Checkpoint 3

K - nearest neighbours

Los resultados de la optimización de hiperparámetros y evaluación del modelo se presentan a continuación.

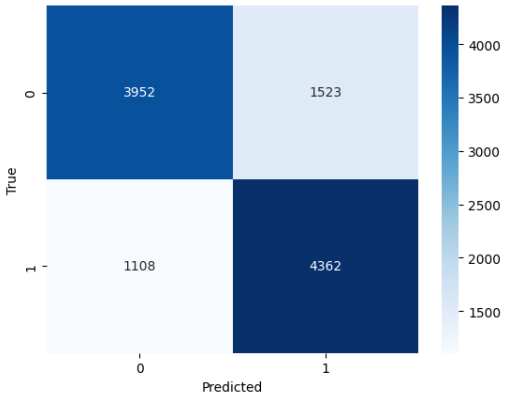
Grilla de parametros usada para k-fold CrossValidation

```
params_grid={ 'n_neighbors':range(1,30),
              'weights':['distance','uniform'],
              'algorithm':['ball_tree', 'kd_tree', 'brute'],
              'metric':['euclidean','manhattan','chebyshev']}
```

Mejores metricas obtenidas

Metrica	Valor usado	Descripcion
weights	distance	Determina cómo se ponderan las distancias de los vecinos cercanos.
n_neighbors	19	Número de vecinos más cercanos que se utilizarán para hacer una predicción
metric	manhattan	Métrica de distancia que se utilizará para medir la distancia entre los puntos.
algorithm	brute	Algoritmo que se utilizará para encontrar los vecinos más cercanos.

Matriz de confusion

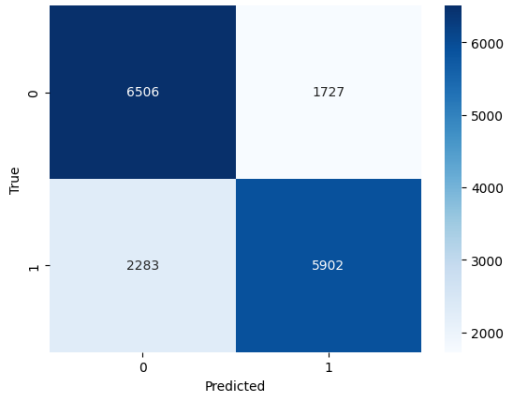


SVM

Mejores metricas obtenidas

Metrica	Valor	Descripcion
kernel	rbf	Función kernel que se utilizará para transformar los datos de entrada en un espacio de características de mayor dimensión.
C	42	Margen de error permitido en la clasificación errónea de los puntos.

Matriz de Confusion



XGBoost

Los resultados de la optimización de hiperparámetros y evaluación del modelo se presentan a continuación.

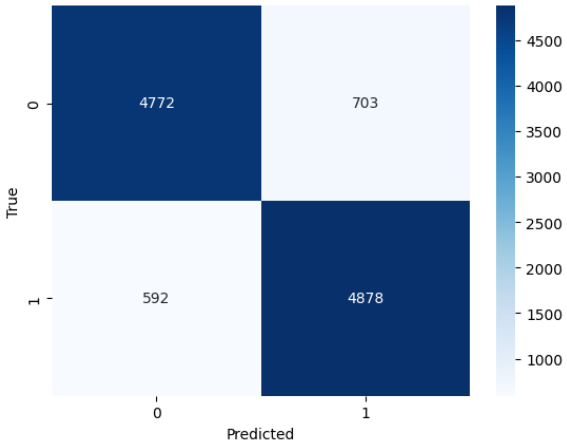
Grilla de parametros usada para k-fold CrossValidation

```
params_grid={ 'learning_rate':np.linspace(0.1,0.5,5),
              'n_estimators':range(10,100),
              'max_depth':range(4,10),
              'objective':['reg:logistic','binary:logistic']}
```

Mejores metricas obtenidas

Metrica	Valor	Descripcion
objective	reg:logistic	Función de pérdida que se utilizará para optimizar el modelo.
n_estimators	93	Número de árboles de decisión que se utilizarán en el modelo. En este caso, se establece en 93, lo que significa que se entrenarán 93 árboles de decisión.
max_depth	8	Profundidad máxima de cada árbol de decisión. Un valor más alto de max_depth permitirá que los árboles sean más complejos y se ajusten mejor a los datos de entrenamiento, pero también puede llevar a sobreajuste.
learning_rate	0.3	Controla la tasa de aprendizaje en el modelo, es decir, la cantidad en la que se ajustan los pesos del modelo después de cada iteración. Un valor más bajo de learning_rate puede llevar a una convergencia más lenta del modelo, pero puede ayudar a evitar el sobreajuste.

Matriz de confusion



Random Forest

Los resultados de la optimización de hiperparámetros y evaluación del modelo se presentan a continuación.

Grilla de parametros usada para k-fold CrossValidation

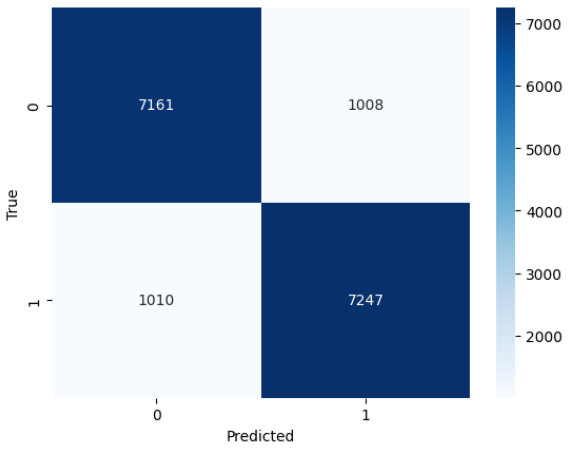
```
params_grid = { 'criterion':['gini','entropy'],
                'min_samples_leaf':list(range(1,10)),
                'min_samples_split': list(range(1,10)),
                'n_estimators':list(range(20,50))
               }
```

Mejores metricas obtenidas

Metrica	Valor	Descripcion
n_estimators	34	Número de árboles de decisión que se utilizarán. En este caso, se establece en 34, lo que significa que se entrenarán 34 árboles de decisión.
min_samples_split	7	Controla el número mínimo de muestras requeridas para dividir un nodo interno del árbol de decisión. Un valor más alto de min_samples_split puede ayudar a prevenir el sobreajuste al hacer que el árbol de decisión sea menos complejo.
min_samples_leaf	1	Controla el número mínimo de muestras requeridas para formar un nodo hoja en el árbol de decisión. Un valor más alto de min_samples_leaf puede ayudar a prevenir el sobreajuste al hacer que el árbol de decisión sea menos complejo.
criterion	entropy	Función de calidad que se utilizará para medir la calidad de una división en el árbol de decisión.

Acuracy score obtenido: 0.87

Matriz de confusion



Voting & Stacking

Los modelos usados para entrenar los ensambles son los que encontramos en los incisos anteriores con sus respectivos mejores hiperparametros.

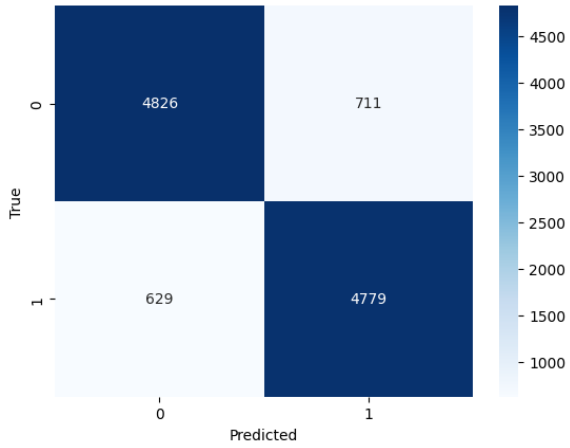
Voting

Para voting usamos los modelos XGBoost, Random Forest y Knn con los siguientes hiperparametros

```
xgb.XGBClassifier(objective='reg:logistic',max_depth=8,learning_rate=0.3, n_estimators= 93)
RandomForestClassifier(max_features='auto', oob_score=True, random_state=3, n_jobs=-1, n_estimators=34, min_samples_split=7,min_sa
KNeighborsClassifier(weights='distance',n_neighbors= 19, metric= 'manhattan', algorithm='brute')
```

Accuracy score obtenido: 0.87

Matriz de confusion



Stacking

Usamos Rnadom Forest y Knn como modelos base y XGBoost como meta modelo.

```
#Modelos Base
base_models = [('random_forest', RandomForestClassifier(max_features='auto', oob_score=True, random_state=3, n_jobs=-1, n_estimato
               ('knn', KNeighborsClassifier(weights='distance',n_neighbors= 19, metric= 'manhattan', algorithm='brute'))]

#Meta Modelo
meta_model = xgb.XGBClassifier(objective='reg:logistic',max_depth=8,learning_rate=0.3, n_estimators= 93)
```

Accuracy score obtenido: 0.87

Matriz de confusion