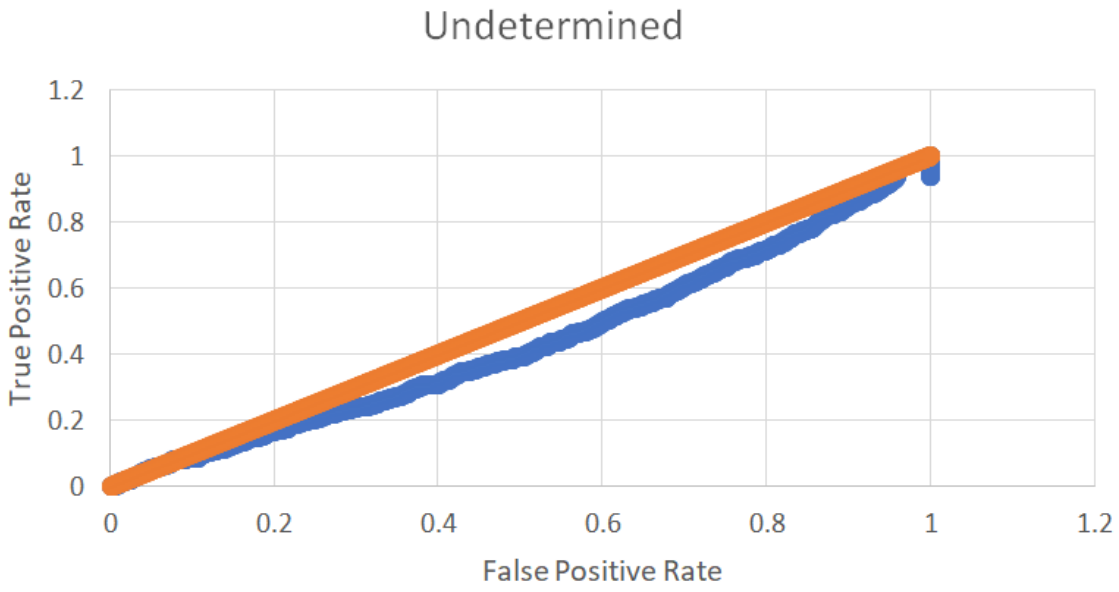
$$FPR = \frac{FP}{TN + FP}$$

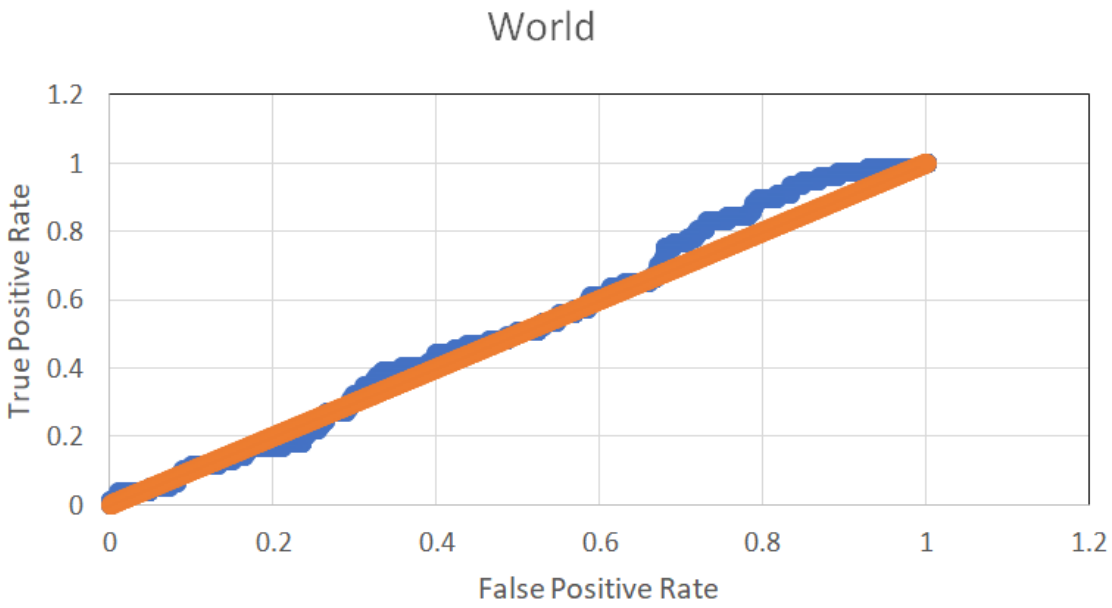
El TPR es básicamente el recall de la clase positiva y también es llamado “sensitividad”.

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




USA - Only
True Positive Rate
False Positive Rate
0
0.2
0.4
0.6
0.8
1
1.2


Undetermined
True Positive Rate
False Positive Rate
0
0.2
0.4
0.6
0.8
1
1.2



5.4.3. 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*:

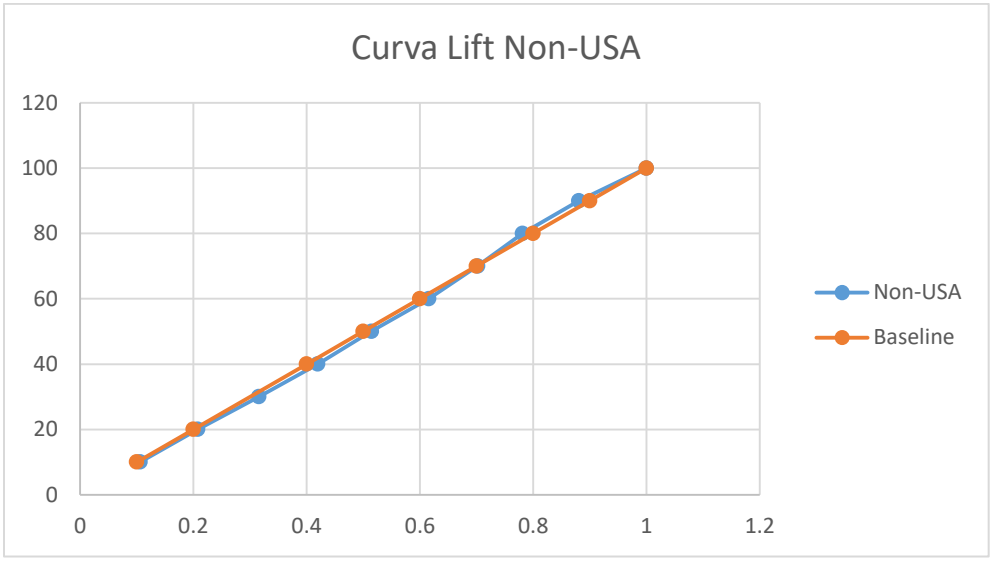
Non-USA	Aciertos en el Paquete	Porcentaje	P.Acumulado	Baseline	453
Paquete 01	48	0.10596026	0.105960265	10	
Paquete 02	46	0.10154525	0.207505519	20	
Paquete 03	49	0.10816777	0.315673289	30	
Paquete 04	47	0.10375276	0.419426049	40	
Paquete 05	43	0.09492274	0.514348786	50	
Paquete 06	46	0.10154525	0.61589404	60	
Paquete 07	39	0.08609272	0.701986755	70	
Paquete 08	36	0.0794702	0.781456954	80	
Paquete 09	45	0.09933775	0.880794702	90	
Paquete 10	54	0.1192053	1	100	

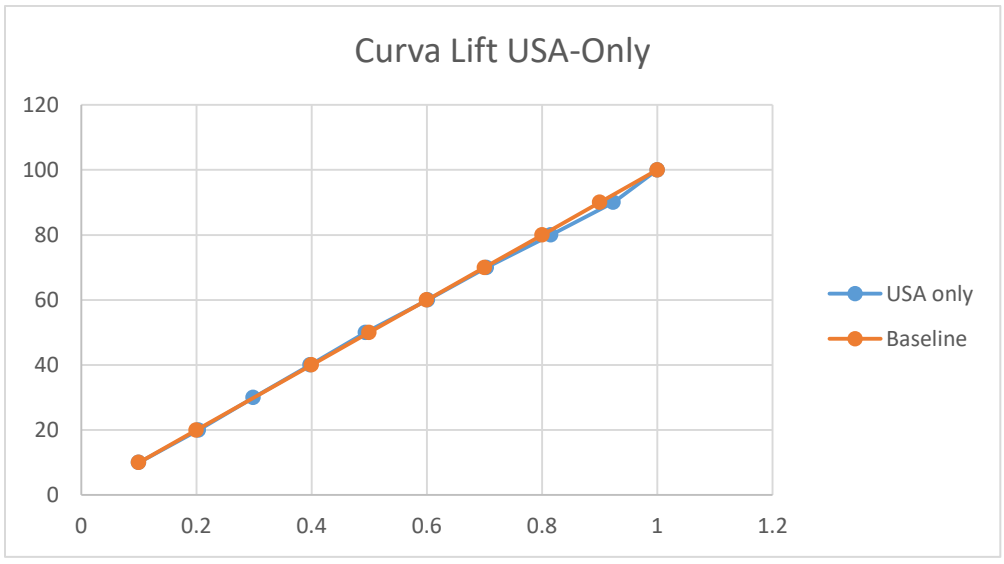
USA only	Aciertos en el Paquete	Porcentaje	P.Acumulado	Baseline	800
Paquete 01	80	0.1	0.1	10	
Paquete 02	83	0.10375	0.20375	20	
Paquete 03	76	0.095	0.29875	30	
Paquete 04	79	0.09875	0.3975	40	
Paquete 05	77	0.09625	0.49375	50	
Paquete 06	86	0.1075	0.60125	60	
Paquete 07	82	0.1025	0.70375	70	
Paquete 08	89	0.11125	0.815	80	
Paquete 09	87	0.10875	0.92375	90	
Paquete 10	61	0.07625	1	100	

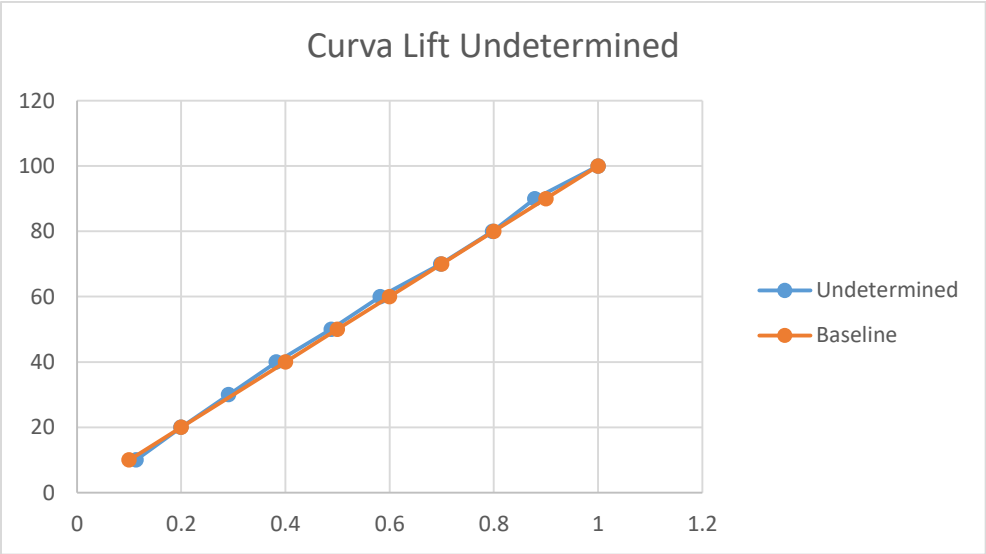
Undetermined	Aciertos en el Paquete	Porcentaje	P.Acumulado	Baseline	371
Paquete 01	42	0.11320755	0.113207547	10	
Paquete 02	32	0.08625337	0.199460916	20	
Paquete 03	34	0.0916442	0.291105121	30	
Paquete 04	34	0.0916442	0.382749326	40	
Paquete 05	39	0.10512129	0.48787062	50	
Paquete 06	35	0.09433962	0.582210243	60	
Paquete 07	43	0.11590296	0.698113208	70	
Paquete 08	37	0.09973046	0.797843666	80	
Paquete 09	30	0.08086253	0.878706199	90	
Paquete 10	45	0.1212938	1	100	

World	Aciertos en el Paquete	Porcentaje	P.Acumulado	Baseline	33
Paquete 01	2	0.06060606	0.060606061	10	
Paquete 02	3	0.09090909	0.151515152	20	
Paquete 03	4	0.12121212	0.272727273	30	
Paquete 04	7	0.21212121	0.484848485	40	
Paquete 05	3	0.015	0.499848485	50	
Paquete 06	0	0	0.499848485	60	
Paquete 07	4	0.02	0.519848485	70	
Paquete 08	4	0.02	0.539848485	80	
Paquete 09	2	0.01	0.549848485	90	
Paquete 10	4	0.02	0.569848485	100	

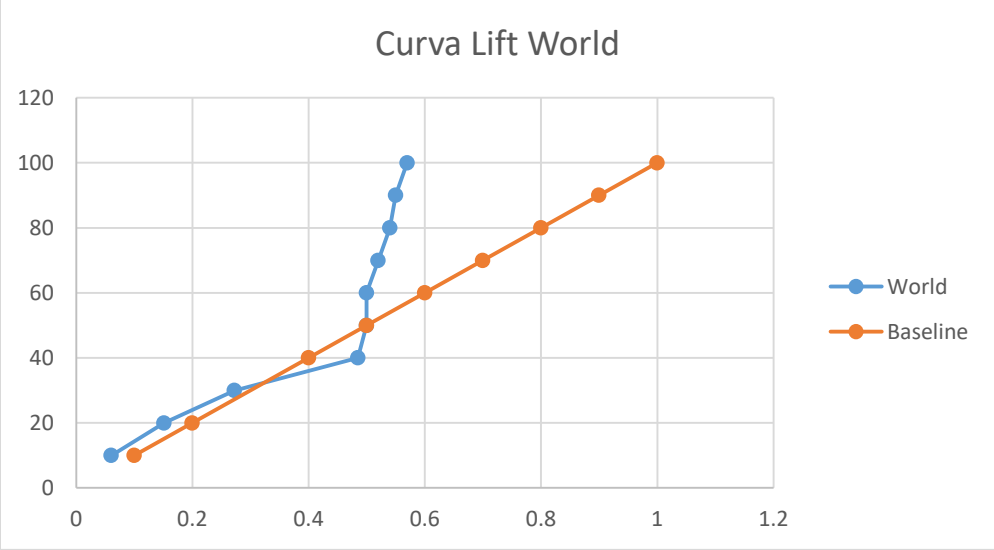
Luego, se presentan las curvas:







Pero mira esa curva papu



5.4.4. 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:

$$AUC_{total} = \frac{2}{|C|(|C|-1)}$$

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

El AUC para la presente problemática equivale a 0.166666666.

6. 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.