GRID SEARCH RESULTS

Linear SVC trained with: *C = [1, 1.5, 2, 2.5, 3]*

```
print("Mejores parametros: "+str(model.best_params_))
print("Mejor Score: "+str(model.best_score_)+'\n')

scores = pd.DataFrame(model.cv_results_)
scores
```

```
Mejores parametros: {'C': 1}
Mejor Score: 0.4943129251700681
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_C	params	split0_test_score	split1_test_score	split2_test_sc
0	9.059723	0.624642	0.192437	0.142013	1	{'C': 1}	0.495034	0.495374	0.493
1	11.224115	0.223814	0.092886	0.007992	1.5	{'C': 1.5}	0.495204	0.495340	0.492
2	14.356893	0.491114	0.100254	0.004118	2	{'C': 2}	0.495000	0.495238	0.492
3	16.536136	0.438183	0.102634	0.007650	2.5	{'C': 2.5}	0.494898	0.495068	0.492
4	20.136198	1.249134	0.100601	0.012904	3	{'C': 3}	0.494898	0.495102	0.492

m_C	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score
1	{'C': 1}	0.495034	0.495374	0.493027	0.492619	0.495510	0.494313	0.001233
1.5	{'C': 1.5}	0.495204	0.495340	0.492857	0.492585	0.495510	0.494299	0.001295
2	{'C': 2}	0.495000	0.495238	0.492721	0.492551	0.495374	0.494177	0.001265
2.5	{'C': 2.5}	0.494898	0.495068	0.492823	0.492619	0.495374	0.494156	0.001184
3	{'C': 3}	0.494898	0.495102	0.492925	0.492619	0.495306	0.494170	0.001153

Como el mejor resultado fue el menor valor, investigamos valores aún más pequeños de C.

Linear SVC trained with: *C = [0.01, 0.1, 1, 1.1]*

```
print("Mejores parametros: "+str(model.best_params_))
print("Mejor Score: "+str(model.best_score_)+'\n')

scores = pd.DataFrame(model.cv_results_)
scores
```

```
Mejores parametros: {'C': 0.01}
Mejor Score: 0.49779591836734693
```

n_C	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score
0.01	{'C': 0.01}	0.499660	0.498367	0.497347	0.494082	0.499524	0.497796	0.002039
0.10	{'C': 0.1}	0.497279	0.498265	0.496293	0.493980	0.498095	0.496782	0.001566
1	{'C': 1}	0.494966	0.496429	0.493776	0.493061	0.496259	0.494898	0.001329
1.10	{'C': 1.1}	0.495034	0.496497	0.493776	0.493061	0.496259	0.494925	0.001346

GRID SEARCH RESULTS

Evaluación del modelo optimizado

Los datos se vectorizan con el tfidf con un `mìn_df = 0.001` y `max_features = 15000`.
Luego se entrena un modelo `LinearSVC` con `C = 0.1`.

```
svcop = LinearSVC(C=0.01)
svcop.fit(xtrain,ytrain)

y_pred = svcop.predict(xtest)
print(color.BOLD + 'Accuracy : ' + color.END + color.YELLOW, accuracy_score(ytes
```

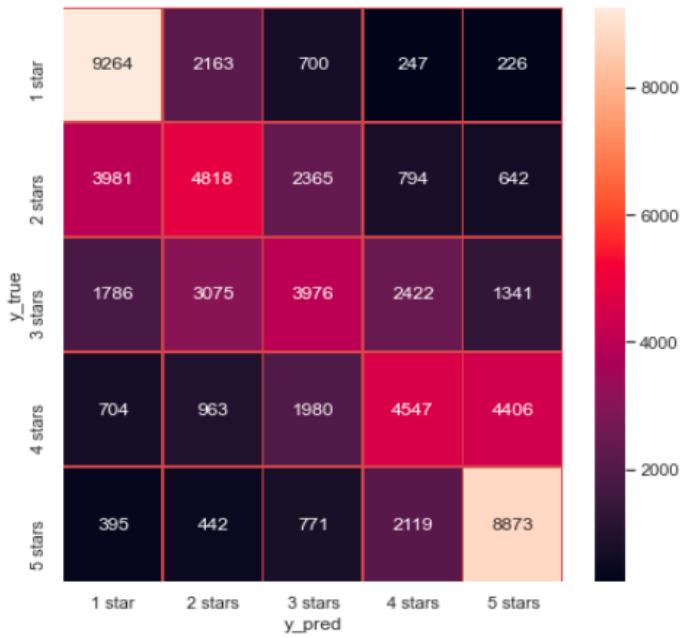
```
Accuracy :  0.4996507936507936
```

```
print(color.BOLD + 'Reporte de clasificación : '+ color.END + '\n \n' ,classifi
```

```
Reporte de clasificación :

              precision    recall  f1-score   support

           1       0.57      0.74      0.64     12600
           2       0.42      0.38      0.40     12600
           3       0.41      0.32      0.36     12600
           4       0.45      0.36      0.40     12600
           5       0.57      0.70      0.63     12600

    accuracy                           0.50     63000
   macro avg       0.48      0.50      0.49     63000
weighted avg       0.48      0.50      0.49     63000
```

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

```
Matriz de confusión :
```



Observaciones

Lo mejor que se pudo obtener con los hiperparámetros óptimos establecidos es una precisión del 50%.

Si bien este valor es relativamente bajo, está al límite del rendimiento aceptable, ya que acertará en la mitad de las predicciones de cada clase.

Como vemos en el informe y en la matriz de confusión, los más fáciles de identificar son las clases extremos y los más conflictivos son el 2 y el 4, ya que se los confunde en gran cantidad de casos con el 1 y el 5, respectivamente.

En cuanto a la clase 3, tiene el accuracy más bajo pero es el mejor rendimiento comparado con los otros modelos en esa clase.