COMPARACIÓN DE MODELOS CON DATASET LEMMATIZADO Y STEMMIZADO

Lemma – Naive Bayes

Naive Bayes - Modelo Benchmark

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

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

```
Accuracy :  0.4091746031746032
```

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

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

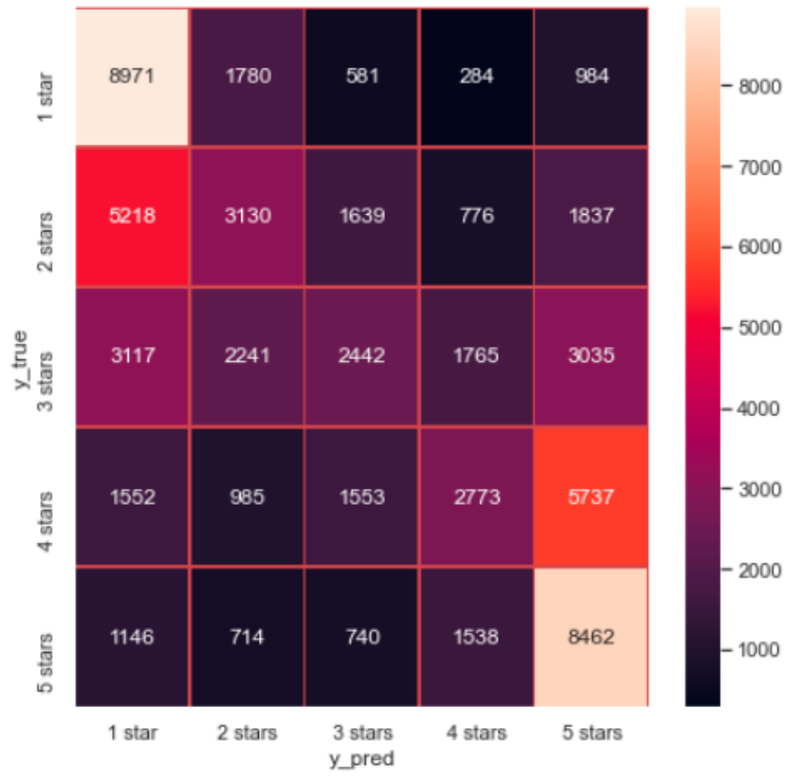
```
print(color.BOLD + 'Reporte de clasificación : '+ color.
```

```
Reporte de clasificación :
```

	precision	recall	f1-score	support
1	0.45	0.71	0.55	12600
2	0.35	0.25	0.29	12600
3	0.35	0.19	0.25	12600
4	0.39	0.22	0.28	12600
5	0.42	0.67	0.52	12600
accuracy			0.41	63000
macro avg	0.39	0.41	0.38	63000
weighted avg	0.39	0.41	0.38	63000

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

```
Matriz de confusión :
```



Observaciones:

Los mayores aciertos están en las de 1 y 5 estrellas. Aún así, existen bastantes de 5 estrellas que predijo como de 1. Esto podría deberse a que usan palabras parecidas, tanto para hablar positivamente como diciendo que "no" son así.

También vemos que hay gran confusión también entre los de 4 y 5 estrellas y los de 1 y 2 estrellas. Es entendible porque hay mucho parecido entre ellas.

Finalmente, el de 3 estrellas es el que más fácilmente se confunde con los demás ya que es la media y tiene palabras que se repiten en todas las reseñas.

COMPARACIÓN DE MODELOS CON DATASET LEMMATIZADO Y STEMMIZADO

Stem – Naive Bayes

Naive Bayes - Modelo Benchmark

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

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

```
Accuracy :  0.4146666666666667
```

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

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

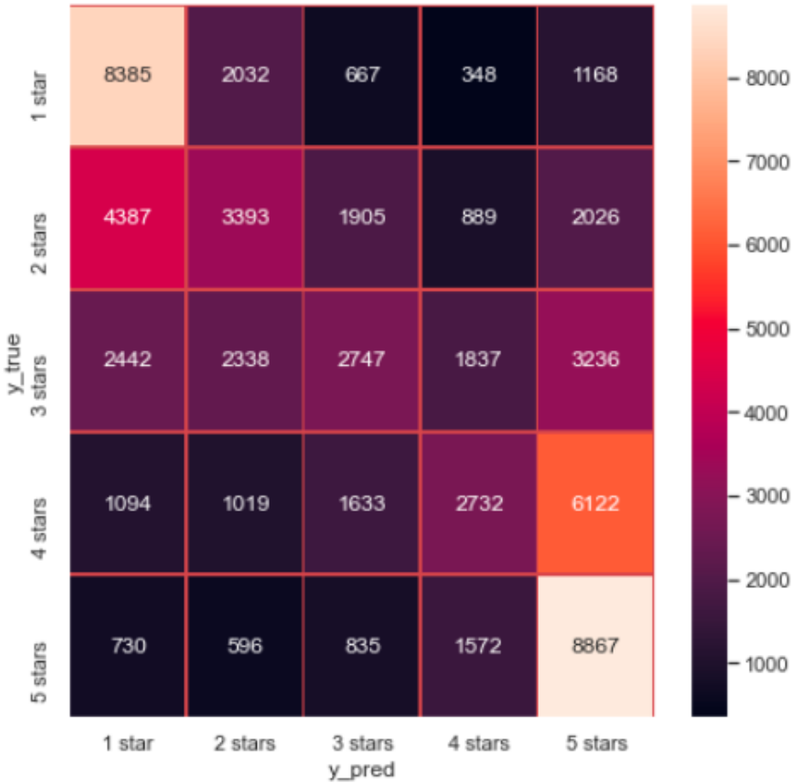
```
print(color.BOLD + 'Reporte de clasificación : '+ color.
```

```
Reporte de clasificación :
```

	precision	recall	f1-score	support
1	0.49	0.67	0.57	12600
2	0.36	0.27	0.31	12600
3	0.35	0.22	0.27	12600
4	0.37	0.22	0.27	12600
5	0.41	0.70	0.52	12600
accuracy			0.41	63000
macro avg	0.40	0.41	0.39	63000
weighted avg	0.40	0.41	0.39	63000

```
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.

Lemma – Random Forest

Random Forest

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

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

```
Accuracy :  0.42758730158730157
```

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

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

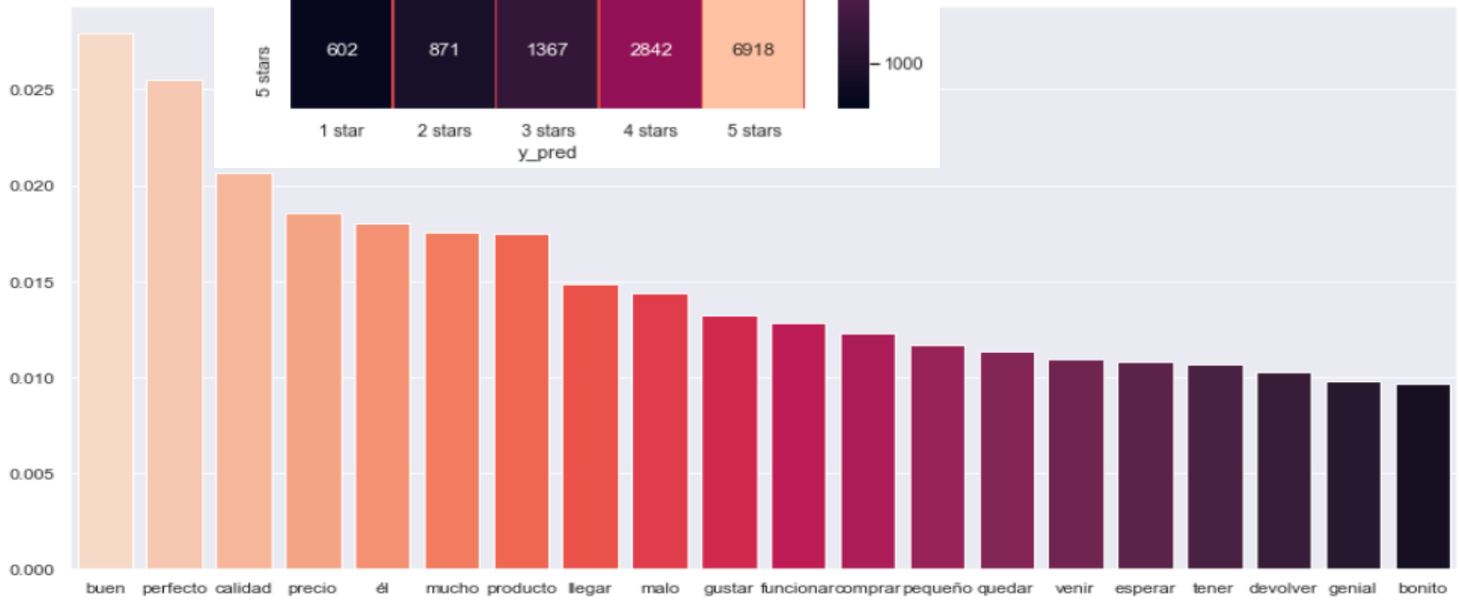
```
print(color.BOLD + 'Reporte de clasificación : '+ color
```

```
Reporte de clasificación :
```

	precision	recall	f1-score	support
1	0.53	0.61	0.56	12600
2	0.35	0.32	0.33	12600
3	0.33	0.32	0.32	12600
4	0.38	0.34	0.36	12600
5	0.51	0.55	0.53	12600
accuracy			0.43	63000
macro avg	0.42	0.43	0.42	63000
weighted avg	0.42	0.43	0.42	63000

```
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.

Stem – Random Forest

Random Forest

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

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

```
Accuracy :  0.44323809523809526
```

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

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

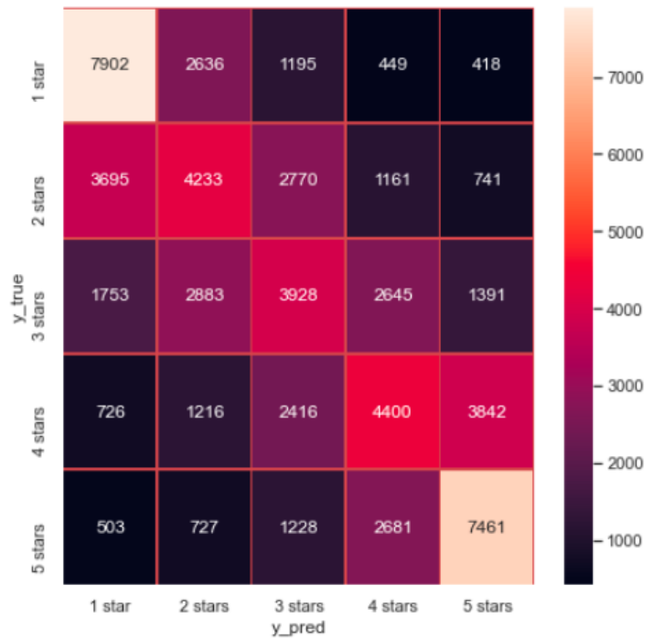
```
print(color.BOLD + 'Reporte de clasificación : '+ colo
```

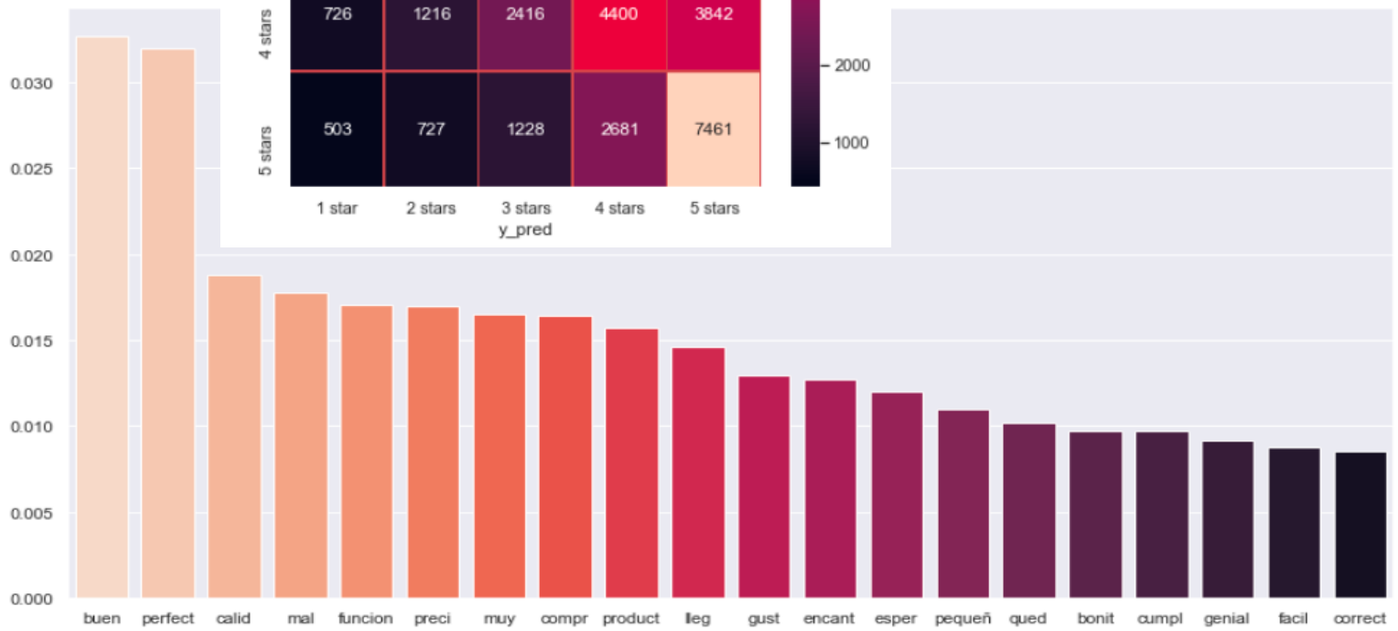
```
Reporte de clasificación :
```

	precision	recall	f1-score	support
1	0.54	0.63	0.58	12600
2	0.36	0.34	0.35	12600
3	0.34	0.31	0.33	12600
4	0.39	0.35	0.37	12600
5	0.54	0.59	0.56	12600
accuracy			0.44	63000
macro avg	0.43	0.44	0.44	63000
weighted avg	0.43	0.44	0.44	63000

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

```
Matriz de confusión :
```





Observaciones:

Mucho mejor que su contraparte lemmatizado, ya que concentra las confusiones entre las clases adyacentes. Ya no hay errores muy marcados.

Las *feature importances* también son más adecuadas a lo que pretendemos.

Lemma - SVM

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 :  0.44536507936507935
```

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

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

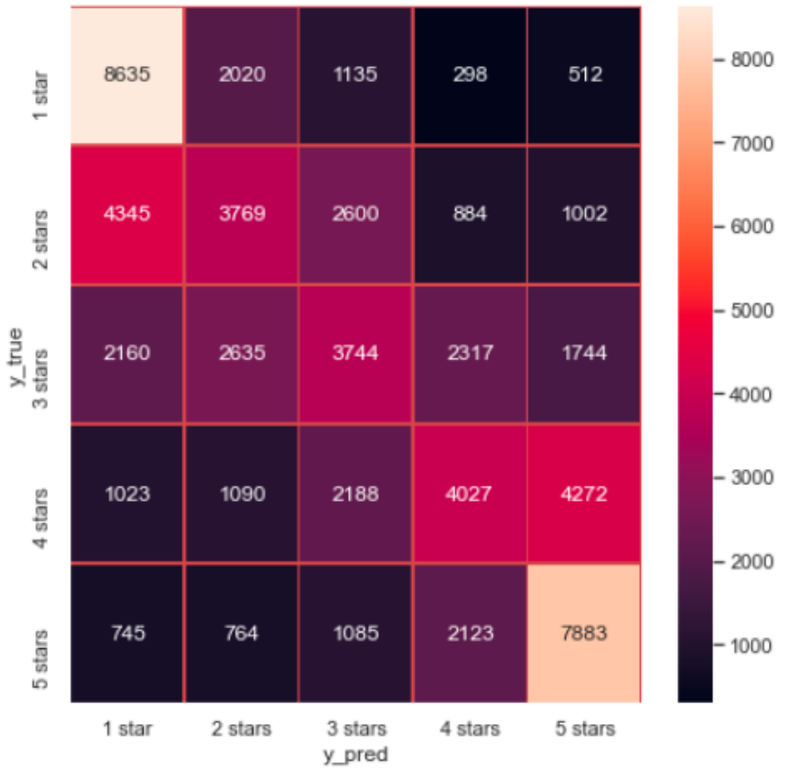
```
print(color.BOLD + 'Reporte de clasificación : '+ color
```

```
Reporte de clasificación :
```

	precision	recall	f1-score	support
1	0.51	0.69	0.59	12600
2	0.37	0.30	0.33	12600
3	0.35	0.30	0.32	12600
4	0.42	0.32	0.36	12600
5	0.51	0.63	0.56	12600
accuracy			0.45	63000
macro avg	0.43	0.45	0.43	63000
weighted avg	0.43	0.45	0.43	63000

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

```
Matriz de confusión :
```



Observaciones:

Si bien mejora considerablemente la predicción en los extremos, empeora respecto del *random forest* en las predicciones de la clase de 3 estrellas.

En los casos de las falsas predicciones de 1 y 5 para reviews que son de 3 puede suponerse que es porque son ambiguas y no logra identificar a cuál extremo pertenecen, por lo que les asigna un punto medio.

COMPARACIÓN DE MODELOS CON DATASET LEMMATIZADO Y STEMMIZADO

Stem - SVM

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 :  0.4607777777777778
```

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

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

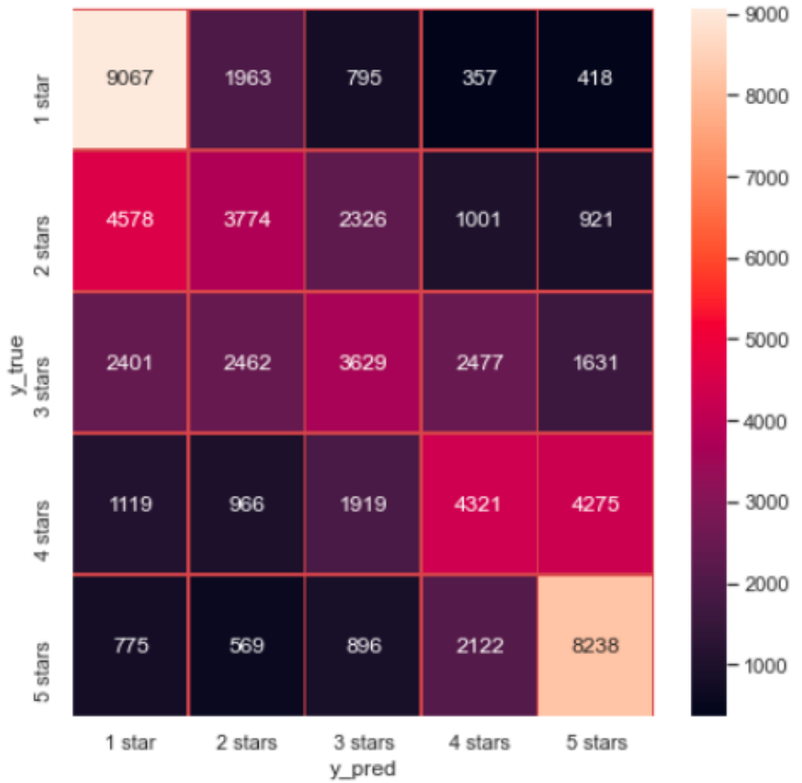
```
print(color.BOLD + 'Reporte de clasificación : '+ color.E
```

```
Reporte de clasificación :
```

	precision	recall	f1-score	support
1	0.52	0.71	0.60	12600
2	0.39	0.32	0.35	12600
3	0.37	0.28	0.32	12600
4	0.43	0.33	0.38	12600
5	0.52	0.66	0.58	12600
accuracy			0.46	63000
macro avg	0.44	0.46	0.45	63000
weighted avg	0.44	0.46	0.45	63000

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

```
Matriz de confusión :
```



Observaciones:

Es el modelo que hasta ahora dio el mejor score con 0.46 de accuracy.

Las confusiones más significativas están entre el 4 y el 5 y el 2 y el 1, lo cual es razonable.

También en el reporte genérico es el que más precisión tiene, aunque no así con el recall.