Preprocesamiento y Clasificación para la Predicción de Vacunación contra la Gripe

Aplicación de Modelos de Machine Learning en la Gripe H1N1 y Estacional

Nombre del equipo:
Preprocesadores

Integrantes y Modelos:

Pablo Gradolph Oliva
- Árbol y Gradient Boosting

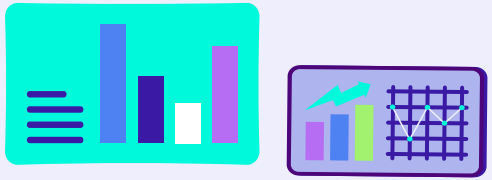
Enric Morella Violeta
- Naïve Bayes y Voting

Jaime De Castro Escribano
- SVM y Bagging

María Ruxandra Cojocaru
- Regresión logística y AdaBoost

Adrián Sánchez Carrión
- KNN y Stacking

Índice



1. Descripción del problema

Descripción del problema

Predecir la probabilidad de que una persona reciba dos tipos de vacunas: contra la gripe H1N1 y contra la gripe estacional.

Objetivo: construir modelos de predicción que puedan identificar patrones en esta información y ayudar a estimar las probabilidades de vacunación.

Comprender los factores de vacunación puede guiar la salud pública.

Competición y métricas de evaluación

Se trata de una competición organizada por DRIVENDATA, que se plantea como un problema de clasificación multilabel.

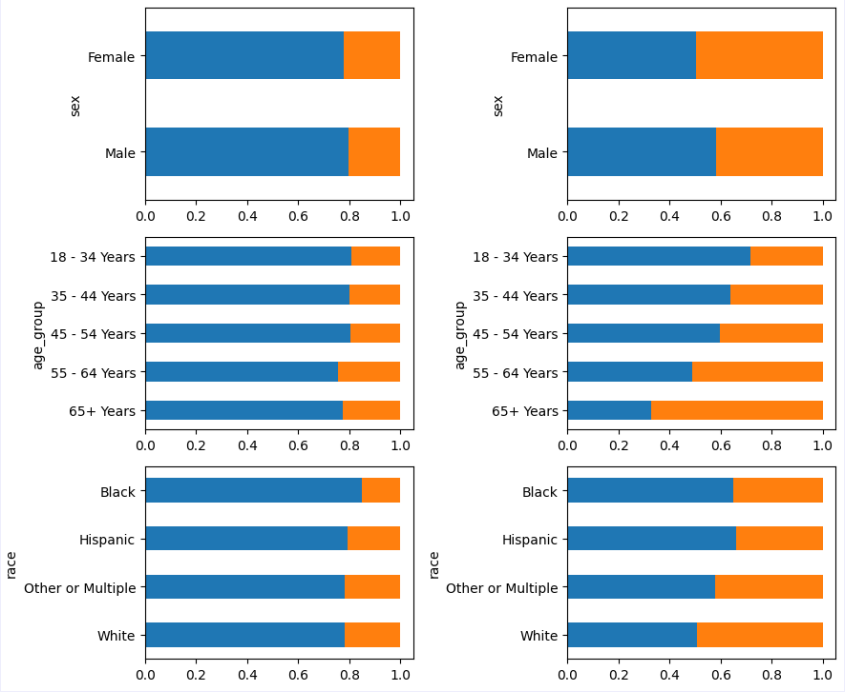
La evaluación se basa principalmente en la métrica del área bajo la curva ROC, o ROC-AUC.

También evaluamos el desempeño de los modelos utilizando otras métricas como accuracy.

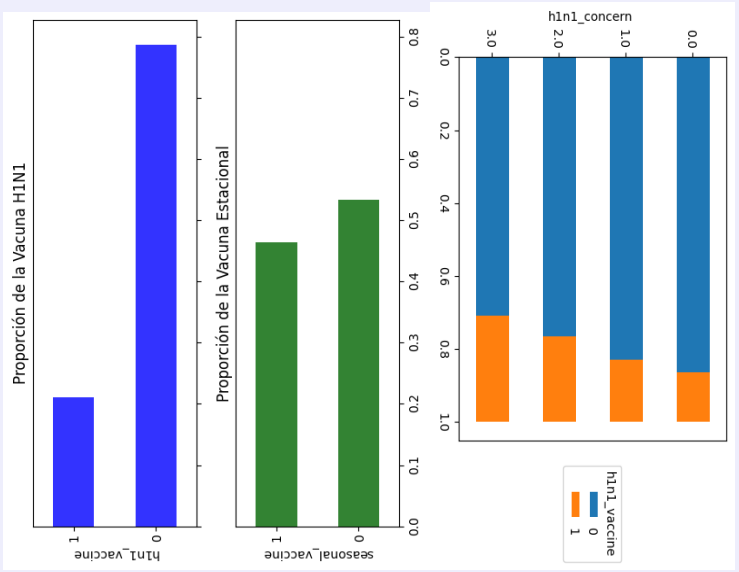
2. Exploratory Data Analysis

Análisis de las variables (características y etiquetas)

Dimensiones de las características de entrenamiento: (26707, 35)
Dimensiones de las etiquetas de entrenamiento: (26707, 2)
Dimensiones de las características de prueba: (26708, 35)



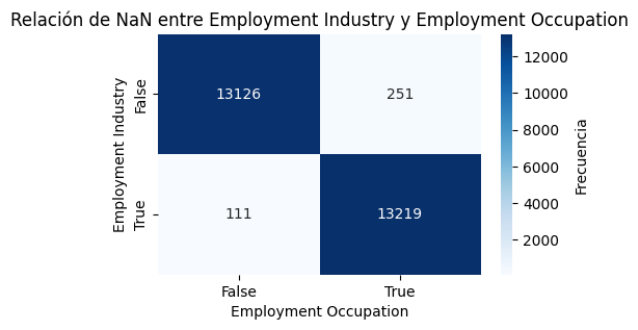
Coeficiente phi = 0.37, lo que indica una correlación positiva moderada. Esto también se puede ver en la tabulación cruzada. La mayoría de las personas que se vacunaron contra la gripe H1N1 también se vacunaron contra la gripe estacional. Si bien una minoría de las personas que se vacunaron contra la gripe estacional se vacunaron contra la gripe H1N1.



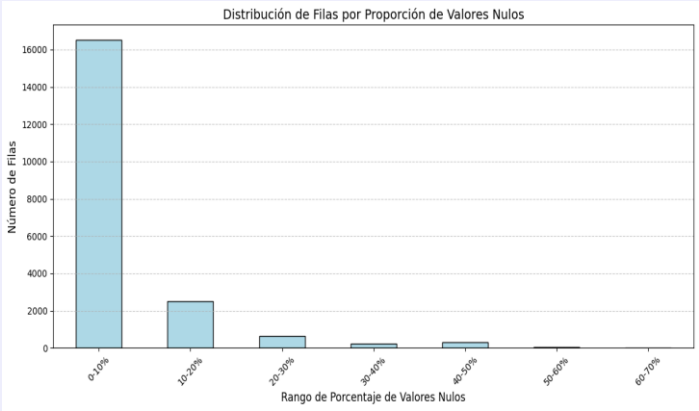
Detección de Outliers y Valores Perdidos

Se analizaron las variables individualmente, en busca de posibles errores en la anotación. Todos los valores observados para las distintas cuestiones estaban indexados en el rango permitido, por lo que concluimos con la detección de **valores anómalos**.

El **análisis de valores nulos** se realizó tanto por columnas como por registros, revelando información de gran utilidad para el preprocesamiento. Los análisis realizados por columnas se basaron en entender mejor la aparición de valores nulos en una proporción desmesurada para las columnas *health_insurance, employment_industry y employment occupation*.



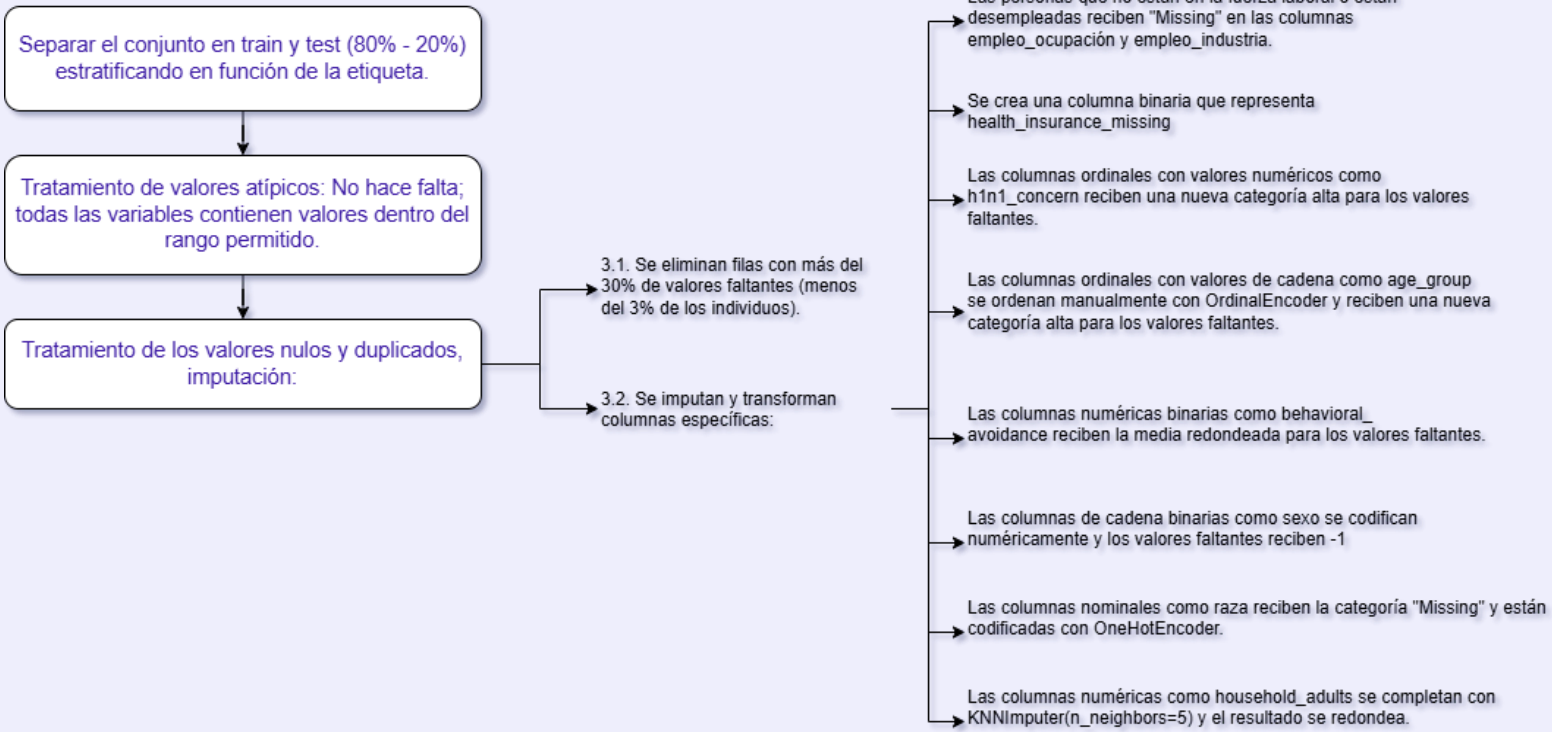
Análisis de falta de respuesta:
Se observó que 43 observaciones presentan una tasa de no respuesta superior al 50%. Alrededor del 2.5% de individuos no respondieron, al menos, al 30% del cuestionario. Inicialmente el 75% de los encuestados presentan falta de respuesta en alguna cuestión



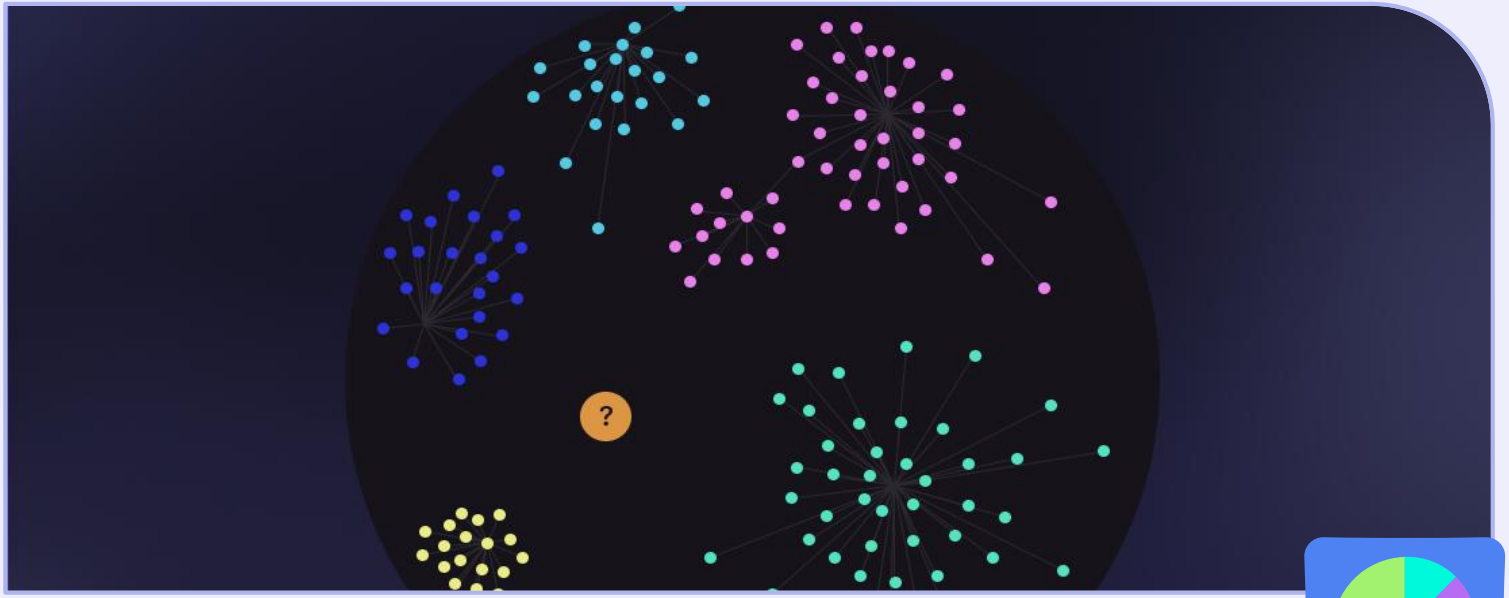
CUIDADO: Los resultados pueden estar demasiado sesgados para imputarle a algunas filas más del 50 % de sus resultados

Hipótesis: La causa puede ser el desempleo o similares

3. Preprocesamiento general


Separar el conjunto en train y test (80% - 20%) estratificando en función de la etiqueta.
Tratamiento de valores atípicos: No hace falta; todas las variables contienen valores dentro del rango permitido.
Tratamiento de los valores nulos y duplicados, imputación:
3.1. Se eliminan filas con más del 30% de valores faltantes (menos del 3% de los individuos).
3.2. Se imputan y transforman columnas específicas:
Las personas que no están en la fuerza laboral o están desempleadas reciben "Missing" en las columnas empleo_ocupación y empleo_industria.
Se crea una columna binaria que representa health_insurance_missing
Las columnas ordinales con valores numéricos como h1n1_concern reciben una nueva categoría alta para los valores faltantes.
Las columnas ordinales con valores de cadena como age_group se ordenan manualmente con OrdinalEncoder y reciben una nueva categoría alta para los valores faltantes.
Las columnas numéricas binarias como behavioral_ avoidance reciben la media redondeada para los valores faltantes.
Las columnas de cadena binarias como sexo se codifican numéricamente y los valores faltantes reciben -1
Las columnas nominales como raza reciben la categoría "Missing" y están codificadas con OneHotEncoder.
Las columnas numéricas como household_adults se completan con KNNImputer(n_neighbors=5) y el resultado se redondea.

4. Algoritmos



4.1. KNN

Preprocesamiento

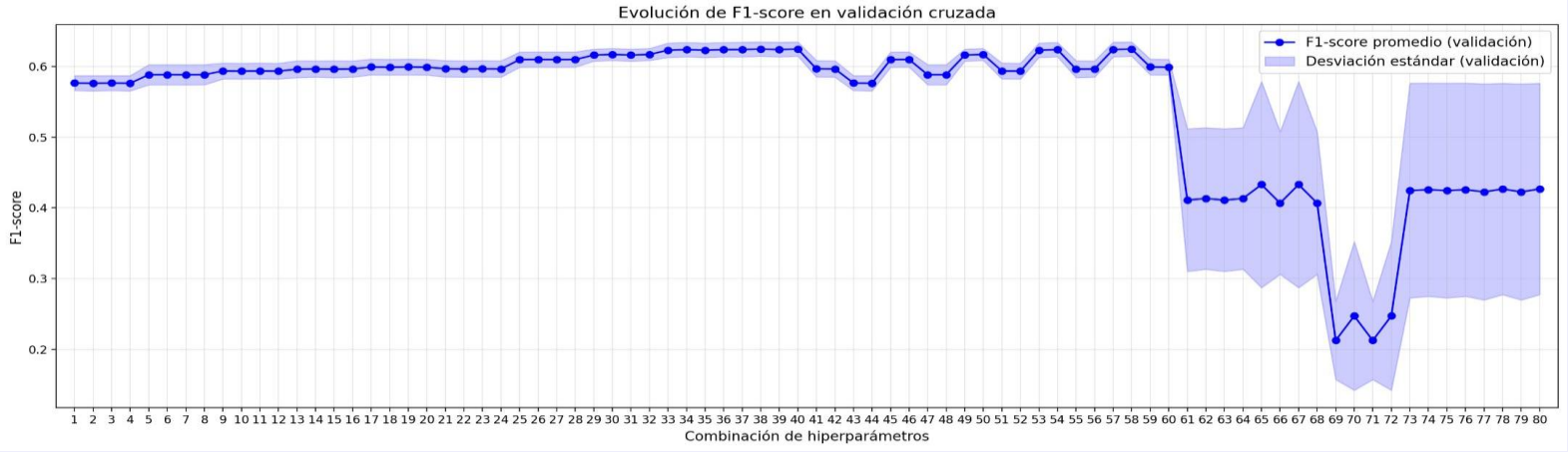
Para aplicar el algoritmo kNN, se han seguido los mismos pasos de **preprocesamiento** descritos en la **parte común.**

Además, **se han escalado los datos usando MinMaxScaler**, dado que este algoritmo se basa en distancias, y las características con rangos más grandes pueden dominar las más pequeñas, distorsionando los resultados.

División de los datos

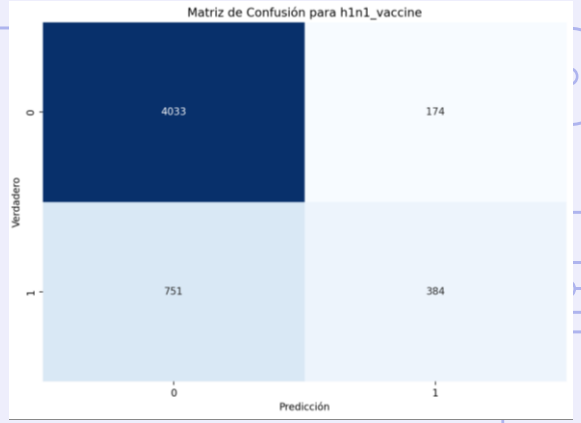
```
Training Features shape: (21365, 96)
Training Labels shape: (21365, 2)
Validation Features shape: (5342, 96)
Validation Labels shape: (5342, 2)
Test Features shape: (26708, 96)
```

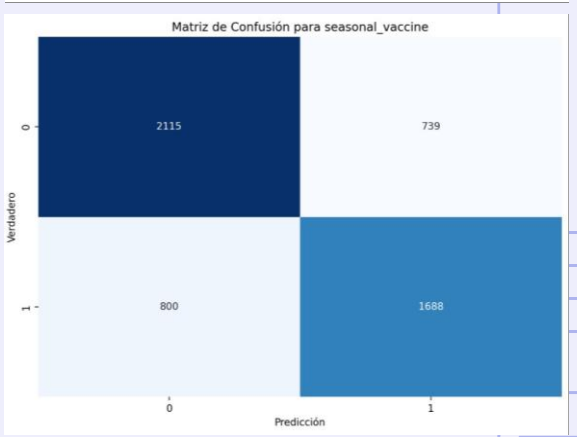
Definimos un clasificador K-Nearest Neighbors y lo envolvemos en un `MultiOutputClassifier` para manejar múltiples salidas. Luego, definimos una cuadrícula de parámetros (`param_grid`) que incluye diferentes métricas de distancia, números de vecinos y tipos de pesos. Después, creamos un objeto `GridSearchCV` para realizar una búsqueda en la cuadrícula de parámetros con validación cruzada de 10 pliegues (`cv=10`). Además, empleamos un scorer personalizado (`average='weighted'`) para configurar cómo se evalúa el desempeño del modelo durante el proceso de búsqueda de hiperparámetros.



```
Accuracy en validación: 0.6023961063272183

F1 Score (weighted): 0.6138069975638539
Precision (weighted): 0.693210408279402
Recall (weighted): 0.5719017388904223

Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.69      0.34      0.45      1135
           1       0.70      0.68      0.69      2488

   micro avg       0.69      0.57      0.63      3623
   macro avg       0.69      0.51      0.57      3623
weighted avg       0.69      0.57      0.61      3623
 samples avg       0.31      0.29      0.29      3623
```

```
Reporte de clasificación para h1n1_vaccine:
              precision    recall  f1-score   support

           0       0.84      0.96      0.90      4207
           1       0.69      0.34      0.45      1135

    accuracy                           0.83      5342
   macro avg       0.77      0.65      0.68      5342
weighted avg       0.81      0.83      0.80      5342

Reporte de clasificación para seasonal_vaccine:
              precision    recall  f1-score   support

           0       0.73      0.74      0.73      2854
           1       0.70      0.68      0.69      2488

    accuracy                           0.71      5342
   macro avg       0.71      0.71      0.71      5342
weighted avg       0.71      0.71      0.71      5342
```





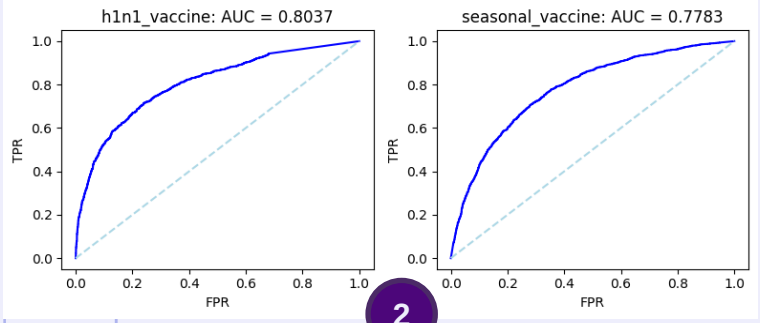
Mejores hiperparámetros

```
'estimator__metric':'manhattan','estimator__n_neighbors':11,
'estimator__p':1, 'estimator__weights':'distance'
```

	Metric	Neighbors	Weights	Minkowski_p	Mean F1-Score	Std F1-Score
57	minkowski	11	distance	1	0.624599	0.010024
37	manhattan	11	distance	1	0.624599	0.010024
39	manhattan	11	distance	2	0.624599	0.010024
36	manhattan	11	uniform	1	0.623943	0.010412
38	manhattan	11	uniform	2	0.623943	0.010412
56	minkowski	11	uniform	1	0.623943	0.010412
35	manhattan	9	distance	2	0.623815	0.010098
33	manhattan	9	distance	1	0.623815	0.010098
53	minkowski	9	distance	1	0.623815	0.010098
52	minkowski	9	uniform	1	0.622974	0.010116

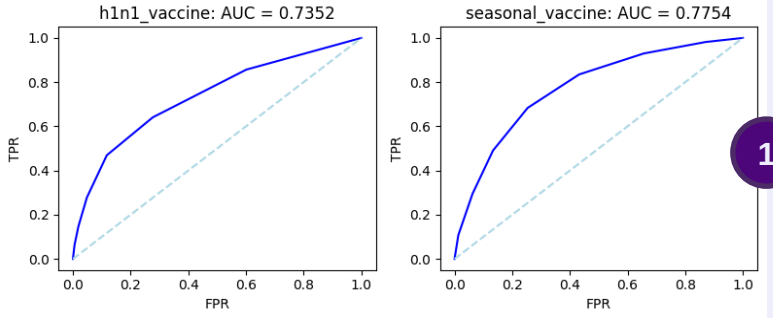
Predicción de probabilidades

```
AUROC para H1N1: 0.80367742
AUROC para vacuna estacional: 0.77834460
AUROC promedio: 0.79101101
```



2

```
'estimator__metric':'manhattan',
'estimator__n_neighbors':11,
'estimator__p':1,
'estimator__weights':'distance'
```

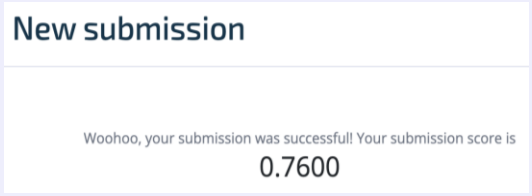


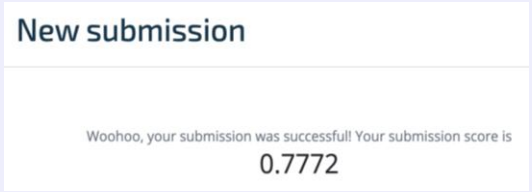
1

```
'estimator__metric': 'manhattan',
'estimator__n_neighbors': 7,
'estimator__weights': 'uniform'
```

Archivo de entrega en la competición

```
Dimensiones del archivo de entrega: (26708, 2)
```





respondent_id	h1n1_vaccine	seasonal_vaccine
26707	0.083824	0.543200
26708	0.091159	0.360170
26709	0.000000	0.395527
26710	0.732223	0.644614
26711	0.225263	0.409081

respondent_id	h1n1_vaccine	seasonal_vaccine
26707	0	1
26708	0	0
26709	0	0
26710	1	1
26711	0	0

0.7419  adrichez id-278405

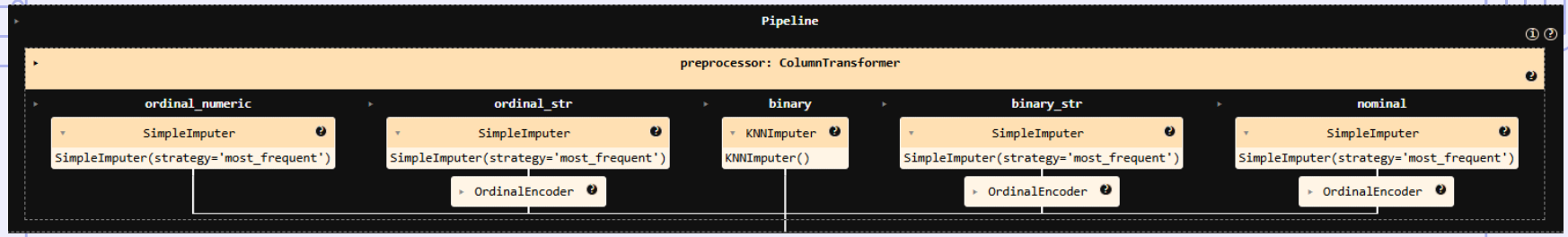
4.2. Naïve Bayes

Búsqueda de rejilla (**GridSearch**) para el mejor modelo.
Métrica de comparación: **ROC area under the curve**

Preprocesamiento\ A priori distribution	Versión 1	Versión 2 (común)
Multinomial		
Bernouilli		
Gaussian		

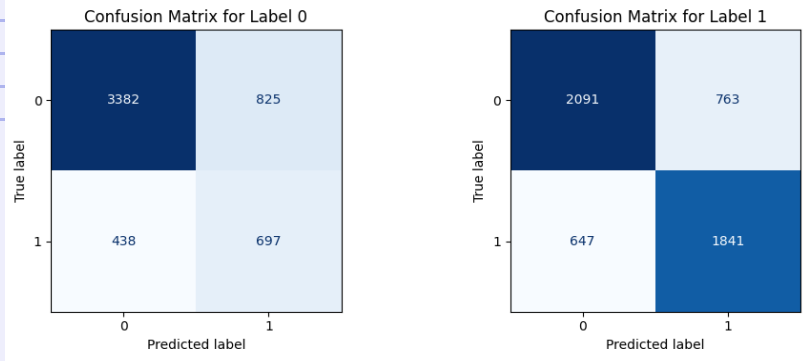
4.2. Naïve Bayes – Preprocesamiento

Descripción de la versión de preprocesamiento final:

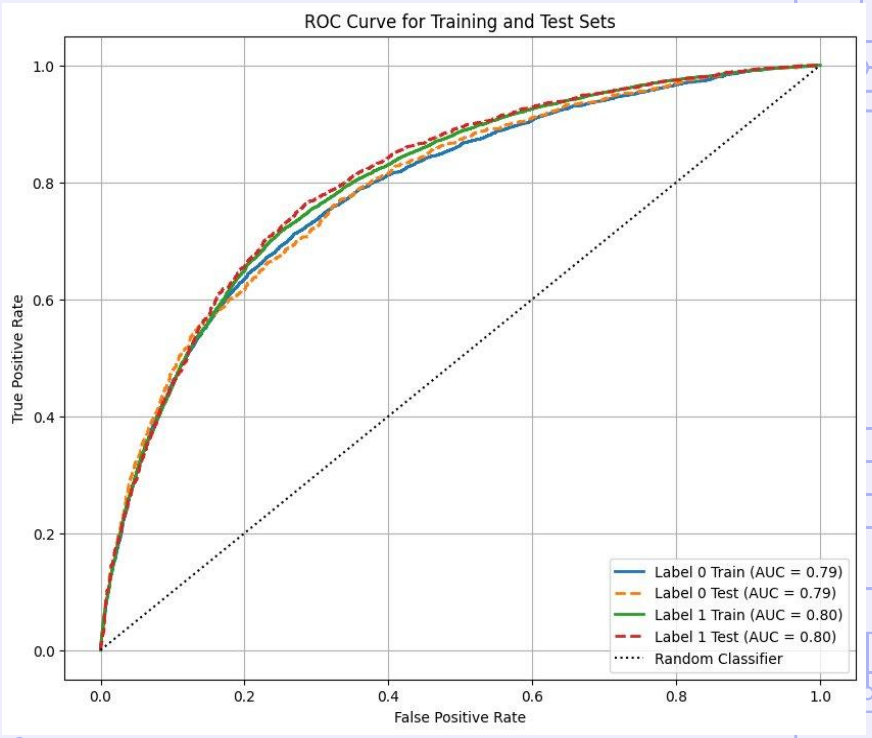



```
ordinal_cols = ['h1n1_concern', 'h1n1_knowledge', 'opinion_h1n1_vacc_effective', 'opinion_h1n1_risk',
                'opinion_h1n1_sick_from_vacc', 'opinion_seas_vacc_effective', 'opinion_seas_risk', 'opinion_seas_sick_from_vacc']

ordinal_cols_str = ['age_group', 'education', 'income_poverty']

binary_cols = ['behavioral_antiviral_meds', 'behavioral_avoidance', 'behavioral_face_mask', 'behavioral_wash_hands',
               'behavioral_large_gatherings', 'behavioral_outside_home', 'behavioral_touch_face', 'doctor_recc_h1n1',
               'doctor_recc_seasonal', 'chronic_med_condition', 'child_under_6_months', 'health_worker', 'health_insurance',
               ]

binary_cols_str = ['sex', 'marital_status', 'rent_or_own']  #needs inputing of nan


nominal_cols_str = ['race', 'employment_status', 'hhs_geo_region', 'census_msa', 'employment_industry', 'employment_occupation']

numeric_cols = ['household_adults', 'household_children']
```

4.2. Naïve Bayes – Análisis de Resultados




```
TEST:
Primeras 5 probabilidades para label 0: [1.62095552e-03 9.90029029e-01 3.79913914e-04 3.66722749e-04
 1.80916793e-02]
Primeras 5 probabilidades para label 1: [2.37988700e-01 9.99189459e-01 7.32631433e-04 1.25016481e-03
 4.94526243e-01]

ROC AUC global: 0.7226507177746844
Accuracy global: 0.5922875327592662
ROC AUC Scores de cada label: [0.7909586393141701, 0.8042581968783025]
Accuracy Scores de cada label: [0.7635716959940098, 0.7360539123923624]

Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.46      0.61      0.52      1135
           1       0.71      0.74      0.72      2488

   micro avg       0.62      0.70      0.66      3623
   macro avg       0.58      0.68      0.62      3623
weighted avg       0.63      0.70      0.66      3623
 samples avg       0.31      0.34      0.32      3623
```



4.3. SVM

4.3. Support Vector Machine

Preprocesamiento

Las máquinas de soporte vectorial trabajan buscando el hiperplano de separación máxima entre dos clases. Por este mismo motivo se requiere que, además del preprocesamiento básico aplicado, donde se eliminan los registros con una alta tasa de no respuesta, y de la codificación específica para cada tipo de variable; también es necesario escalar los datos iniciales.

De las técnicas aplicables a esta base de datos se ha escogido un escalado MinMax, ya que conserva las características en las distancias de dos observaciones

Abordaje del problema con SVM

El modelo genérico de SVM no es capaz de manejar los problemas multietiqueta, para solventar este problema se proponen dos tipos de soluciones:

1. **Usar la función MultiOutputClassifier. Esta función envuelve a los modelos básicos para poder manejar problemas multietiqueta.**
2. **Usar estrategias específicas para manejar problemas multiclase, como OVO y OVR (aclarar que estos métodos son usados para extender modelos binarios a problemas multiclase, no son usados como tal para problemas multisalida)**

4.3. Support Vector Machine

Configuraciones de los algoritmos y optimización con GridSeach

1. Modelo SVM con Kernel lineal y desbalanceo

Parámetros de búsqueda (param_grid):

- **C:[0.01, 0.1, 1, 10]**
- **Iteraciones máximas**: [10000, 20000, 50000]
- **Dual**: [True, False]
- **Tolerancia**: [1e-3, 1e-2]
- CV: 3 folds
- Random state: 42

2. Modelo SVM con Kernel lineal y balanceo

Parámetros de búsqueda (param_grid):

- **C:[0.01, 0.1, 1, 10]**
- **Iteraciones máximas**: [10000, 20000, 50000]
- **Dual**: [True, False]
- **Tolerancia**: [1e-3, 1e-2]
- **Class Weight**: 'balanced'
- CV: 3 folds
- Random state: 42

3. Modelo SVM con Kernel RBF y desbalanceo

Parámetros de búsqueda (param_grid):

- **C:**[0.1, 1, 10]
- **Gamma:** [0.01, 0.1, 1, 10]
- **Iteraciones máximas**: [10000, 50000]
- **Probability**: True
- **Class_Weight:** [`balanced`, None]
- CV: 3 folds
- Random state: 42

4. Modelo OVR para LinearSVM

Parámetros optimizados para modelo base

- **C:**[0.1]
- **Tol:** [0.01]
- **Iteraciones máximas**: [10000]
- **Dual**: False
- **Class_weight**: 'balanced'
- CV: 3 folds
- Random state: 42

5. Modelo OVR para SVM RBF

Parámetros optimizados para modelo base

- **C:**[10]
- **Gamma:** [0.01]
- **Iteraciones máximas**: [50000]
- **Probability**: True
- CV: 3 folds
- Random state: 42

4.3. Support Vector Machine

Resultados

1. Modelo SVM con Kernel lineal y desbalanceo

Modelo óptimo
LinearSVC(dual=True, C=10, tol=0.001, max_iter=50000, random_state= 42)

ROC_AUC en test: 0.7506
Accuracy en test: 0.6858

2. Modelo SVM con Kernel lineal y balanceo

Modelo óptimo
LinearSVC(dual=False, C=0.1, tol=0.001, max_iter=10000, random_state= 42, class_weight='balanced')

ROC_AUC en test: 0.7906
Accuracy en test: 0.6591

3. Modelo SVM con Kernel RBF y balanceo

Modelo óptimo
LinearSVC(kernel='rbf', probability=True, C=10, gamma=0.01, max_iter=50000, random_state= 42, class_weight='balanced')

ROC_AUC en test: 0.794767
Accuracy en test: 0.6683

4. Modelo OVR para LinearSVM

Modelo:
LinearSVC(dual=False, C=0.1, tol=0.001, max_iter=10000, random_state= 42, class_weight='balanced')

ROC_AUC en test: 0.7906
Accuracy en test: 0.6858

5. Modelo OVR para SVM RBF

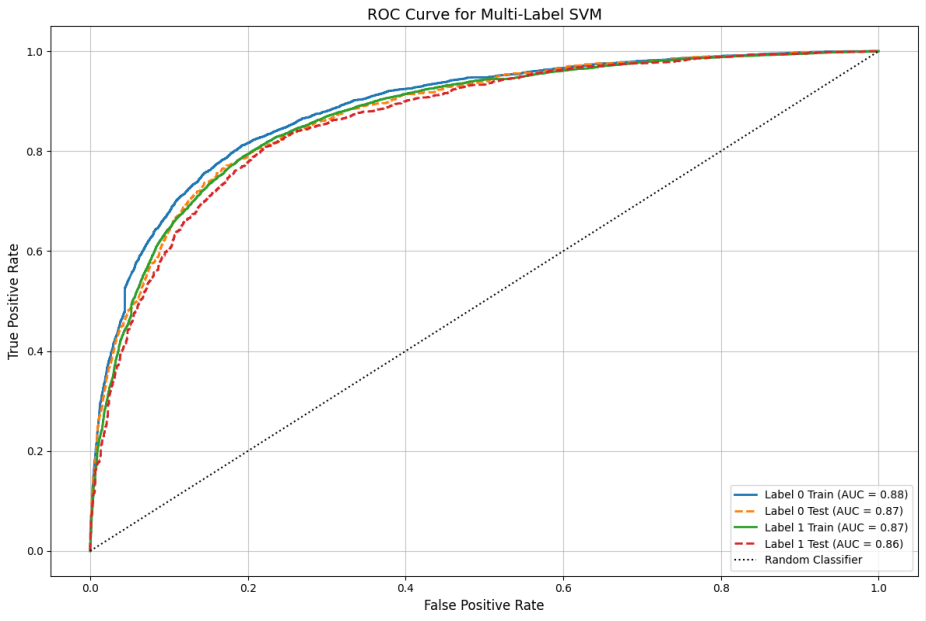
Modelo:
SVC(kernel='rbf', probability=True, C=10, gamma=0.01, max_iter=50000, random_state= 42, class_weight='balanced')

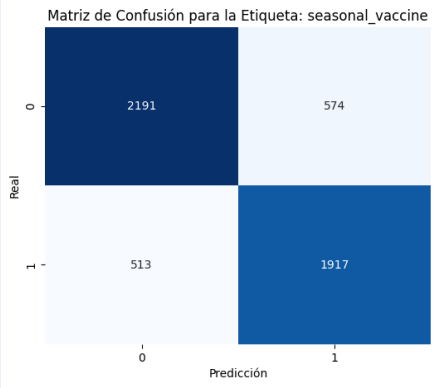
ROC_AUC en test: 0.7947
Accuracy en test: 0.0.6683

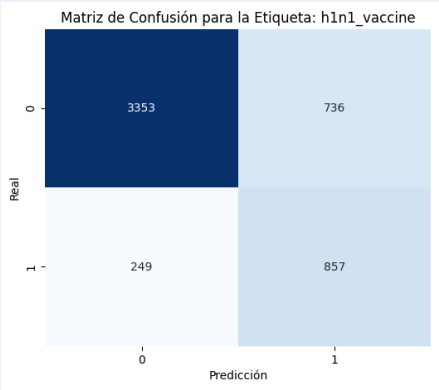
Los mejores modelos vienen dado por SVM RBF balanceado y OVR para SVM RBF balanceado (medimos la precisión del área bajo la curva ROC)

4.3. Support Vector Machine

Resultados







4.4. Regresión Logística

Preprocesamiento

Para aplicar la regresión logística se han seguido los mismos pasos de preprocesamiento que los descritos en la parte común.

Además, se ha probado escalar los datos para apurar la convergencia de los algoritmos y observar si hay diferencias en la capacidad de predicción.

También se ha probado un preprocesamiento sencillo donde los valores faltantes se imputaban por la moda, con peores resultados.

Configuraciones probadas del algoritmo

1. Modelo sin regularización, con GridSearch y balanceo

Parámetros de búsqueda (param_grid):
- **Solver**: ['saga', 'lbfgs']
- **Penalización**: [None]
- **Iteraciones máximas**: [500, 1000, 2000]
- **Intercepto**: [True, False]
- **Tolerancia**: [1e-3, 1e-4, 1e-5]
- CV: 5 folds

2. Modelo con regularización l1, l2, con Grid Search, con/sin balanceo

Parámetros de búsqueda (param_grid):
- **Solver**: ['liblinear', 'saga']
- **Penalización**: ['l1', 'l2']
- **Iteraciones máximas**: [100, 1000]
- **C**: [0.01, 0.1, 1, 10, 100, 1000]
- CV: 5 folds

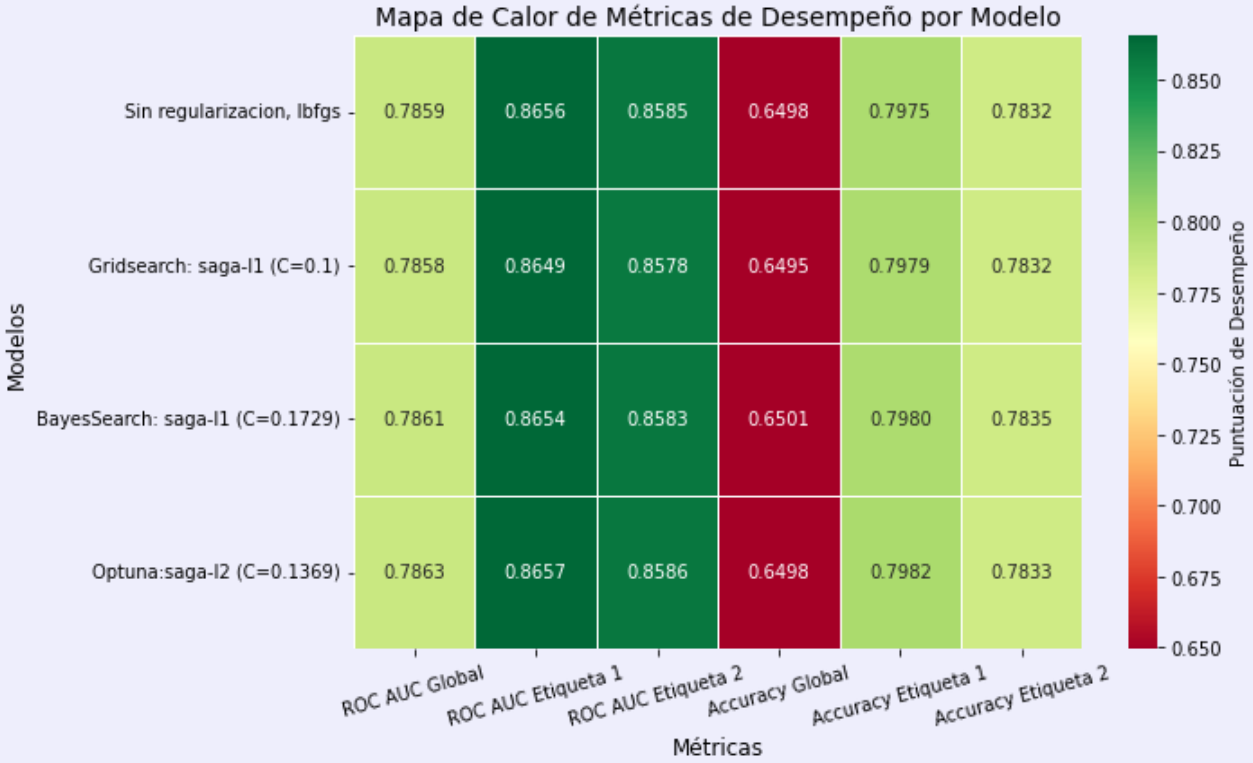
3. Modelo con con regularización l1, l2, con Bayes Search y balanceo

Espacio de parámetros (param_space):
- **Solver**: ['liblinear', 'saga']
- **Penalización**: ['l1', 'l2']
- **C**: (1e-6, 1e+6, 'log-uniform')
- CV: 5 folds

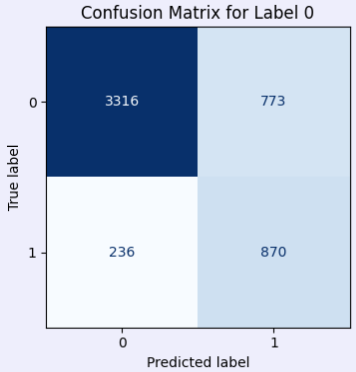
4. Modelo con con regularización l1, l2, con Optuna y balanceo

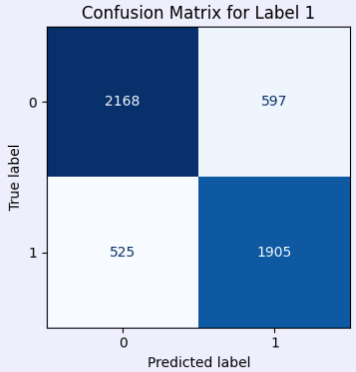
Configuración de parámetros:
- **Solver**: ['liblinear', 'saga']
- **Penalización**: ['l1', 'l2']
- **C**: entre 1e-6 y 1e+6 (escala logarítmica)
- **Iteraciones máximas**: 5000
- CV: 5 folds

Mejores Modelos en training, sin escalar



Modelo final, indicado por BayesSearch

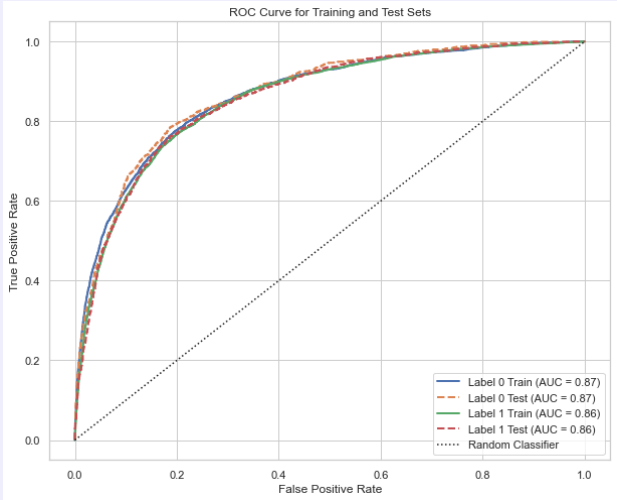


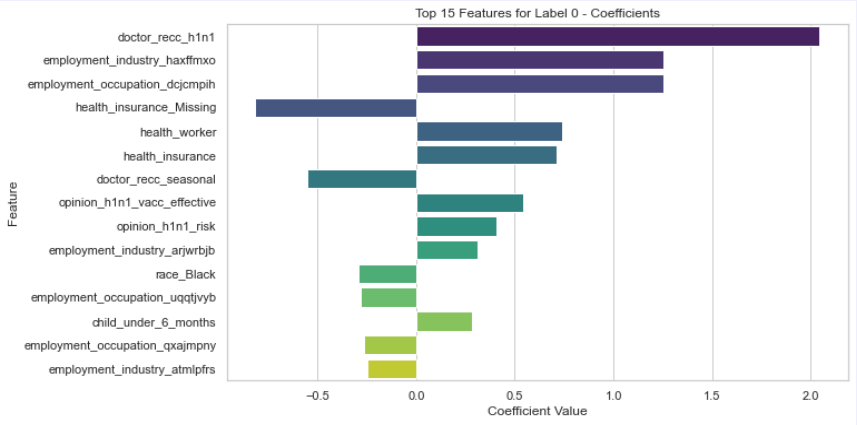


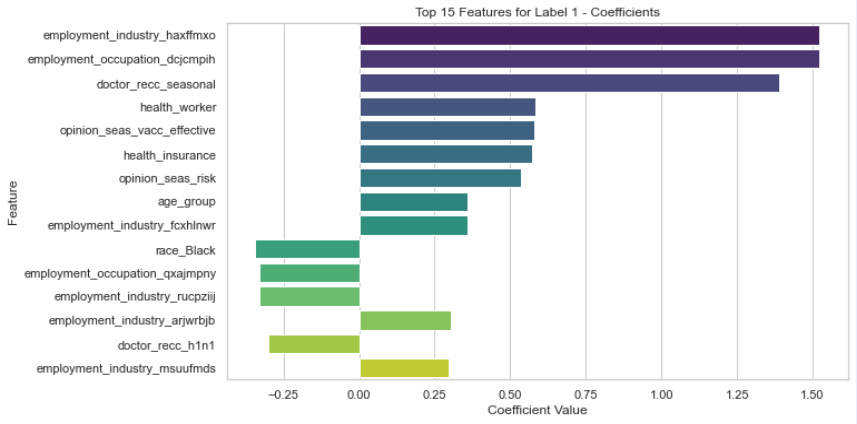

```
Primeras 5 probabilidades para label 0: [0.03566363 0.78700736 0.22852407 0.35257528 0.44705833]
Primeras 5 probabilidades para label 1: [0.5190729  0.80887705 0.15751482 0.14493832 0.38309469]

ROC AUC global: 0.791403021355642
Accuracy global: 0.6565928777670837
ROC AUC Scores de cada label: [0.8692350181340401, 0.8587983241429092]
Accuracy Scores de cada label: [0.8057747834456208, 0.7840230991337824]

Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.53      0.79      0.63      1106
           1       0.76      0.78      0.77      2430

   micro avg       0.67      0.78      0.72      3536
   macro avg       0.65      0.79      0.70      3536
weighted avg       0.69      0.78      0.73      3536
 samples avg       0.36      0.39      0.36      3536
```

El solver SAGA, con regularizacion de tipo LASSO y C igual a 0.1729 ofrece un accuracy global solo un poco mas grande que los otros modelos.LASSO es capaz de descartar variables, por lo cual es también conveniente.

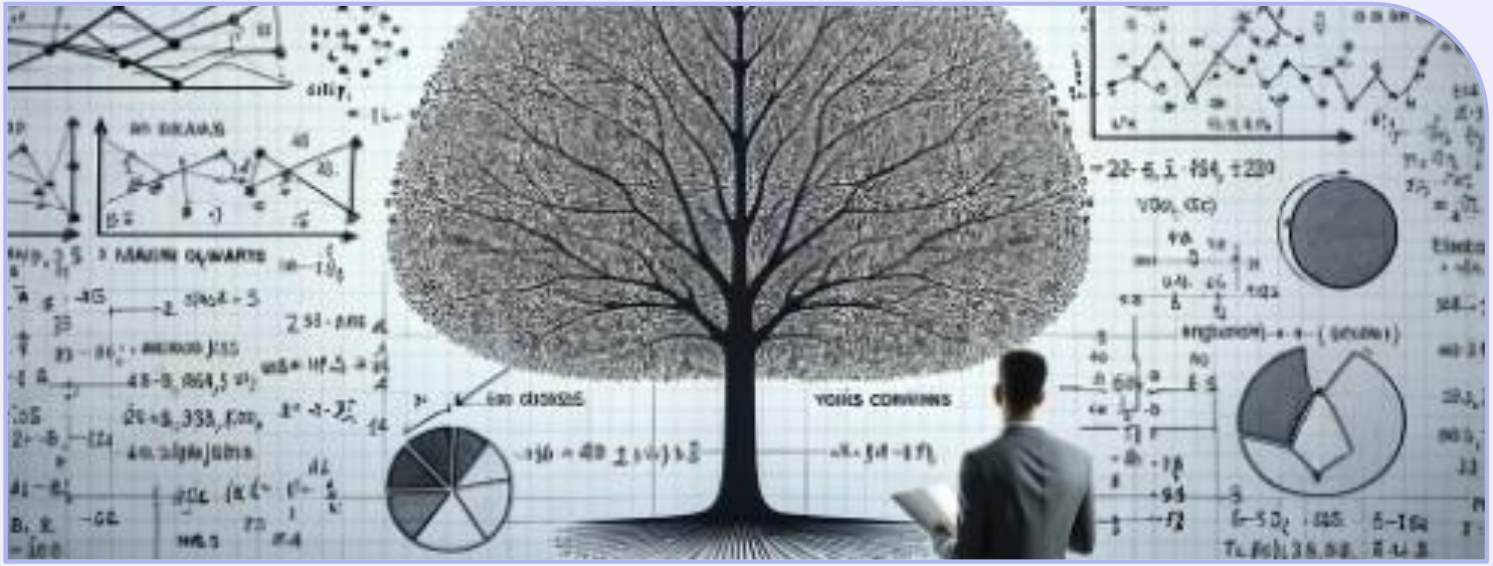
Los valores de ROC AUC para las etiquetas 0 y 1 son aproximadamente 0.8653 y 0.8582, mientras que la métrica accuracy indica una exactitud de 0.7979, respectivamente 0.7835.







¿Que influye en la decisión de vacunarse con h1n1_vaccine y seasonal_vaccine?



4.5. *Árbol de Clasificación*

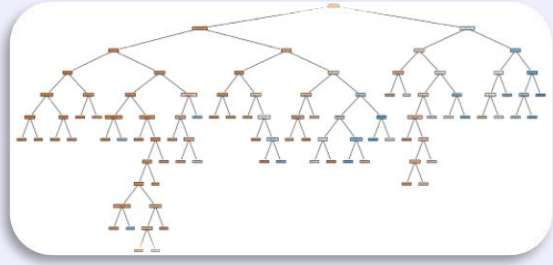
CART (Classification and Regression Tree)

Se implementó este modelo de árbol utilizando *DecissionTreeClassifier* combinado con *MultiOutputClassifier* para abordar la clasificación multietiqueta.

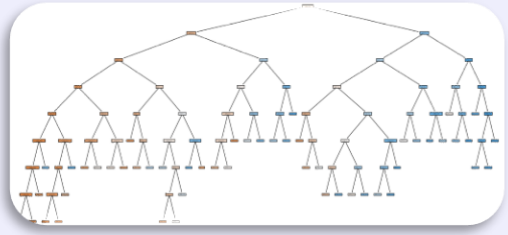
Se ajustó el preprocesamiento para obtener mejores métricas tanto en CART como en ensembles (GradientBoosting):

1. Asignar 'Not Applicable' en estado laboral.
2. Orden explícito para columnas ordinales_str.
3. Categorías faltantes en columnas categóricas.
4. Indicadores para valores faltantes en numéricas.
5. Imputación extrema en columnas numéricas.
6. Codificación de columnas categóricas y ordinales.

Árbol H1N1.



Árbol Seasonal.



Optimización e Hiperparámetros

Se probó la optimización con GridSearchCV y con Optuna

Para **DecisionTreeClassifier**, GridSearchCV funciona mejor porque el espacio de hiperparámetros es pequeño y discreto. Sin embargo, para métodos de **Gradient Boosting** también se ha probado la librería Optuna y, en algún caso, supera a la búsqueda de scikit-learn debido a la naturaleza más compleja y amplia del espacio de hiperparámetros. Se mostrarán siempre los resultados de los modelos con mejores métricas obtenidas.

GridSearchCV: {‘criterion': 'entropy', ‘max_depth': None, ‘max_features': None, ‘max_leaf_nodes': 50, ‘min_samples_leaf': 5, ‘min_samples_split': 2, ‘splitter': 'best'}.

Optuna: {‘criterion’: gini, ‘max_depth’: 6, ‘min_samples_split’: 9, ‘min_sample_leaf’: 10, ‘max_features’: None, ‘min_impurity_decrease’: 0.0, ‘splitter’: best, ‘class_witght’: None.}

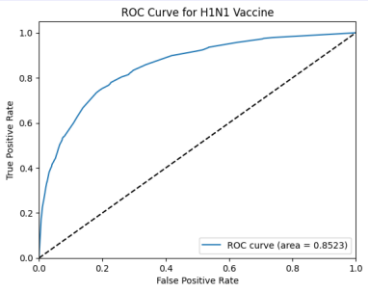
Evaluación del modelo

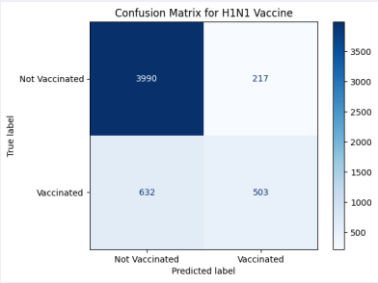
Se muestran resultados y gráficos con las métricas del mejor modelo, tanto en nuestra división de test como en competición.

Woohoo, your submission was successful! Your submission score is

0.8326

H1N1

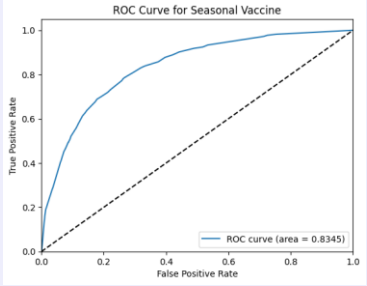


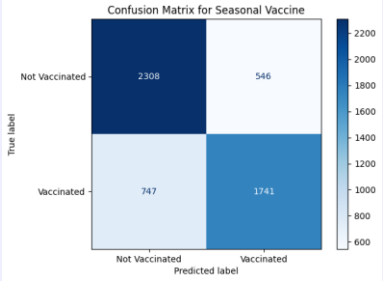


Classification Report for H1N1 Vaccine:

	precision	recall	f1-score	support
No	0.86	0.95	0.90	4207
Yes	0.70	0.44	0.54	1135
accuracy			0.84	5342
macro avg	0.78	0.70	0.72	5342
weighted avg	0.83	0.84	0.83	5342

Seasonal





Classification Report for seasonal Vaccine:

	precision	recall	f1-score	support
No	0.76	0.81	0.78	2854
Yes	0.76	0.70	0.73	2488
accuracy			0.76	5342
macro avg	0.76	0.75	0.76	5342
weighted avg	0.76	0.76	0.76	5342

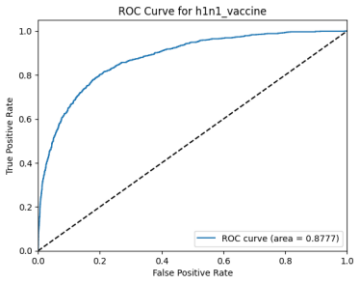
05. *Ensembles*

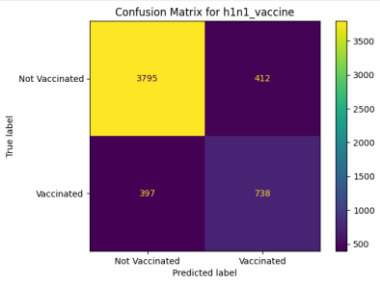
5.1. Gradient Boosting

CatBoost

H1N1

Mejores hiperparámetros:
{'auto_class_weights': 'SqrtBalanced', 'bagging_temperature': 1.0, 'depth': 5, 'iterations': 404, 'l2_leaf_reg': 10.0, 'learning_rate': 0.0428709991527715, 'random_strength': 0.0}



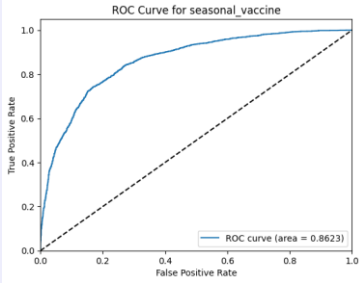


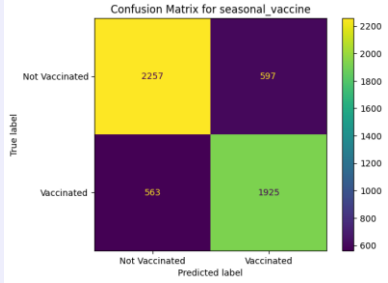
h1n1_vaccine Classification Report:

	precision	recall	f1-score	support
0	0.91	0.90	0.90	4207
1	0.64	0.65	0.65	1135
accuracy			0.85	5342
macro avg	0.77	0.78	0.77	5342
weighted avg	0.85	0.85	0.85	5342

Seasonal

Mejores hiperparámetros:
{'auto_class_weights': 'Balanced', 'bagging_temperature': 1.0, 'depth': 8, 'iterations': 500, 'l2_leaf_reg': 10.0, 'learning_rate': 0.0202879645, 'random_strength': 0.0}





seasonal_vaccine Classification Report:

	precision	recall	f1-score	support
0	0.80	0.79	0.80	2854
1	0.76	0.77	0.77	2488
accuracy			0.78	5342
macro avg	0.78	0.78	0.78	5342
weighted avg	0.78	0.78	0.78	5342

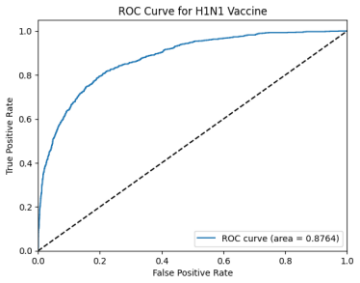
Woohoo, your submission was successful! Your submission score is

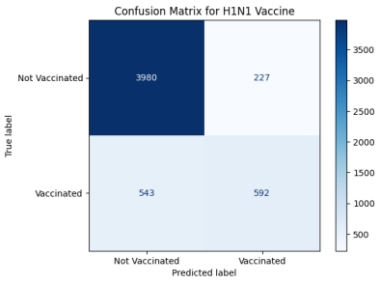
0.8595

XGBoost

Mejores Hiperparámetros: {'estimator__colsample_bytree': 1.0, 'estimator__gamma': 5.298302625766, 'estimator__learning_rate': 0.0326482054329, 'estimator__max_depth': 10, 'estimator__min_child_weight': 1, 'estimator__n_estimators': 403, 'estimator__reg_alpha': 0.941380446393, 'estimator__reg_lambda': 1.0, 'estimator__subsample': 0.459940784457}

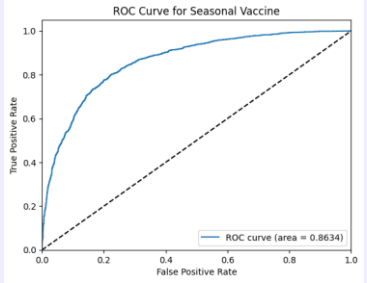
H1N1

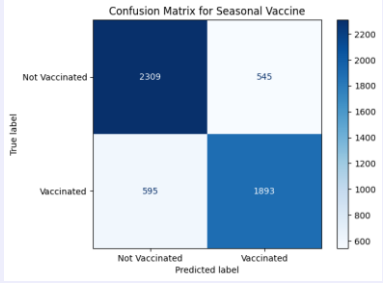




Classification Report for H1N1 Vaccine:	precision	recall	f1-score	support
No	0.88	0.95	0.91	4207
Yes	0.72	0.52	0.61	1135
accuracy			0.86	5342
macro avg	0.80	0.73	0.76	5342
weighted avg	0.85	0.86	0.85	5342

Seasonal





Classification Report for seasonal Vaccine:	precision	recall	f1-score	support
No	0.80	0.81	0.80	2854
Yes	0.78	0.76	0.77	2488
accuracy			0.79	5342
macro avg	0.79	0.78	0.79	5342
weighted avg	0.79	0.79	0.79	5342

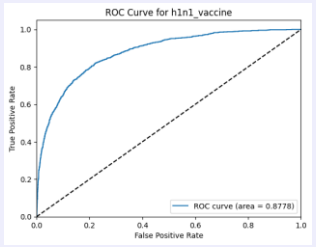
Woohoo, your submission was successful! Your submission score is

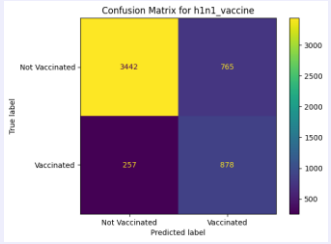
0.8601

LightGBM

H1N1

Mejores hiperparámetros: {'n_estimators': 350, 'max_depth': 9, 'reg_alpha': 1.6438000000000001, 'colsample_bytree': 0.378, 'learning_rate': 0.0592, 'reg_lambda': 83.161, 'subsample': 0.8698000000000001, 'class_weight': 'balanced', 'min_gain_to_split': 0.418, 'min_data_in_leaf': 3}

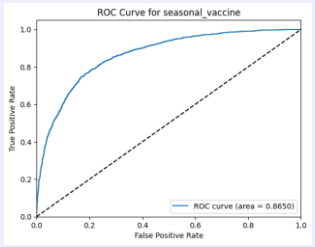


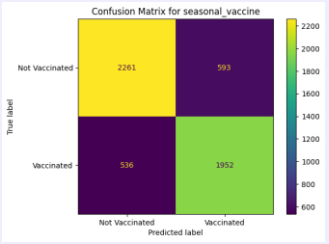


h1n1_vaccine Classification Report:	precision	recall	f1-score	support
0	0.93	0.82	0.87	4207
1	0.53	0.77	0.63	1135
accuracy			0.81	5342
macro avg	0.73	0.80	0.75	5342
weighted avg	0.85	0.81	0.82	5342

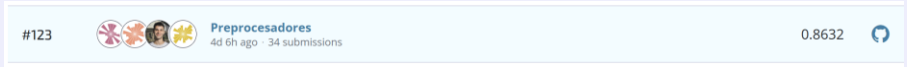
Seasonal

Mejores hiperparámetros: {'n_estimators': 350, 'learning_rate': 0.0463, 'max_depth': 10, 'reg_alpha': 1.7789000000000001, 'reg_lambda': 34.0818, 'subsample': 0.4188, 'colsample_bytree': 0.36719999999999997, 'class_weight': 'balanced', 'min_gain_to_split': 0.115, 'min_data_in_leaf': 10}

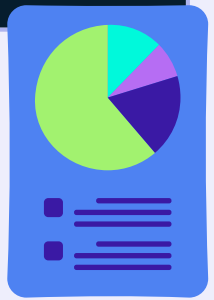




seasonal_vaccine Classification Report:	precision	recall	f1-score	support
0	0.81	0.79	0.80	2854
1	0.77	0.78	0.78	2488
accuracy			0.79	5342
macro avg	0.79	0.79	0.79	5342
weighted avg	0.79	0.79	0.79	5342



5.2. Stacking



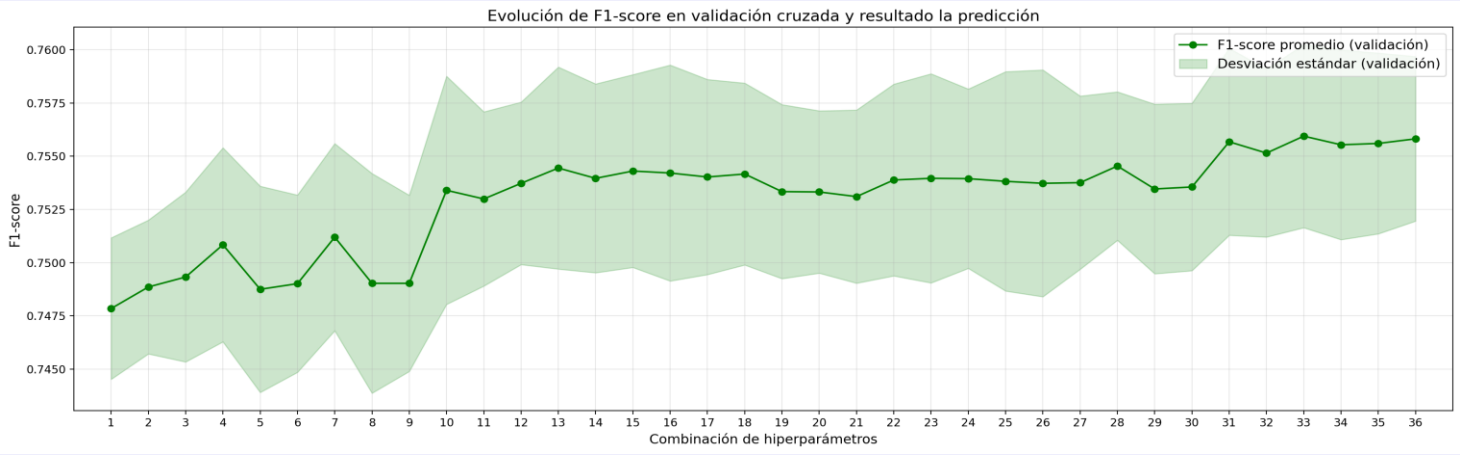
Preprocesamiento

Para aplicar el ensemble Stacking, dado que también empleamos como modelo base kNN, se han seguido los mismos pasos de **preprocesamiento** descritos en la **parte común, escalando** además **los datos usando MinMaxScaler**, dado que este algoritmo se basa en distancias, y las características con rangos más grandes pueden dominar las más pequeñas, distorsionando los resultados.

Estructura de los datos después del preprocesamiento

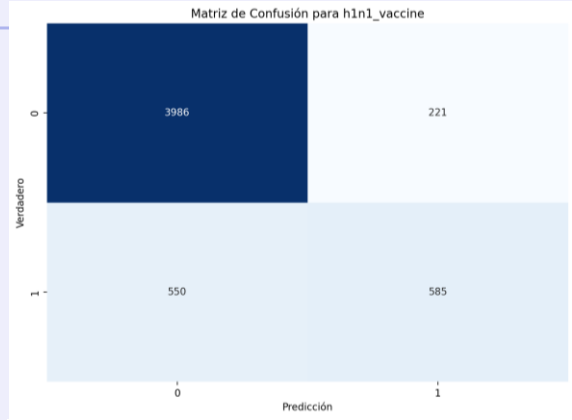
```
Training Features shape: (21365, 96)
Training Labels shape: (21365, 2)
Validation Features shape: (5342, 96)
Validation Labels shape: (5342, 2)
Test Features shape: (26708, 96)
```

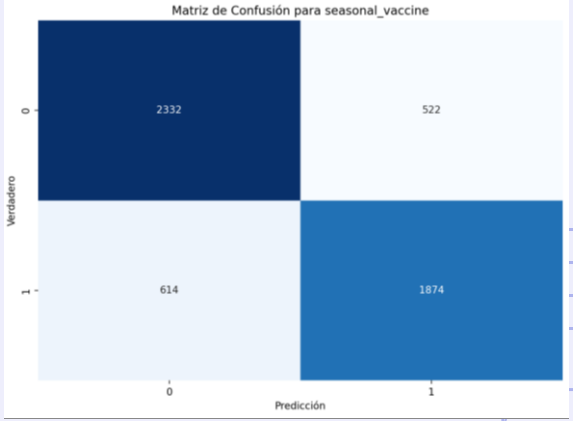
Tras probar diferentes combinaciones de modelos e hiperparámetros, implementamos un ensemble stacking adaptado al problema multietiqueta, enfocado en predecir la probabilidad de recibir dos tipos de vacunas (H1N1 y gripe estacional). Este modelo combina como estimadores base k-Nearest Neighbors, Decision Tree y Gradient Boosting, mientras que una regresión logística sirve como meta-modelo para integrar sus predicciones. Optimizamos los hiperparámetros clave mediante `GridSearchCV`, logrando un equilibrio entre sesgo y varianza. La inclusión de un `MultiOutputClassifier` para manejar ambas etiquetas permitió obtener un F1-score promedio por muestra sobresaliente, destacando este enfoque como el más robusto y efectivo.



```
Accuracy en validación: 0.6916885061774616

F1 Score (weighted): 0.7158316930709608
Precision (weighted): 0.7644899025405792
Recall (weighted): 0.678719293403257

Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.73      0.52      0.60      1135
           1       0.78      0.75      0.77      2488

   micro avg       0.77      0.68      0.72      3623
   macro avg       0.75      0.63      0.69      3623
weighted avg       0.76      0.68      0.72      3623
 samples avg       0.35      0.34      0.34      3623
```

```
Reporte de clasificación para h1n1_vaccine:
              precision    recall  f1-score   support

           0       0.88      0.95      0.91      4207
           1       0.73      0.52      0.60      1135

    accuracy                           0.86      5342
   macro avg       0.80      0.73      0.76      5342
weighted avg       0.85      0.86      0.85      5342

Reporte de clasificación para seasonal_vaccine:
              precision    recall  f1-score   support

           0       0.79      0.82      0.80      2854
           1       0.78      0.75      0.77      2488

    accuracy                           0.79      5342
   macro avg       0.79      0.79      0.79      5342
weighted avg       0.79      0.79      0.79      5342
```





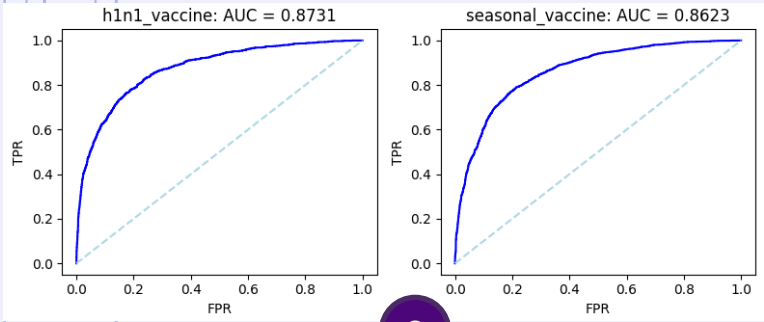
Mejores hiperparámetros

```
'estimator__boost__learning_rate': 0.1, 'estimator__boost__n_estimators':
200, 'estimator__final_estimator__C': 1, 'estimator__tree__max_depth': 10
```

	Final Estimator C	Tree Max Depth	Boost N Estimators	Boost Learning Rate	Mean F1-Score	Std F1-Score
32	1	10	200	0.1	0.755940	0.004299
35	10	10	200	0.1	0.755816	0.003863
30	1	5	200	0.1	0.755675	0.004389
34	10	7	200	0.1	0.755597	0.004244
33	10	5	200	0.1	0.755535	0.004455
31	1	7	200	0.1	0.755145	0.003943
27	0.1	5	200	0.1	0.754536	0.003486
12	1	5	200	0.05	0.754443	0.004742
14	1	10	200	0.05	0.754302	0.004527
15	10	5	200	0.05	0.754209	0.005071

Predicción de probabilidades

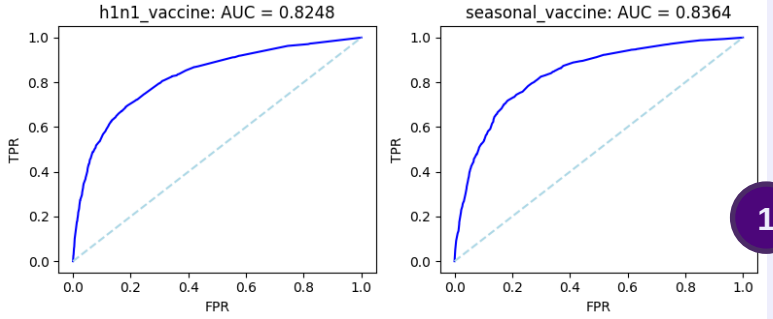
```
AUROC para H1N1: 0.87309089
AUROC para vacuna estacional: 0.86230148
AUROC promedio: 0.86769619
```



2

Modelos base:
1. k-Nearest Neighbors (kNN)
2. Árbol de decisión (Decision Tree)
3. Gradient Boosting.

Meta modelo: Regresión Logística.



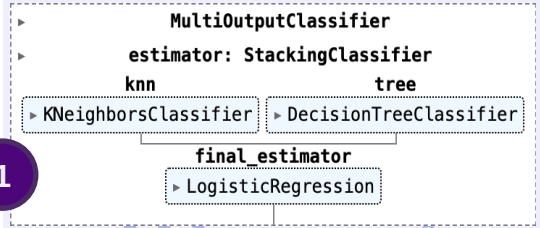
1

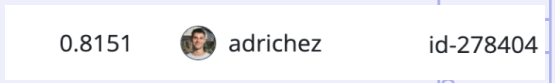
Archivo de entrega en la competición

```
Dimensiones del archivo de entrega: (26708, 2)
```

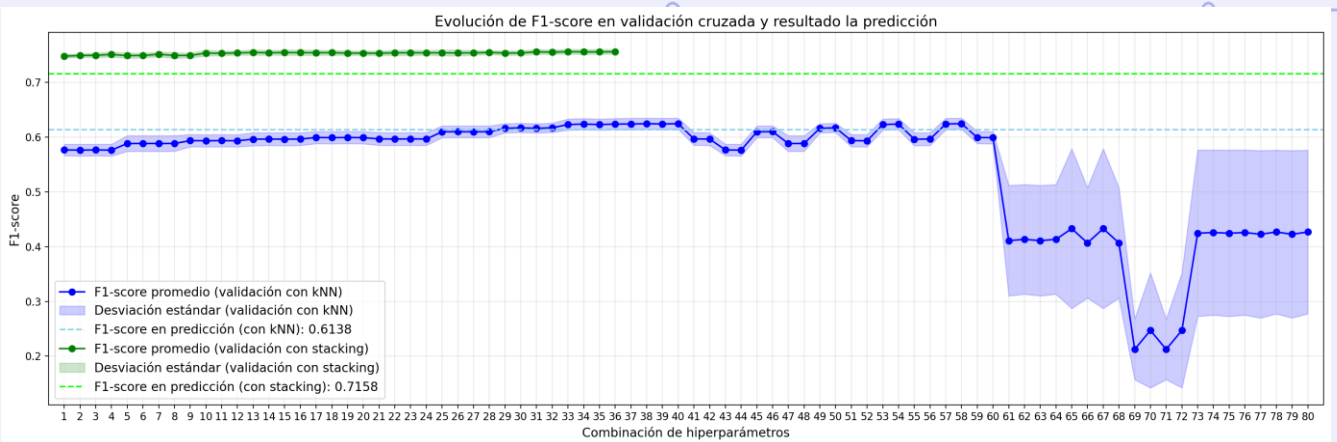
respondent_id	h1n1_vaccine	seasonal_vaccine
26707	0.099733	0.273661
26708	0.062585	0.086467
26709	0.111338	0.752619
26710	0.762210	0.887535
26711	0.196246	0.500750

respondent_id	h1n1_vaccine	seasonal_vaccine
26707	0	0
26708	0	0
26709	0	1
26710	1	1
26711	0	1



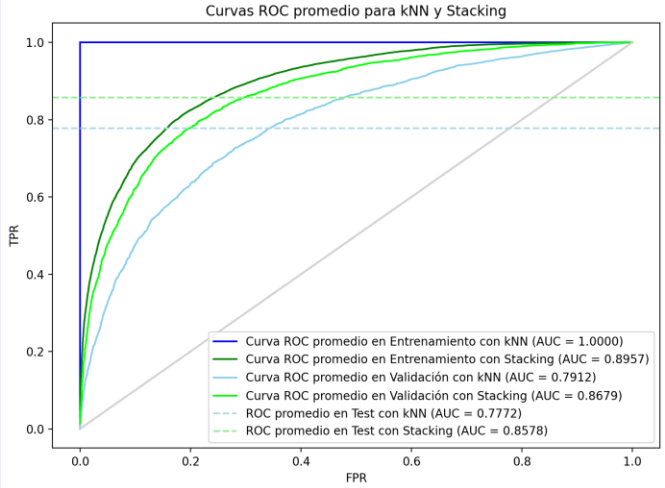


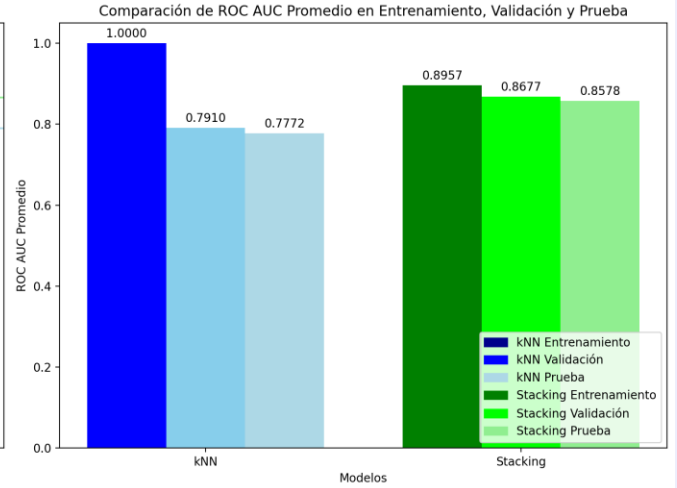
Comentarios finales kNN y Stacking



La línea azul (kNN) y la línea verde (Stacking) representa el F1-score promedio obtenido en las particiones de validación cruzada para cada configuración de hiperparámetros, mientras que la banda sombreada alrededor de la línea muestra la desviación estándar de los F1-scores, lo que indica la variabilidad del rendimiento del modelo entre las particiones.

En resumen, el modelo de Stacking ha logrado un rendimiento superior al modelo kNN en la competición de DrivenData, lo que sugiere que la combinación de múltiples clasificadores en un ensemble de Stacking ha sido efectiva para mejorar la capacidad predictiva del modelo.





El modelo Stacking ha demostrado ser más efectivo y robusto en términos de generalización y rendimiento en los conjuntos de validación y prueba en comparación con kNN. Aunque kNN ha mostrado un rendimiento perfecto en el conjunto de entrenamiento, su sobreajuste lo hace menos adecuado para datos nuevos datos no vistos.

Con ajustes adicionales en los hiperparámetros y el preprocesamiento de datos, ambos modelos podrían mejorar más, pero Stacking ya ha mostrado una ventaja clara en esta práctica.

5.3. Bagging

5.3. Bagging

Preprocesamiento

Para aplicar Bagging (Boostrap Aggregating), se han seguido los mismos pasos de preprocesamiento descritos en la parte común.

Se escalan los datos usando MinMaxScaler para facilitar la separación de las clases con SVM, ya que la complejidad del algoritmo aumenta y la ejecución se alarga.

Configuraciones probadas y resultados

1. Modelo de Bagging Clásico

Criterios utilizados:
- **Estimador base: SVM RBF**
- **N_estimadores: 10**
- **Max_samples= 0.8**
- **Max_features: 1**
- **Boostrap: True**
- **Random State: 42**

```
=== Bagging_SVM ===
Accuracy: 0.6721847930702599
Roc-AUC: 0.7943632909967411
Classification Report:
precision recall f1-score support
0 0.55    0.77   0.64     1106
1 0.77    0.78   0.78     2430
```

2. Modelo de Bagging con Random Subespaces

Criterios utilizados:
- **Estimador base: SVM RBF**
- **N_estimadores: 10**
- **Max_samples= 1**
- **Max_features: 0.5**
- **Boostrap: False**
- **Random State: 42**

```
=== Random_subespaces ===
Accuracy: 0.658710298363811
Roc- AUC: 0.7843039373471437
Classification Report:
precision recall f1-score support 0
0.55    0.75   0.63     1106
1 0.77    0.75   0.76     2430
```

5.4. AdaBoost

Preprocesamiento

Para aplicar AdaBoost, se han seguido los mismos pasos de preprocesamiento descritos en la parte común.

Se escalan los datos usando MinMaxScaler para facilitar la convergencia en el caso de la regresión logística, ya que la complejidad del algoritmo aumenta y la ejecución se alarga.

Configuraciones probadas

1. Modelo en base a Árboles de decisión, con Bayes Search

Espacio de búsqueda :

- Número de estimadores: entre 50 y 300
- Tasa de aprendizaje: entre 0.01 y 1.0 (dist. Log.)
- Profundidad máxima: entre 1 y 10
- Mínimo de muestras para dividir: entre 2 y 20
- Mínimo de muestras por hoja: entre 1 y 10
- Algoritmo: SAMME
- CV: 5 folds

2. Modelo en base a Árboles de decisión, con Optuna

Espacio de búsqueda :

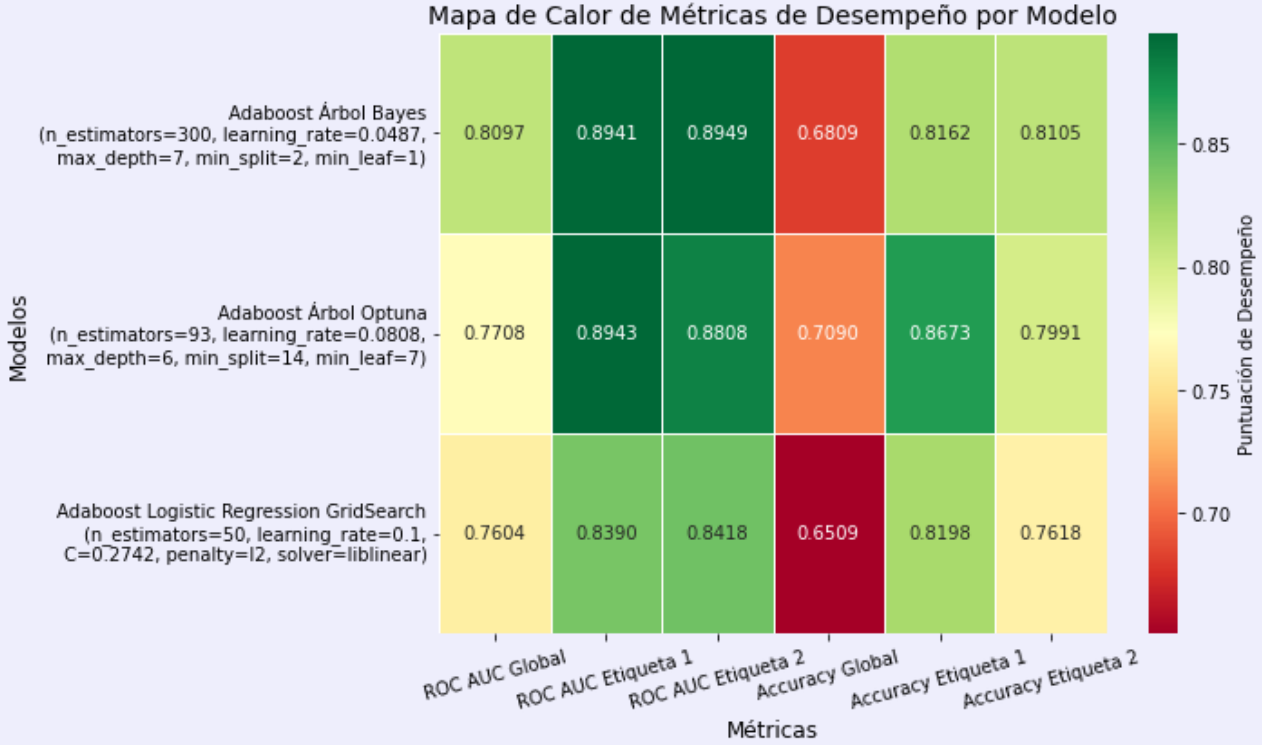
- Número de estimadores: entre 50 y 300
- Tasa de aprendizaje: entre 0.01 y 1.0 (dist. Log.)
- Profundidad máxima: entre 1 y 10
- Mínimo de muestras para dividir: entre 2 y 20
- Mínimo de muestras por hoja: entre 1 y 10
- Algoritmo: SAMME
- CV: 5 folds

3. Modelo en base a la regresión logística, con GridSearch

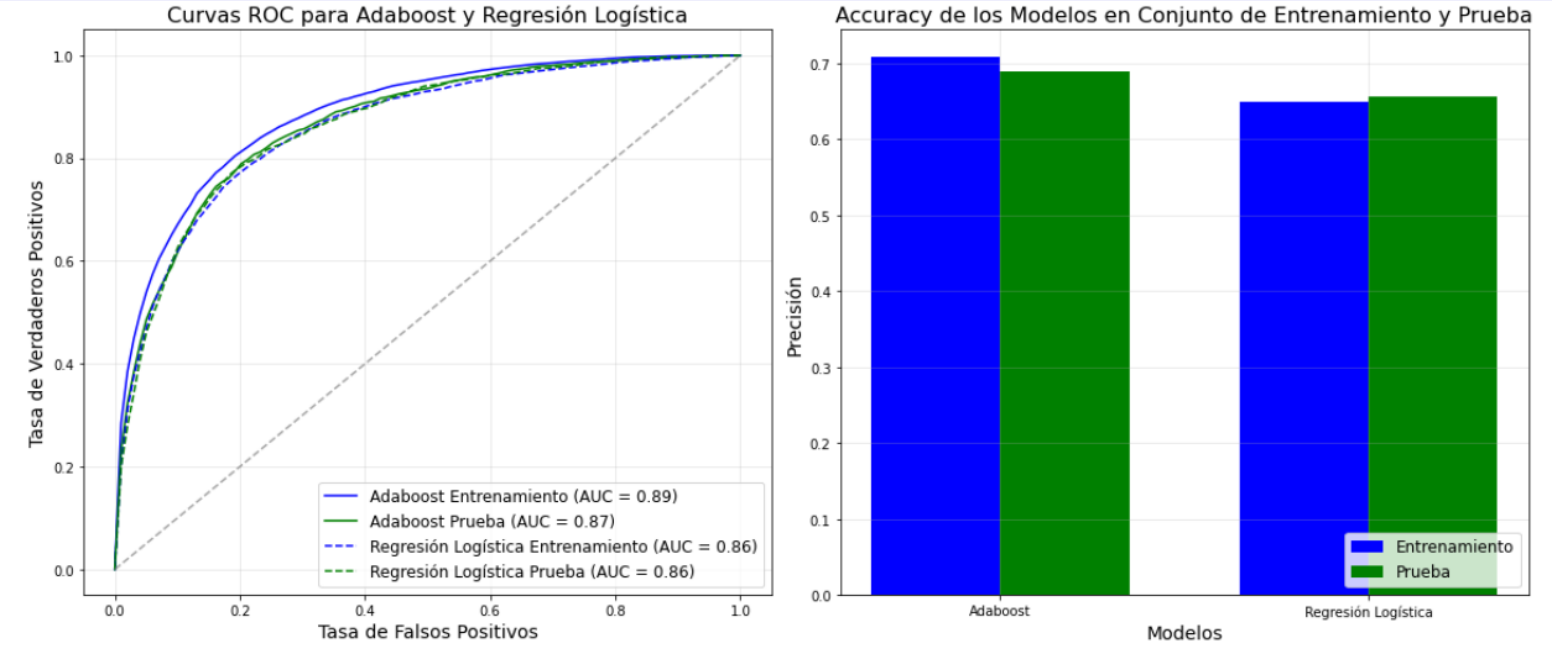
Espacio de búsqueda :

- **Número de estimadores**: [5, 10, 20, 30, 50]
- **Tasa de aprendizaje**: [0.01, 0.1, 1, 10, 100]
- **Algoritmo**: SAMME
- **Penalización**: 'l2'
- **Solver**: 'liblinear'
- **Máximo de iteraciones**: 5000
- **C**: 0.2742
- CV: 5 folds

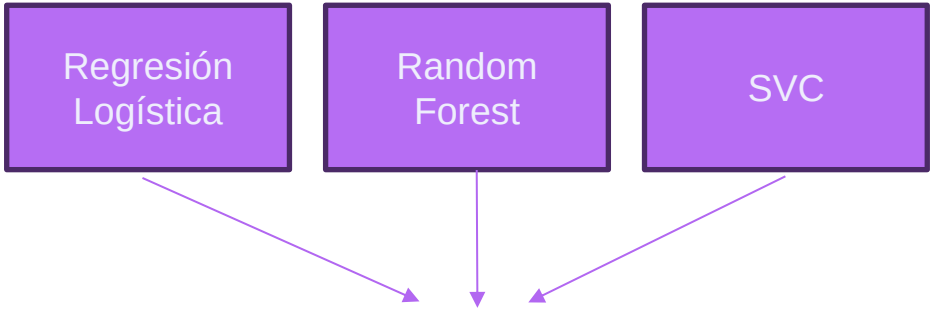
Mejores Modelos en training



El mejor modelo es el Adaboost basado en árboles de decisión y optimización con Optuna



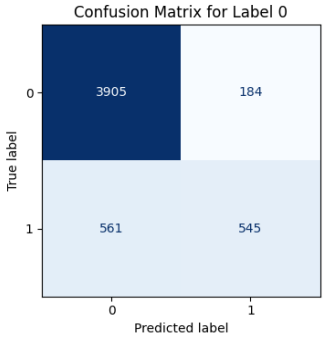
5.5. Voting

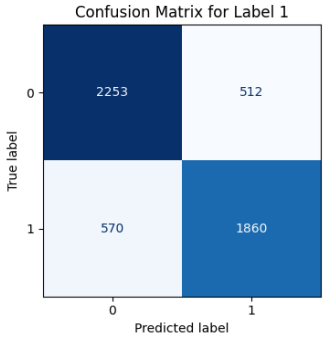


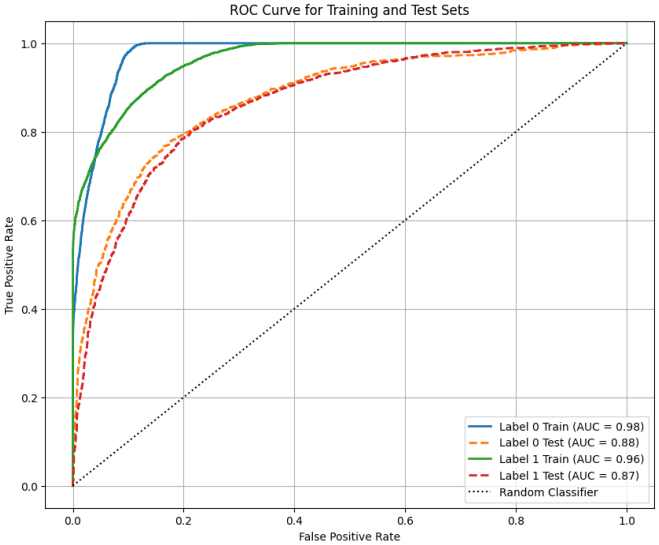
Seguimos los pasos del **preprocesamiento común**!

El Ensemble usa modelos básicos que hemos comprobado que funcionan especialmente bien con dicho preprocesamiento.

5.5. Voting – Análisis de Resultados

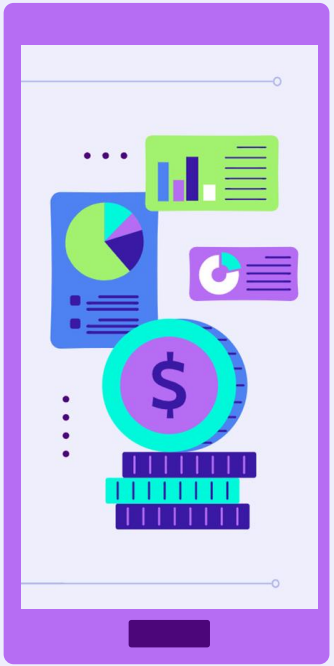







```
TEST:
Primeras 5 probabilidades para label 0: [0.03274049 0.46832802 0.06225028 0.12986161 0.16542537]
Primeras 5 probabilidades para label 1: [0.46899228 0.76517983 0.08933225 0.25107669 0.25651794]

ROC AUC global: 0.7570070645667879
Accuracy global: 0.6937439846005775
ROC AUC Scores de cada label: [0.8754179718266757, 0.8658382634191356]
Accuracy Scores de cada label: [0.8565928777670837, 0.7917228103946102]

Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.75      0.49      0.59      1106
           1       0.78      0.77      0.77      2430

   micro avg       0.78      0.68      0.72      3536
   macro avg       0.77      0.63      0.68      3536
weighted avg       0.77      0.68      0.72      3536
 samples avg       0.36      0.34      0.34      3536
```

06. Conclusión



Preprocesamiento

Aplicación de Modelos

¿Objetivo conseguido?

¡Gracias!

¿Alguna pregunta?

- pablogradolph@correo.ugr.es
- adrian31@correo.ugr.es
- sbowlder@correo.ugr.es
- emorella@correo.ugr.es
- ruxico@correo.ugr.es

GoogleDrive: https://drive.google.com/drive/folders/1tLyUgZ3QGCDYcq859Fb8aP-DyyxlyWhc?usp=sharing

GitHub: https://github.com/PabloGradolph/FluShotLearning