Condiciones iniciales

Vectorizador: TfidfVectorizer

- **max_features = 10000** # Máximo de palabras que toma para la matriz
- **min_df = 0.0001** # Toma todas las palabras que tengan este mínimo de frecuencia
- **ngram_range = (1,1) || (1,2) || (2,2)**

Train-Test Split:

- Tamaño del set de entrenamiento: **15839**
- Tamaño del set de prueba: **6789**

A continuación, se analizarán 3 modelos predictivos y se entrenarán con datos lematizados y stemmizados, y también probando con monogramas, monogramas+bigramas y únicamente bigramas.

En el siguiente cuadro se sintetizan los resultados, evaluados por accuracy en la validación cruzada:

Modelo	Datos	Monogramas	Monogramas + Bigramas	Bigramas
Naive Bayes (Benchmark)	Lemma	0.40	0.70	0.73
	Stem	0.38	0.71	0.74
Random Forest	Lemma	0.87	0.88	0.86
	Stem	0.87	0.87	0.86
SVM	Lemma	**0.90**	0.90	0.86
	Stem	0.90	0.90	0.86

Conclusión:

Se elige el modelo SVM con monogramas lematizados, ya que fue el que demostró un mejor desempeño y también fue el más equilibrado en cuanto a precisión y recall, ya que no se quiere un modelo que tienda a clasificar siempre con el mismo valor ante la duda.

Naive Bayes - Lemma - Monogramas

Naive Bayes - Modelo Benchmark

```
nb = GaussianNB()
nb.fit(xtrain,ytrain)

y_pred = nb.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_s
```

```
Accuracy :  0.41036971571660036
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(nb,X,recom,5)
```

```
Validación cruzada:
0.40 de precisión con desviación estándar de 0.01
```

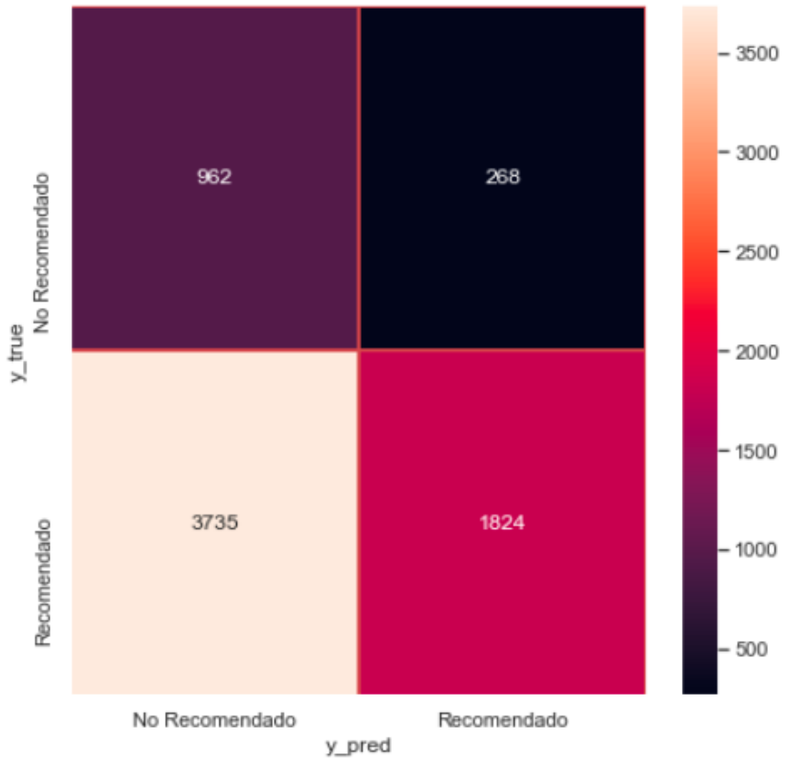
```
print(color.BOLD + 'Reporte de clasificación : '+ color.
```

```
Reporte de clasificación :
```

	precision	recall	f1-score	support
No Rec	0.20	0.78	0.32	1230
Rec	0.87	0.33	0.48	5559
accuracy			0.41	6789
macro avg	0.54	0.56	0.40	6789
weighted avg	0.75	0.41	0.45	6789

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```



Observaciones:

Se ve que identifica bien a los no recomendados, pero tiene una tendencia a predecir que una prenda no será recomendada y se manifiesta en la baja precisión de la clase “No recomendado”.

Con estos parámetros, el modelo no termina siendo efectivo.

Naive Bayes - Lemma - Monogramas + Bigramas

Naive Bayes - Modelo Benchmark

```
nb = GaussianNB()
nb.fit(xtrain,ytrain)

y_pred = nb.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_s
```

```
Accuracy :  0.701870673147739
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(nb,X,recom,5)
```

```
Validación cruzada:
0.70 de precisión con desviación estándar de 0.01
```

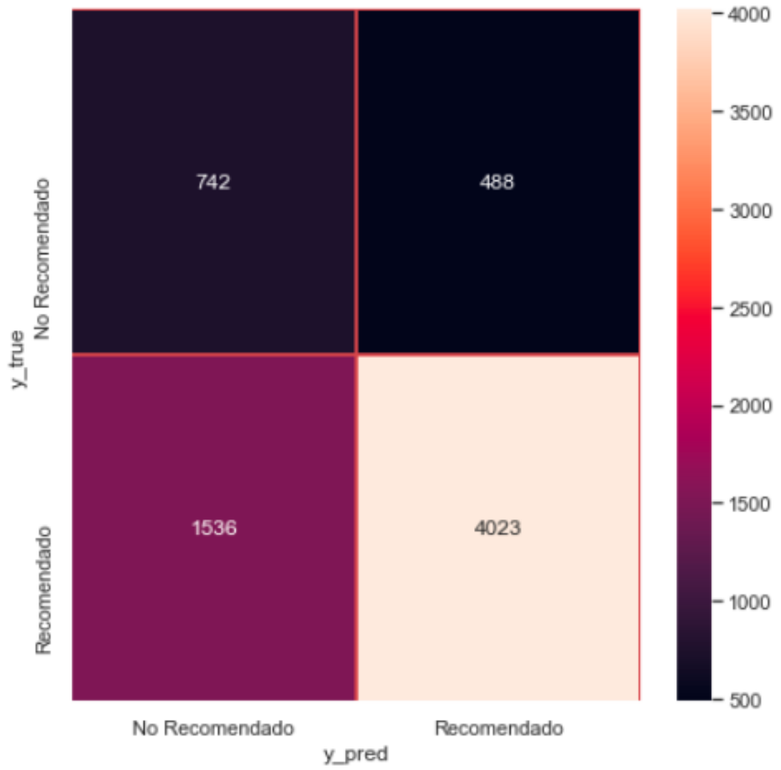
```
print(color.BOLD + 'Reporte de clasificación : '+ color.
```

```
Reporte de clasificación :
```

	precision	recall	f1-score	support
No Rec	0.33	0.60	0.42	1230
Rec	0.89	0.72	0.80	5559
accuracy			0.70	6789
macro avg	0.61	0.66	0.61	6789
weighted avg	0.79	0.70	0.73	6789

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```



Observaciones:

Aumenta considerablemente su score respecto del modelo que sólo tomaba monogramas. Sin embargo, todavía no logra diferenciar del todo bien entre recomendados y no recomendados.

Naive Bayes - Lemma - Bigramas

Naive Bayes - Modelo Benchmark

```
nb = GaussianNB()
nb.fit(xtrain,ytrain)

y_pred = nb.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_s
```

```
Accuracy :  0.7207247017233761
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(nb,X,recom,5)
```

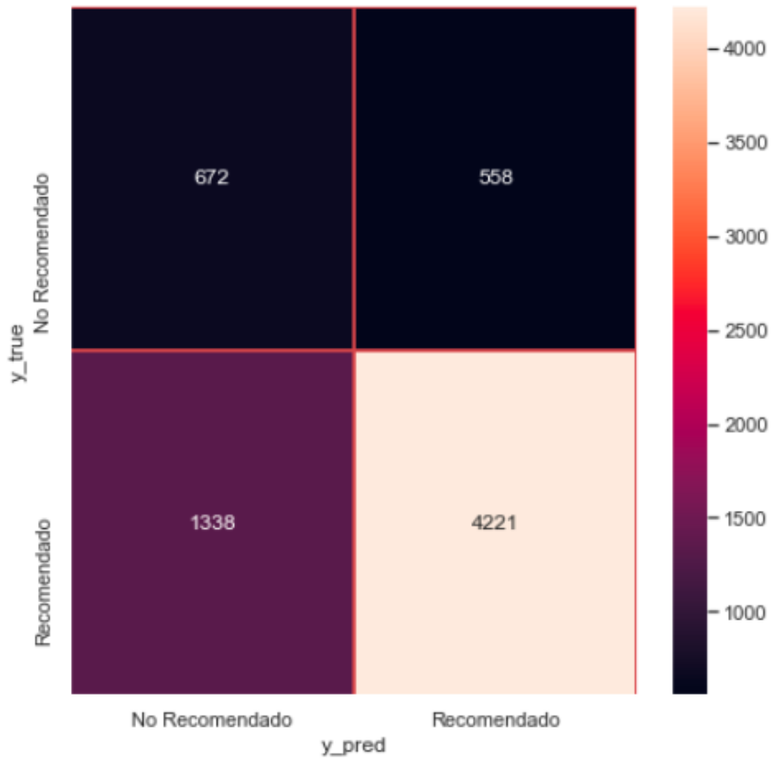
```
Validación cruzada:
0.73 de precisión con desviación estándar de 0.01
```

```
print(color.BOLD + 'Reporte de clasificación : '+ color.
```

```
Reporte de clasificación :

              precision    recall  f1-score   support

      No Rec       0.33      0.55      0.41      1230
         Rec       0.88      0.76      0.82      5559

    accuracy                           0.72      6789
   macro avg       0.61      0.65      0.62      6789
weighted avg       0.78      0.72      0.74      6789
```

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```



Observaciones:

Hay una leve mejora respecto del modelo anterior al tomar sólo bigramas, ya que se hace más preciso. Pero aún hay un gran porcentaje de errores. De hecho, bajó el recall de los no recomendados.

Naive Bayes - Stem - Monogramas

Naive Bayes - Modelo Benchmark

```
nb = GaussianNB()
nb.fit(xtrain,ytrain)

y_pred = nb.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_
```

```
Accuracy :  0.3913683900427162
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(nb,X,recom,5)
```

```
Validación cruzada:
0.38 de precisión con desviación estándar de 0.00
```

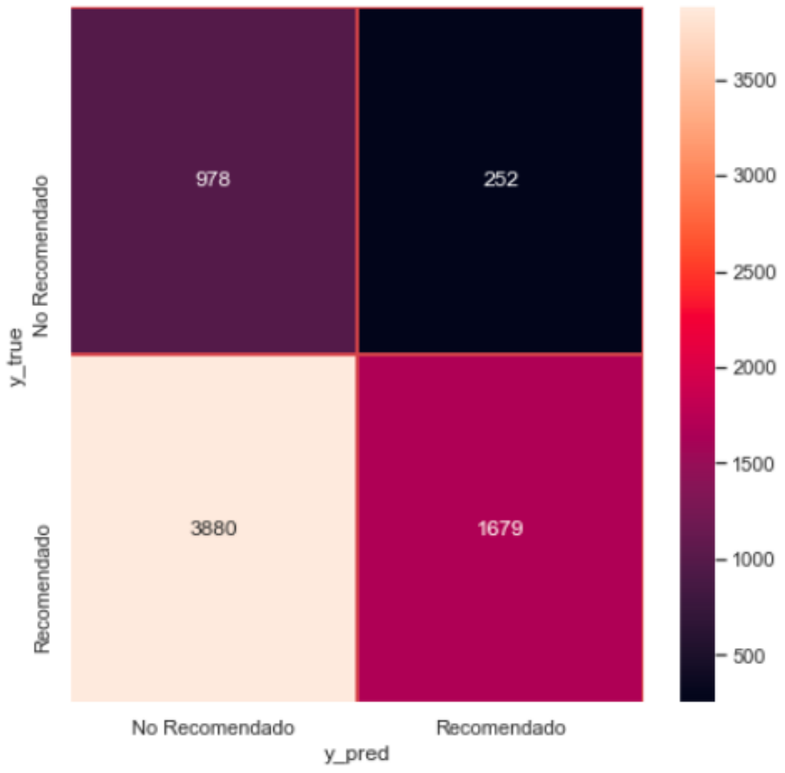
```
print(color.BOLD + 'Reporte de clasificación : '+ color
```

```
Reporte de clasificación :
```

	precision	recall	f1-score	support
No Rec	0.20	0.80	0.32	1230
Rec	0.87	0.30	0.45	5559
accuracy			0.39	6789
macro avg	0.54	0.55	0.38	6789
weighted avg	0.75	0.39	0.43	6789

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```



Observaciones:

Tiene peor desempeño que el mismo modelo con las reviews lematizadas. Confunde gravemente las recomendadas con las no recomendadas. No logra distinguir bien los límites.

Naive Bayes - Stem - Monogramas + Bigramas

Naive Bayes - Modelo Benchmark

```
nb = GaussianNB()
nb.fit(xtrain,ytrain)

y_pred = nb.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_s
```

```
Accuracy :  0.7082044483723671
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(nb,X,recom,5)
```

```
Validación cruzada:
0.71 de precisión con desviación estándar de 0.00
```

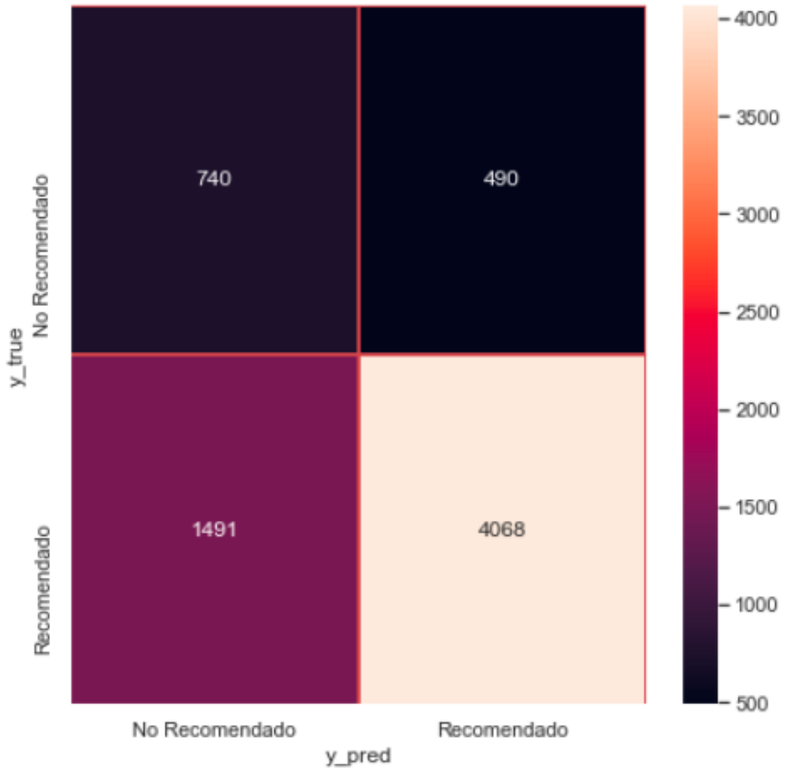
```
print(color.BOLD + 'Reporte de clasificación : '+ color.
```

```
Reporte de clasificación :
```

	precision	recall	f1-score	support
No Rec	0.33	0.60	0.43	1230
Rec	0.89	0.73	0.80	5559
accuracy			0.71	6789
macro avg	0.61	0.67	0.62	6789
weighted avg	0.79	0.71	0.74	6789

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```



Observaciones:

Aumenta la precisión en la identificación de las clases respecto del modelo con el dataset lemmatizado, pero aún siguen habiendo confusiones que hacen que su rendimiento no sea bueno.

Naive Bayes - Stem - Bigramas

Naive Bayes - Modelo Benchmark

```
nb = GaussianNB()
nb.fit(xtrain,ytrain)

y_pred = nb.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_
```

```
Accuracy :  0.7266165856532626
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(nb,X,recom,5)
```

```
Validación cruzada:
0.74 de precisión con desviación estándar de 0.00
```

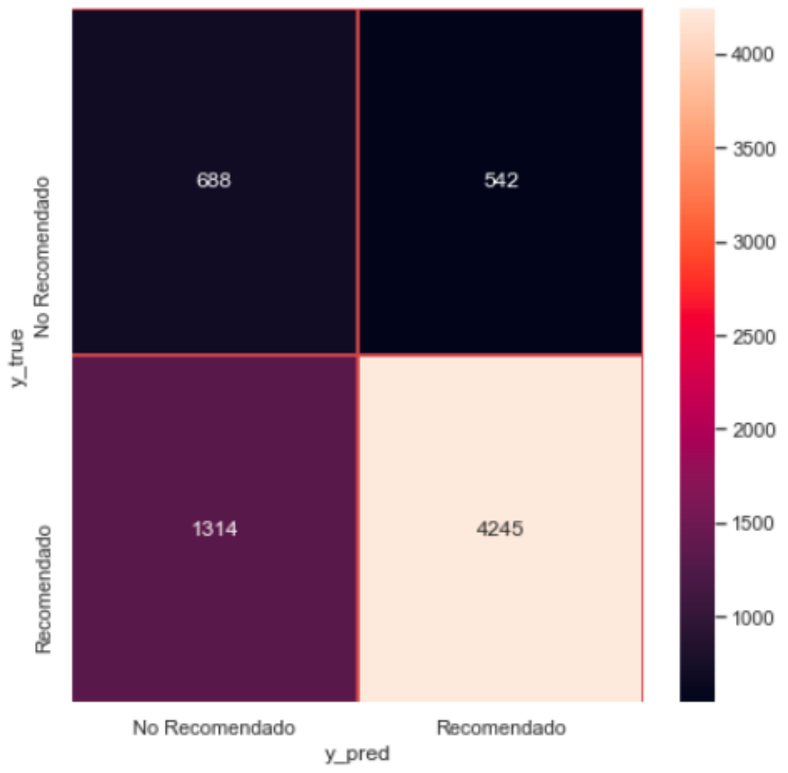
```
print(color.BOLD + 'Reporte de clasificación : '+ color
```

```
Reporte de clasificación :
```

	precision	recall	f1-score	support
No Rec	0.34	0.56	0.43	1230
Rec	0.89	0.76	0.82	5559
accuracy			0.73	6789
macro avg	0.62	0.66	0.62	6789
weighted avg	0.79	0.73	0.75	6789

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```



Observaciones:

Si bien mejora levemente su score respecto de su par lemmatizado, sigue siendo muy desparejo en las métricas de precisión y recall entre clases: las de una son muy buenas y las de la otra son muy bajas.

Random Forest - Lemma - Monogramas

Random Forest

```
rf = RandomForestClassifier()
rf.fit(xtrain,ytrain)

y_pred = rf.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_
```

```
Accuracy :  0.8755339519811459
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(rf,X,recom,3)
```

```
Validación cruzada:
0.87 de precisión con desviación estándar de 0.00
```

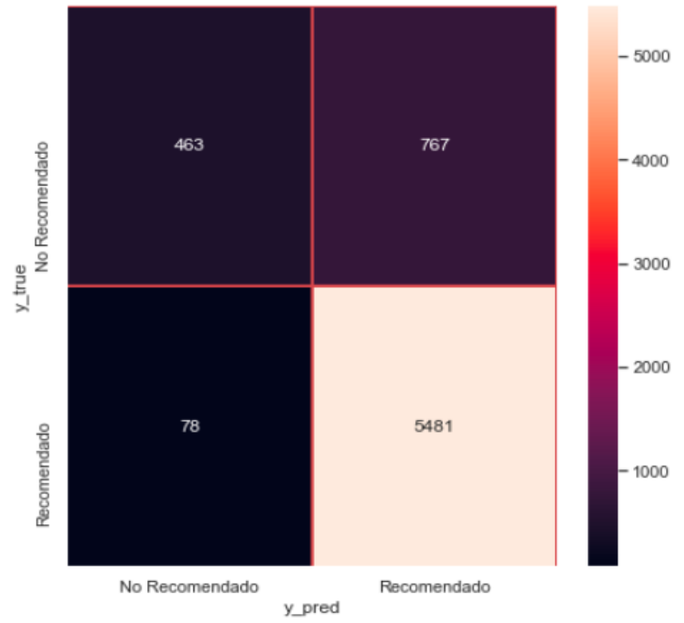
```
print(color.BOLD + 'Reporte de clasificación : '+ color.
```

```
Reporte de clasificación :
```

	precision	recall	f1-score	support
No Rec	0.86	0.38	0.52	1230
Rec	0.88	0.99	0.93	5559
accuracy			0.88	6789
macro avg	0.87	0.68	0.73	6789
weighted avg	0.87	0.88	0.85	6789

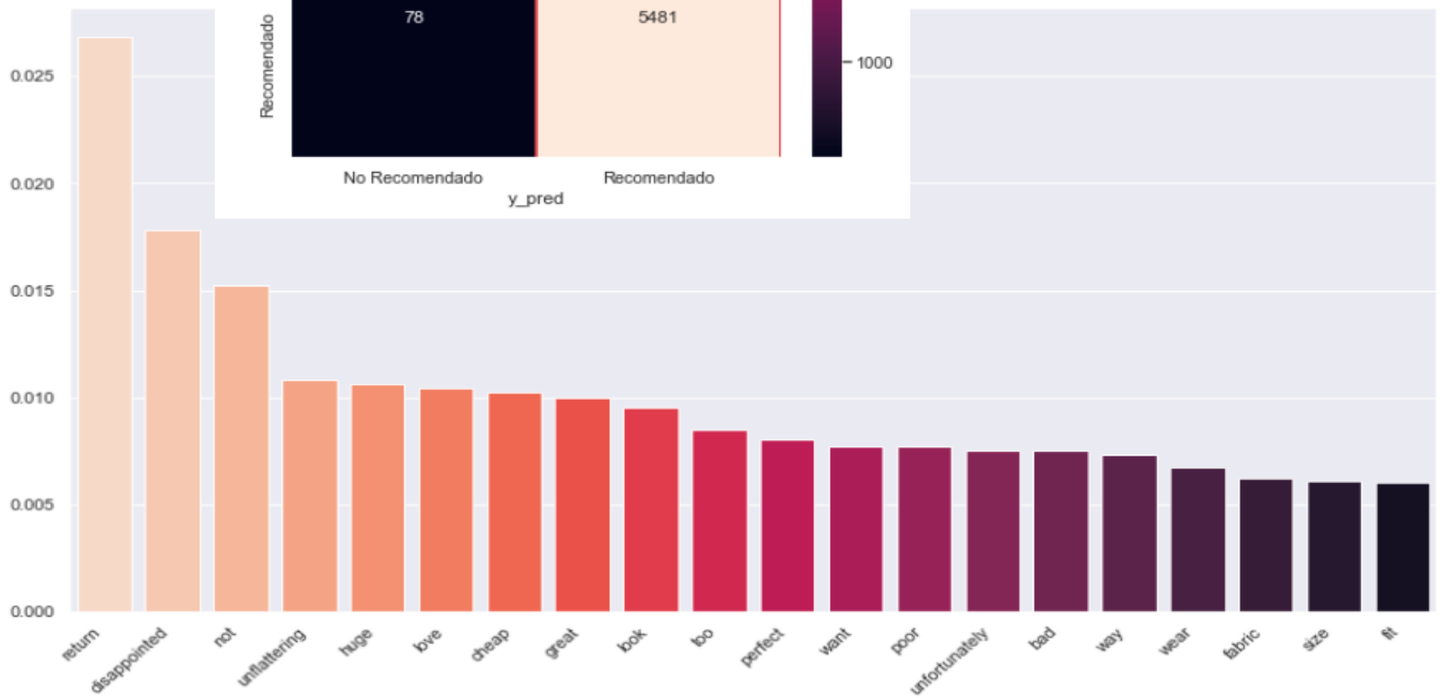
```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```



Observaciones:

Aumenta la precisión para reconocer las clases recomendadas, pero aún confunde con las no recomendadas. Pareciera que tiende a decir que todas son de la clase “Recomendadas”.



Random Forest - Lemma - Monogramas + Bigramas

Random Forest

```
rf = RandomForestClassifier()
rf.fit(xtrain,ytrain)

y_pred = rf.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_
```

```
Accuracy :  0.8773015171601l2
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(rf,X,recom,3)
```

```
Validación cruzada:
0.88 de precisión con desviación estándar de 0.00
```

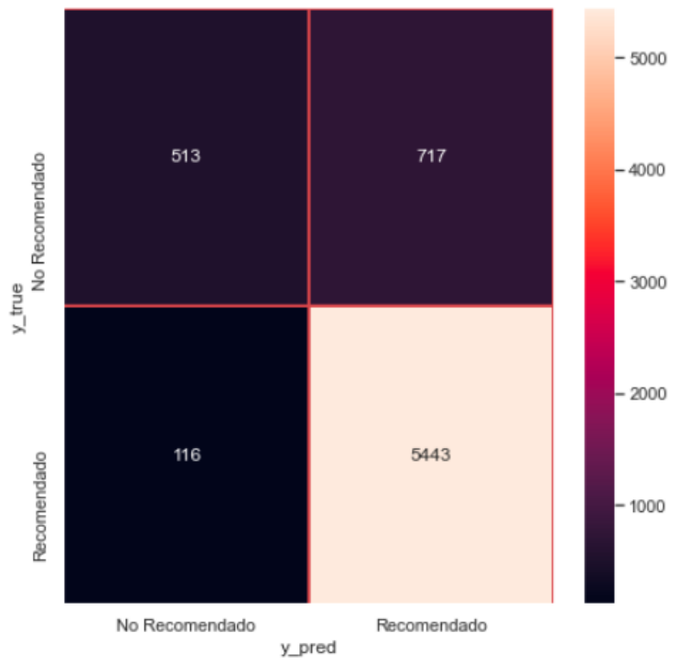
```
print(color.BOLD + 'Reporte de clasificación : '+ color
```

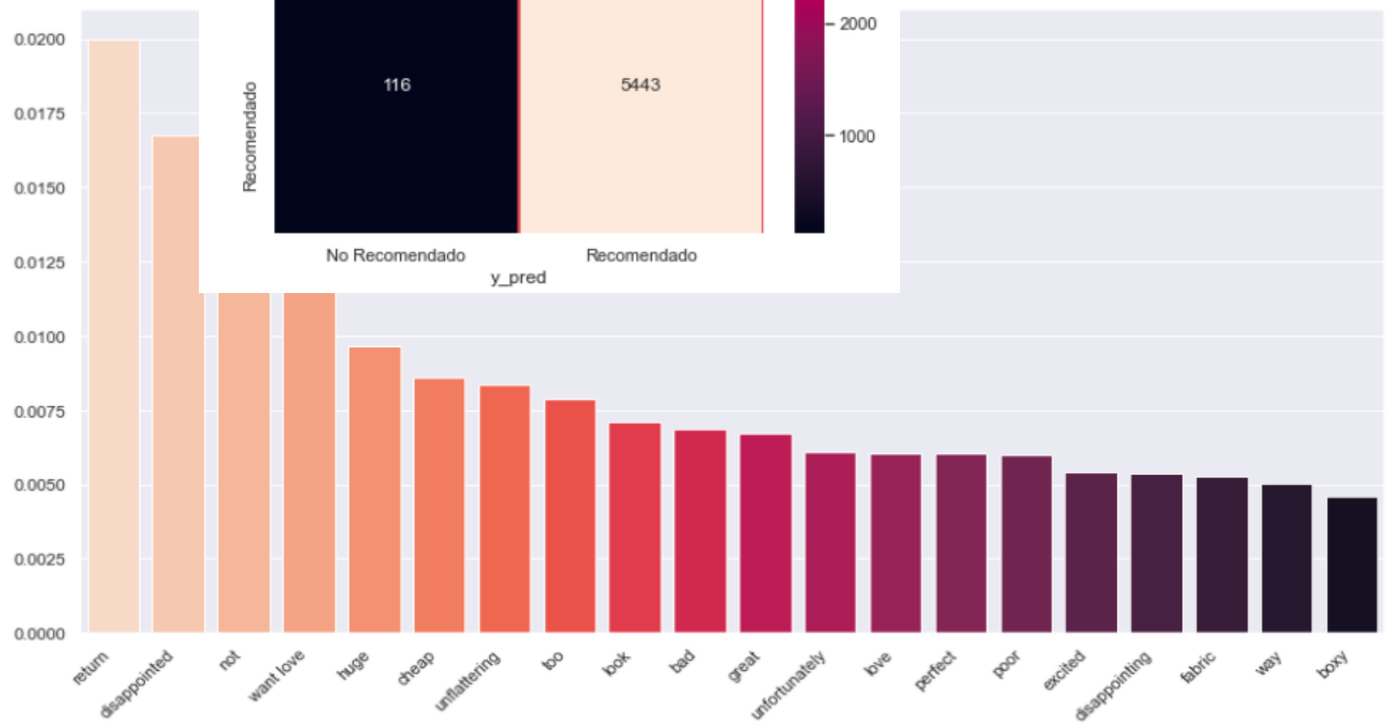
```
Reporte de clasificación :
```

	precision	recall	f1-score	support
No Rec	0.82	0.42	0.55	1230
Rec	0.88	0.98	0.93	5559
accuracy			0.88	6789
macro avg	0.85	0.70	0.74	6789
weighted avg	0.87	0.88	0.86	6789

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```





Observaciones:

Contra lo que se hubiese esperado, no mejora demasiado la precisión al utilizar monogramas y bigramas.

Random Forest - Lemma - Bigramas

Random Forest

```
rf = RandomForestClassifier()
rf.fit(xtrain,ytrain)

y_pred = rf.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_s
```

```
Accuracy :  0.855796140816026
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
#k_validacion_cruzada(rf,X,recom,3)
```

```
Validación cruzada:
```

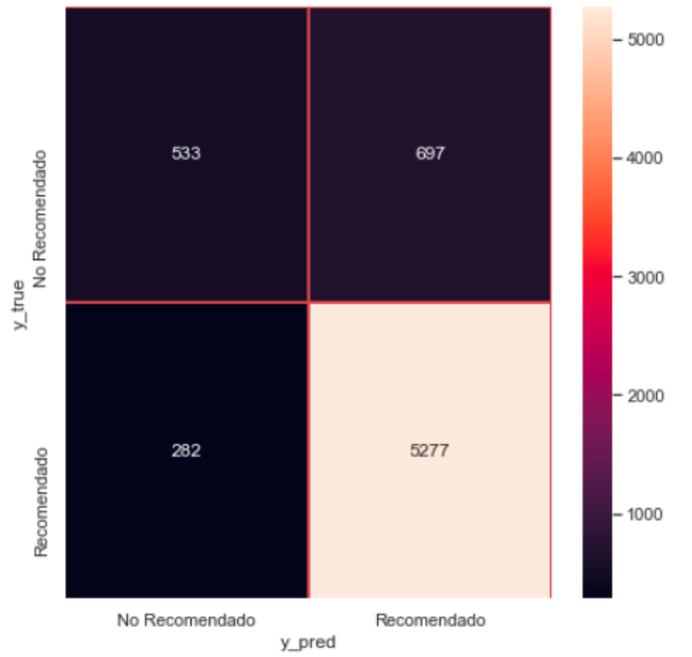
```
print(color.BOLD + 'Reporte de clasificación : '+ color.
```

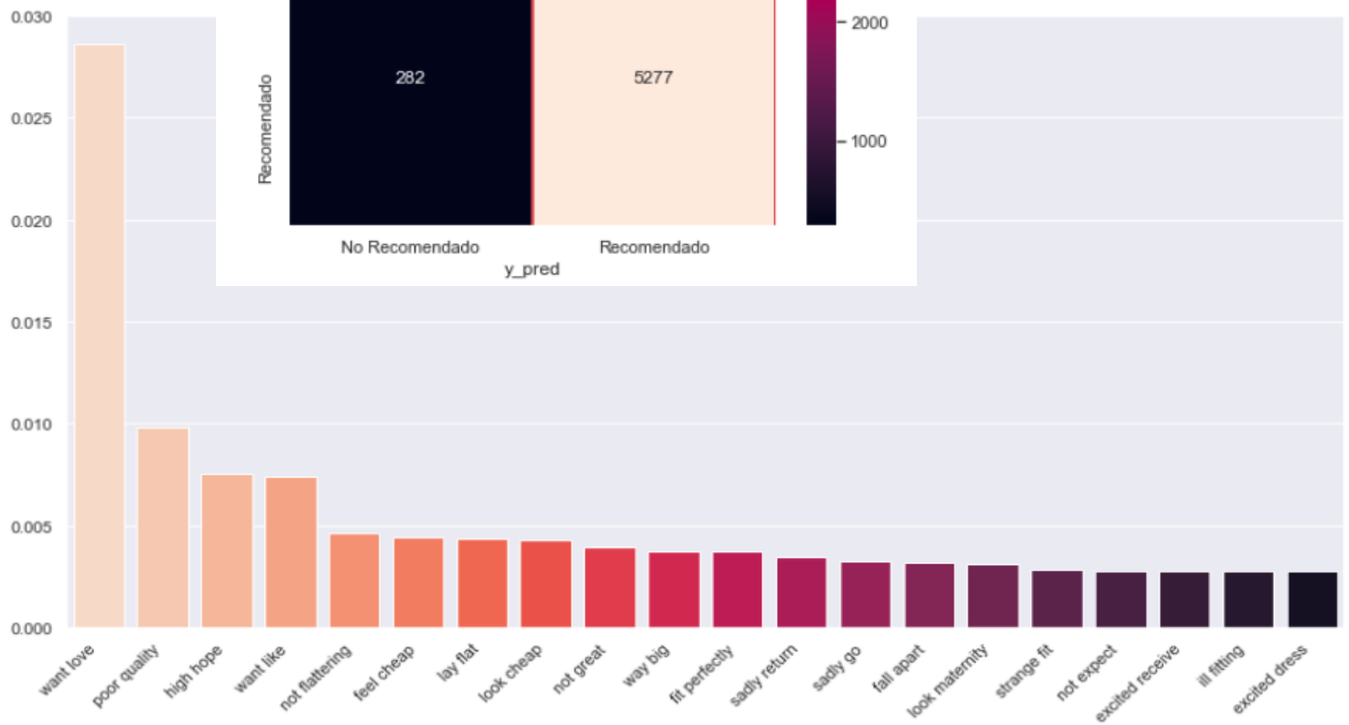
```
Reporte de clasificación :
```

	precision	recall	f1-score	support
No Rec	0.65	0.43	0.52	1230
Rec	0.88	0.95	0.92	5559
accuracy			0.86	6789
macro avg	0.77	0.69	0.72	6789
weighted avg	0.84	0.86	0.84	6789

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```





Observaciones:

Contra lo que se hubiese esperado, baja la precisión en las clases de los extremos, pero en contraparte aumenta en las clases intermedias.

Las *features* de mayor importancia tienen algunas palabras muy generales como ‘producto’, ‘comprar’ y ‘tener’ cuya relevancia puede estar dada por frecuencia pero estimo que no por semántica positiva o negativa. Podría probarse a futuro filtrarlas.

Random Forest – Stem – Monogramas

Random Forest

```
rf = RandomForestClassifier()
rf.fit(xtrain,ytrain)

y_pred = rf.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_
```

```
Accuracy :  0.8722934158197083
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(rf,X,recom,3)
```

```
Validación cruzada:
0.87 de precisión con desviación estándar de 0.00
```

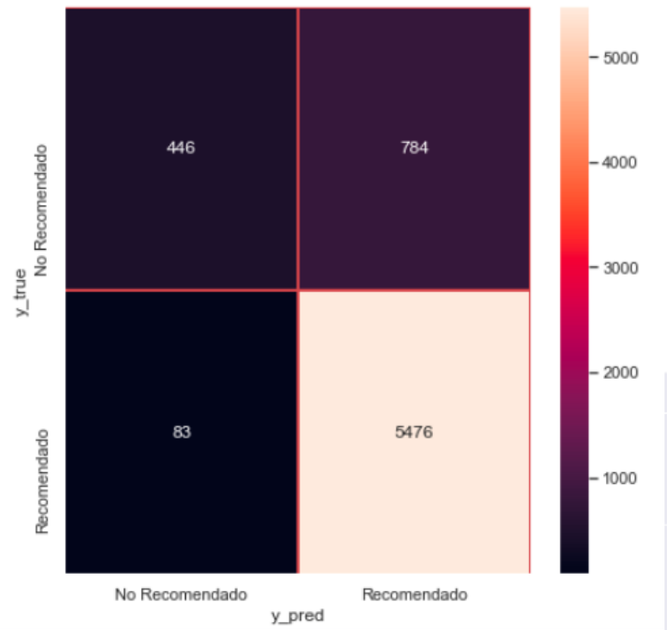
```
print(color.BOLD + 'Reporte de clasificación : '+ color
```

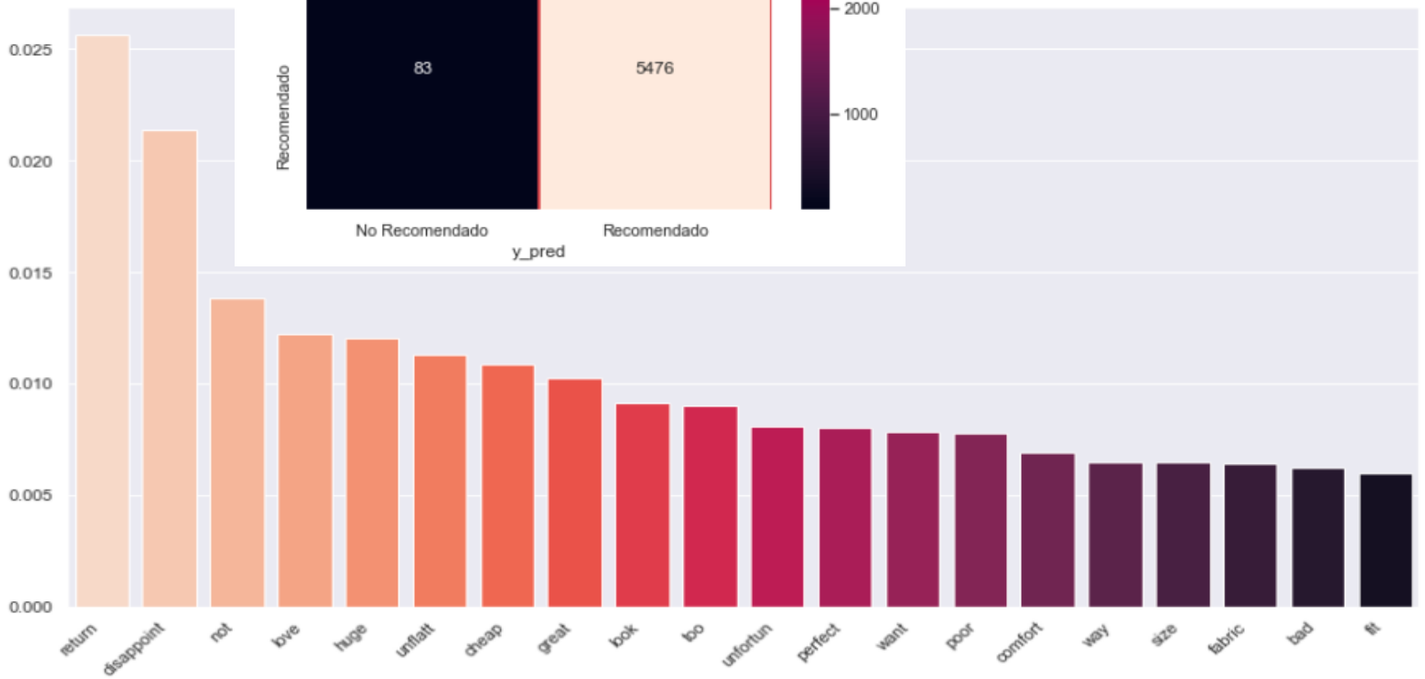
```
Reporte de clasificación :
```

	precision	recall	f1-score	support
No Rec	0.84	0.36	0.51	1230
Rec	0.87	0.99	0.93	5559
accuracy			0.87	6789
macro avg	0.86	0.67	0.72	6789
weighted avg	0.87	0.87	0.85	6789

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```





Observaciones:

Desempeño similar a su contraparte lemmatizada. Identifica muy bien a la clase “Recomendado” pero sigue confundiendo en más del %50 a la clase “No recomendada”.

Random Forest - Stem - Monogramas + Bigramas

Random Forest

```
rf = RandomForestClassifier()
rf.fit(xtrain,ytrain)

y_pred = rf.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_s
```

```
Accuracy :  0.8775961113566063
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(rf,X,recom,3)
```

```
Validación cruzada:
0.87 de precisión con desviación estándar de 0.00
```

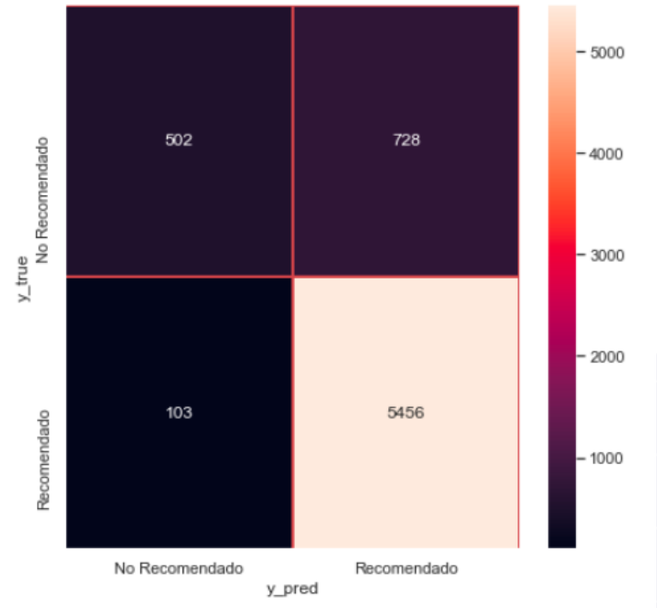
```
print(color.BOLD + 'Reporte de clasificación : '+ color.
```

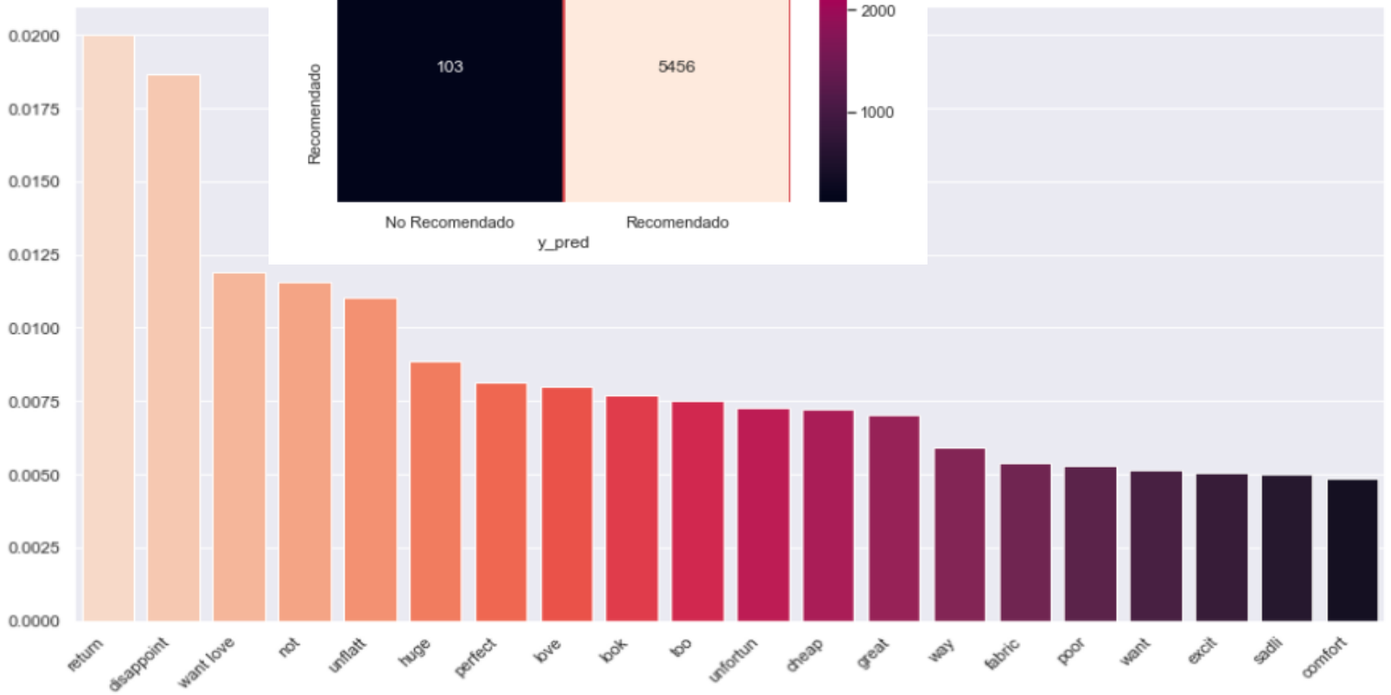
```
Reporte de clasificación :
```

	precision	recall	f1-score	support
No Rec	0.83	0.41	0.55	1230
Rec	0.88	0.98	0.93	5559
accuracy			0.88	6789
macro avg	0.86	0.69	0.74	6789
weighted avg	0.87	0.88	0.86	6789

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```





Observaciones:

Tiene un nivel de recall excelente para la clase recomendada, identificando a casi todos los ítems de dicha clase.

Sin embargo, sigue confundiendo a más de la mitad de los no recomendados.

Random Forest – Stem - Bigramas

Random Forest

```
rf = RandomForestClassifier()
rf.fit(xtrain,ytrain)

y_pred = rf.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_sc
```

```
Accuracy :  0.8596258653704522
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(rf,X,recom,3)
```

```
Validación cruzada:
```

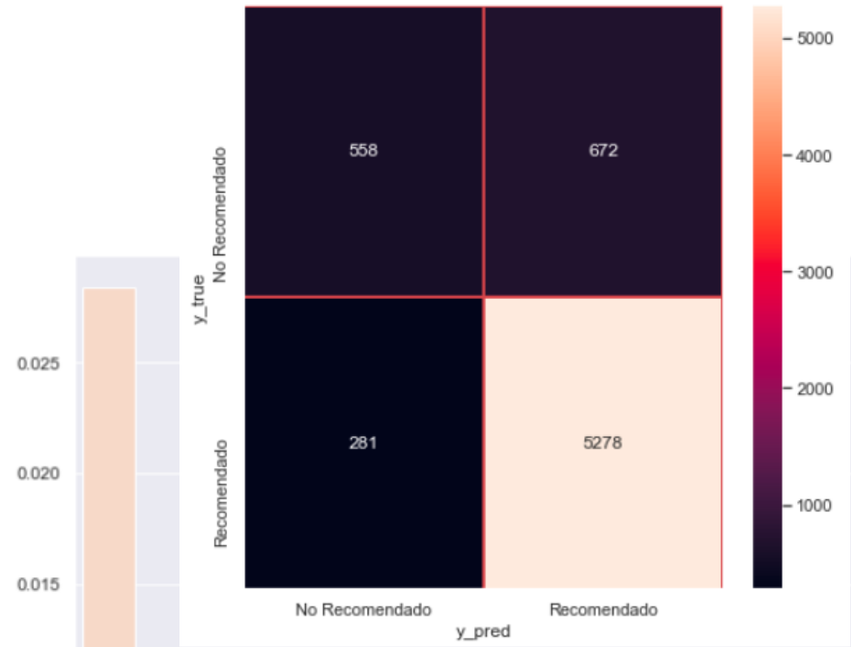
```
print(color.BOLD + 'Reporte de clasificación : '+ color.E
```

```
Reporte de clasificación :
```

	precision	recall	f1-score	support
No Rec	0.67	0.45	0.54	1230
Rec	0.89	0.95	0.92	5559
accuracy			0.86	6789
macro avg	0.78	0.70	0.73	6789
weighted avg	0.85	0.86	0.85	6789

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```



Observaciones:

Baja la precisión pero aumenta un poco el recall respecto del modelo anterior.

SVM - Lemma - Monogramas

SVM

```
# En vez de utilizar SVC, vamos a usar LinearSVC,
# ya que para el Kernel Lineal esta función es MUCHO ma

svc = LinearSVC(C = 1)
svc.fit(xtrain,ytrain)

y_pred = svc.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_
```

```
Accuracy :  0.9019001325673884
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(svc,X,recom,5)
```

```
Validación cruzada:
0.90 de precisión con desviación estándar de 0.00
```

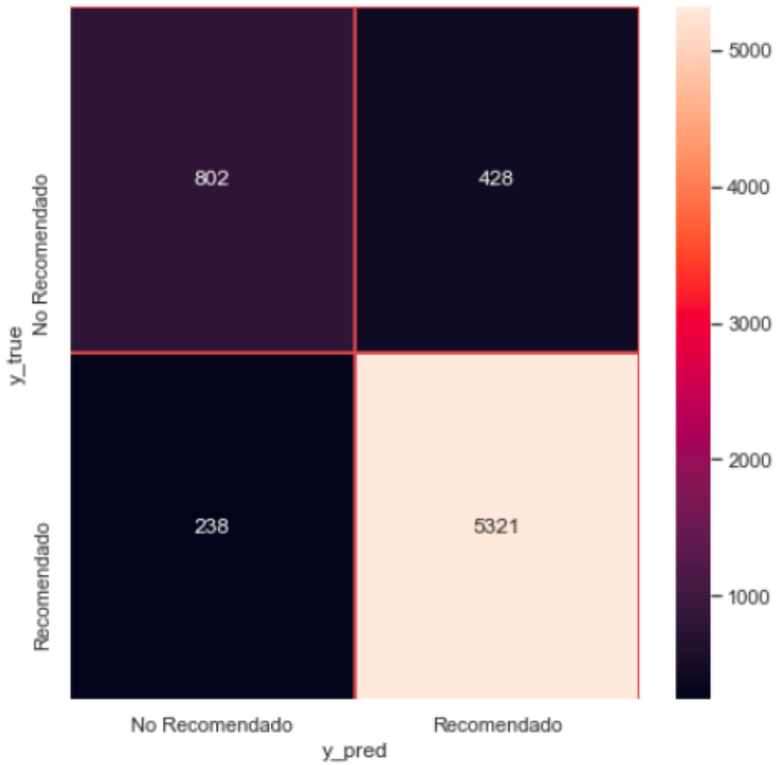
```
print(color.BOLD + 'Reporte de clasificación : '+ color
```

```
Reporte de clasificación :
```

	precision	recall	f1-score	support
No Rec	0.77	0.65	0.71	1230
Rec	0.93	0.96	0.94	5559
accuracy			0.90	6789
macro avg	0.85	0.80	0.82	6789
weighted avg	0.90	0.90	0.90	6789

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```



Observaciones:

Tiene una precisión increíblemente alta y trata de emparejar el recall. Aún así, todavía se podría mejorar.

SVM - Lemma - Monogramas + Bigramas

SVM

```
# En vez de utilizar SVC, vamos a usar LinearSVC,
# ya que para el Kernel Lineal esta función es MUCHO mas

svc = LinearSVC(C = 1)
svc.fit(xtrain,ytrain)

y_pred = svc.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_s
```

```
Accuracy :  0.8968920312269848
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(svc,X,recom,5)
```

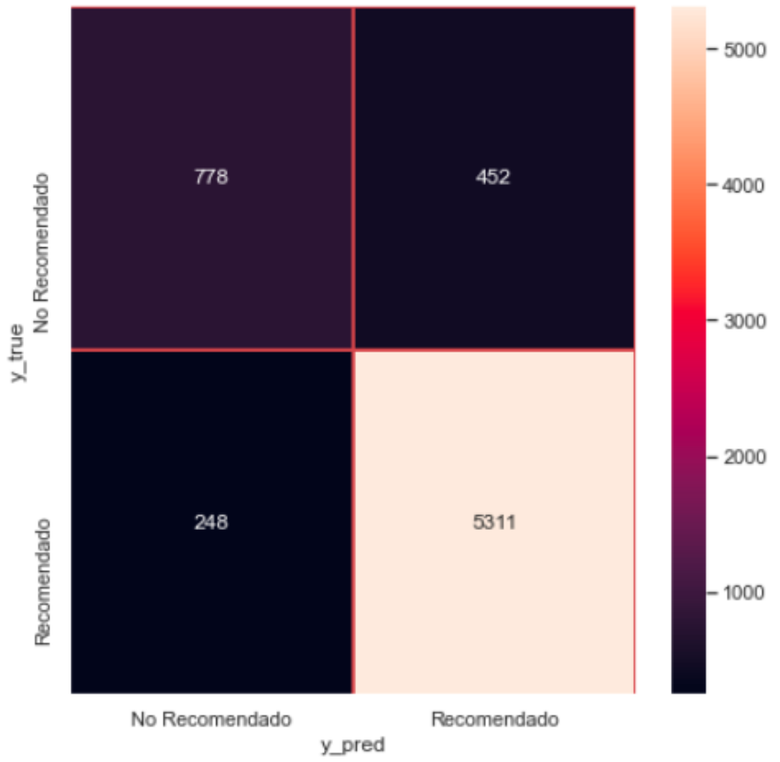
```
Validación cruzada:
0.90 de precisión con desviación estándar de 0.00
```

```
print(color.BOLD + 'Reporte de clasificación : '+ color.
```

```
Reporte de clasificación :

              precision    recall  f1-score   support

      No Rec       0.76      0.63      0.69      1230
         Rec       0.92      0.96      0.94      5559

    accuracy                           0.90      6789
   macro avg       0.84      0.79      0.81      6789
weighted avg       0.89      0.90      0.89      6789
```

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```



Observaciones:

Al utilizar tanto monogramas como bigramas juntos se observa que el desempeño disminuye levemente, aunque el score sigue siendo alto.

SVM - Lemma - Bigramas

SVM

```
# En vez de utilizar SVC, vamos a usar LinearSVC,
# ya que para el Kernel Lineal esta función es MUCHO mas

svc = LinearSVC(C = 1)
svc.fit(xtrain,ytrain)

y_pred = svc.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_s
```

```
Accuracy :  0.8587420827809692
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(svc,X,recom,5)
```

```
Validación cruzada:
0.86 de precisión con desviación estándar de 0.00
```

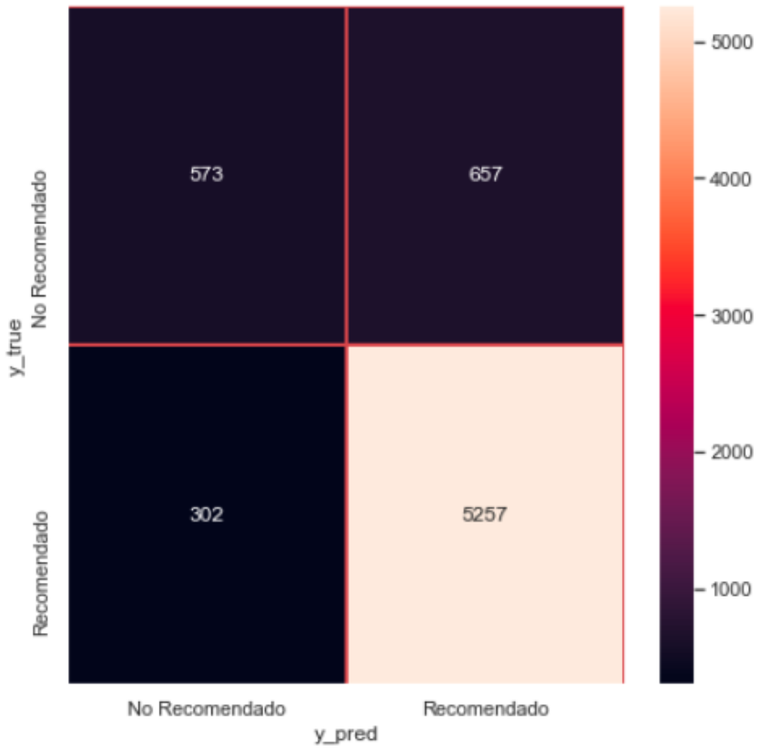
```
print(color.BOLD + 'Reporte de clasificación : '+ color.
```

```
Reporte de clasificación :
```

	precision	recall	f1-score	support
No Rec	0.65	0.47	0.54	1230
Rec	0.89	0.95	0.92	5559
accuracy			0.86	6789
macro avg	0.77	0.71	0.73	6789
weighted avg	0.85	0.86	0.85	6789

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```



Observaciones:

En general baja el rendimiento respecto de los modelos con monogramas.

SVM - Stem - Monogramas

SVM

```
# En vez de utilizar SVC, vamos a usar LinearSVC,
# ya que para el Kernel Lineal esta función es MUCHO mas

svc = LinearSVC(C = 1)
svc.fit(xtrain,ytrain)

y_pred = svc.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_sc
```

```
Accuracy :  0.8983650022094565
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(svc,X,recom,5)
```

```
Validación cruzada:
0.90 de precisión con desviación estándar de 0.00
```

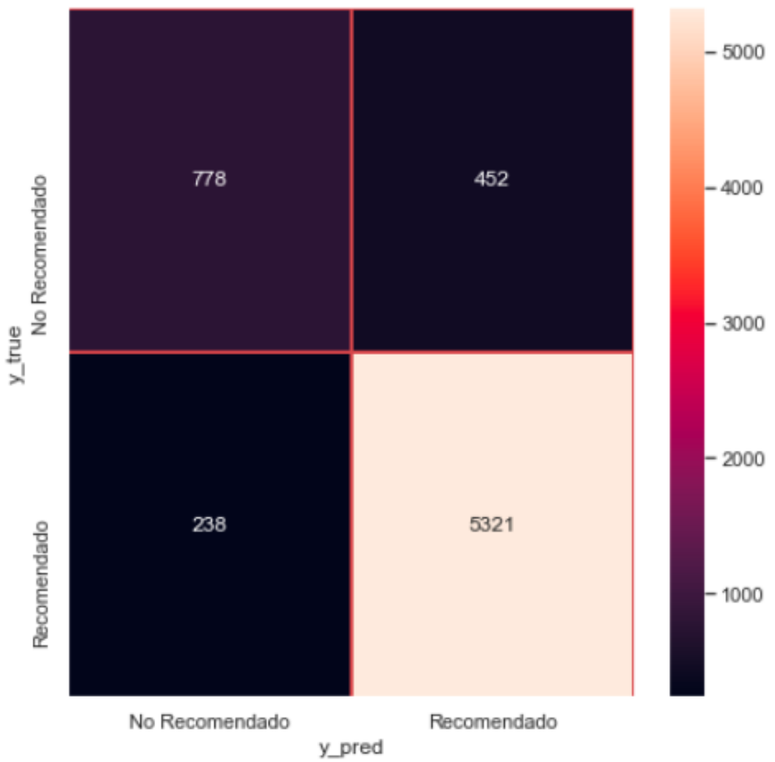
```
print(color.BOLD + 'Reporte de clasificación : '+ color.E
```

```
Reporte de clasificación :
```

	precision	recall	f1-score	support
No Rec	0.77	0.63	0.69	1230
Rec	0.92	0.96	0.94	5559
accuracy			0.90	6789
macro avg	0.84	0.79	0.82	6789
weighted avg	0.89	0.90	0.89	6789

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```



Observaciones:

El desempeño es levemente peor que su contraparte lemmatizada, pero aún así su score es alto. Acierta bien en la clase de "Recomendados", pero no tanto en los "No recomendados".

SVM - Stem - Monogramas + Bigramas

SVM

```
# En vez de utilizar SVC, vamos a usar LinearSVC,
# ya que para el Kernel Lineal esta función es MUCHO mas

svc = LinearSVC(C = 1)
svc.fit(xtrain,ytrain)

y_pred = svc.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_s
```

```
Accuracy :  0.9002798644866696
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(svc,X,recom,5)
```

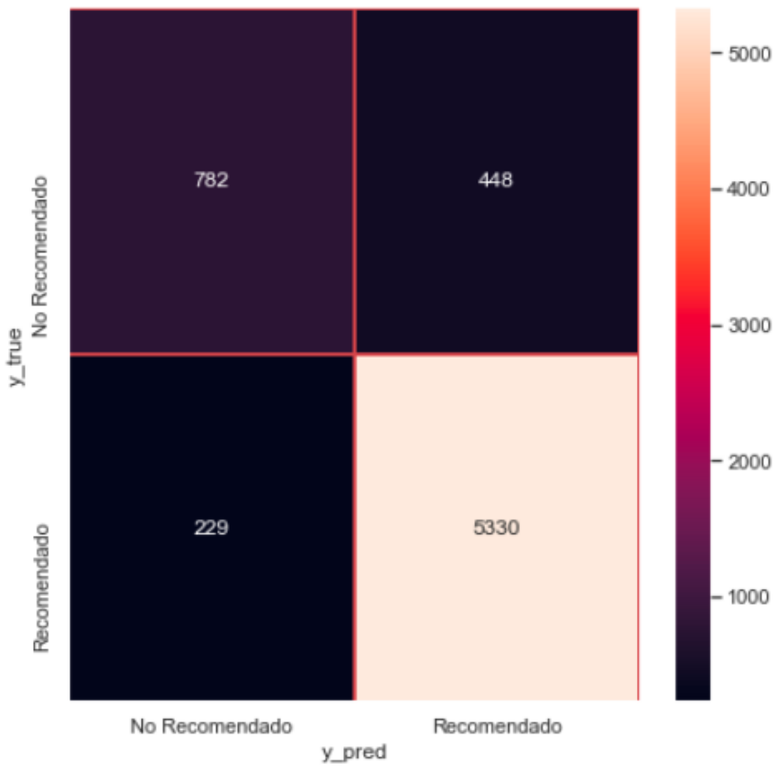
```
Validación cruzada:
0.90 de precisión con desviación estándar de 0.00
```

```
print(color.BOLD + 'Reporte de clasificación : '+ color.
```

```
Reporte de clasificación :

              precision    recall  f1-score   support

      No Rec       0.77      0.64      0.70      1230
         Rec       0.92      0.96      0.94      5559

    accuracy                           0.90      6789
   macro avg       0.85      0.80      0.82      6789
weighted avg       0.90      0.90      0.90      6789
```

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```



Observaciones:

Tiene un muy buen score para la clase recomendada y también es parejo con el score de las no recomendadas, ya que ambas métricas están por encima del 60%.

Este modelo es un buen candidato para ser elegido.

SVM – Stem - Bigramas

SVM

```
# En vez de utilizar SVC, vamos a usar LinearSVC,
# ya que para el Kernel Lineal esta función es MUCHO mas

svc = LinearSVC(C = 1)
svc.fit(xtrain,ytrain)

y_pred = svc.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END, accuracy_s
```

```
Accuracy :  0.8585947856827221
```

```
print(color.BOLD + 'Validación cruzada:' + color.END)
k_validacion_cruzada(svc,X,recom,5)
```

```
Validación cruzada:
0.86 de precisión con desviación estándar de 0.00
```

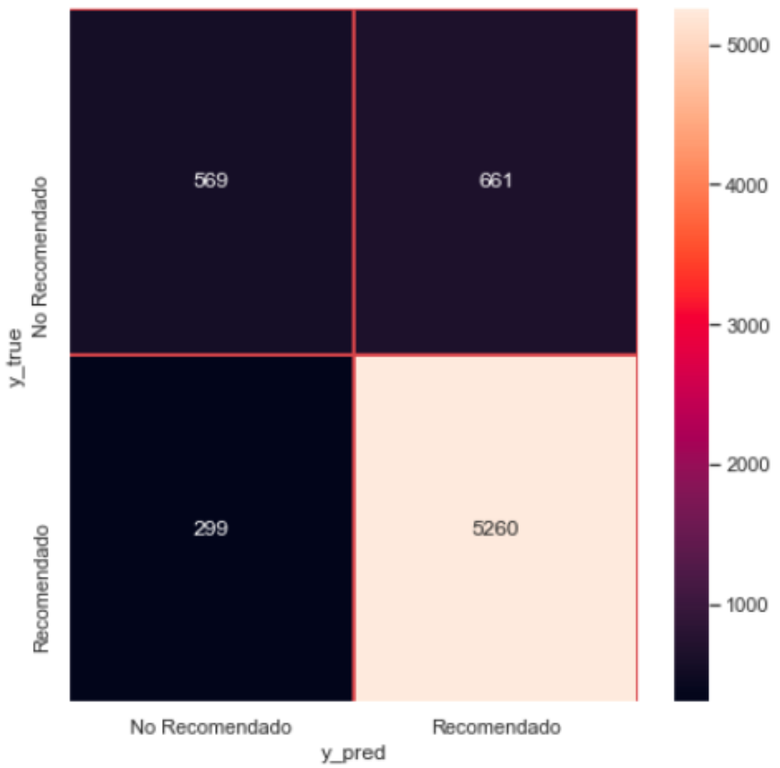
```
print(color.BOLD + 'Reporte de clasificación : '+ color.
```

```
Reporte de clasificación :
```

	precision	recall	f1-score	support
No Rec	0.66	0.46	0.54	1230
Rec	0.89	0.95	0.92	5559
accuracy			0.86	6789
macro avg	0.77	0.70	0.73	6789
weighted avg	0.85	0.86	0.85	6789

```
print(color.BOLD + 'Matriz de confusión : '+ color.END)
confusion(ytest,y_pred)
```

```
Matriz de confusión :
```



Observaciones:

Nuevamente baja el rendimiento respecto de modelos anteriores y empeora el recall de la clase no recomendada.