FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 1 — Checkpoint 3

[75.06/95.58] Organización de Datos
Segundo cuatrimestre de 2023

Grupo 15

Ayala, Tomás Gabriel - tayala@fi.uba.ar - 105336
Giacobbe, Juan Ignacio - jgiacobbe@fi.uba.ar - 109866
Olaran, Sebastian - solaran@fi.uba.ar - 109410

Docente corrector:
Pereira, Francisco

Índice

1. Introduccion

Para este nuevo checkpoint, hemos decido importar el dataset proveniente del checkpoint 2. Otras nuevas modificaciones que se tuvieron que efectuar fueron para los modelos de SVM, se tuvieron que normalizar los datos. Para ello, hemos utilizado la distancia min max para normalizar los datos en estos casos. Tambien dependiendo de los casos, para la busqueda de hiperparámetros, hemos intercalado entre la utilizacion entre grid search y randomize search

2. Construcción del modelo

- Los hiperparámetros que se optimizaron para **KNN** fueron:
  - **Algorithm**, el cual determina con que algoritmo se determina los k vecinos mas cercanos
  - **n_neighbors**, de cuantos vecinos estan formados los clusters de vecinos
  - **Metric**, que tipo de distancia se utiliza para medir a los vecinos, *euclidean*, *manhattan* o *chebyshev*
  - **Weights**, cuyo valores pueden ser *distance* y *uniform*
- **SVN**: Los hiperparámetros del SVM, los cuales alteran los parametros que conforman las ecuaciones que definen a los vectores que separan el espacio del dataset para categorizar los resultados son:
  - **Gamma**
  - **kernel** Puede tomar tres valores posibles para determinar el tipo de funcion que se utiliza para determinar los vectores que determinan los conjuntos *linear*, *poly*, *rbf (radial o guassiana)*
  - **C** Determina el margen de error de soft limit de manera de evitar el overfitting
- **RF**: Los hiperparámetros que modificamos del RF, los cuales determinan las caracterisiticas de los multiples arboles de decisión son
  - **random_state**: Este parámetro controla la aleatoriedad en el modelo. Un random_state fijo garantiza la reproducibilidad.
  - **n_jobs**: Especifica el número de núcleos de CPU a utilizar durante el entrenamiento. -1 significa utilizar todos los núcleos disponibles.
  - **criterion**: La función para medir la calidad de una división.
  - **min_samples_leaf**: El número mínimo de muestras requeridas en un nodo hoja del árbol de decisión.
  - **min_samples_split**: El número mínimo de muestras requeridas para dividir un nodo interno del árbol de decisión.
  - **n_estimators**: El número de árboles de decisión que se crearán en el Random Forest.
- **XGBoost**: Para este modelo modificamos los siguientes hiperparámetros
  - **learning_rate**: tasa de aprendizaje
  - **max_depth:** máxima profundidad de cada árbol
  - **subsample:** porcentaje de muestras usadas para cada árbol (valor muy bajo, posible underfitting)
  - **colsample_bytree:** porcentaje de features usadas para cada árbol (valores muy alto, posible overfitting)

  - **n_estimators:** cantidad de árboles a construir.

- Para el ensamblaje de modelos del tipo **Voting** se utilizaron los siguientes 3 modelos, debido a su buen tiempo de entrenamiento y variacion de resultados de manera en que estos modelos puedan compensarse entre sí:
  - KNN
  - RF
  - XGBoost
- Para el modelo de ensamble de **Stacking** se utilizaron los siguiientes modelos para emitir las predicciones, y el XGBoost como meta-modelo para entrenar las combinaciones de los modelos base
  - RF
  - SVM
  - KNN
  - **Meta Learner:** XGBoost

3. Cuadro de Resultados

A continuacion podemos observar un cuadro de las metricas de cada mejor predictor de cada modelo:

Modelo	F1-Test	Presicion Test	Recall Test	Accuracy	Kaggle
KNN	0.789	0.773	0.806	0.783	0.742
SVM	0.830	0.842	0.819	0.831	0.805
Random Forest	0.841	0.858	0.825	0.843	0.829
XGBoost	0.870	0.8.68	0.873	0.869	0.804
Voting	0.853	0.865	0.841	0.854	0.820
Stacking	0.865	0.865	0.864	0.863	0.796

El modelo de **KNN** busca agrupar a las observaciones del dataset mediante algun criterio, de manera en que se pueda clasificar a las observaciones en distintas clases. La idea es que estas clases contengan todas algunas caracteristica particular. Este es un modelo muy util para conjunto balanceados, sin embargo es muy sensible a outliers.

El modelo de **SVM** busca tambien clasificar las observaciones del dataset en distintas clases pero en vez de agrupar las observaciones, esta busca dividir el espacio, de manera que todas las observaciones dentro de un espacio pertenezcan a una clase en particular. Hace esta division mediante vectores y dependiendo el kernel se calculan estos vectores de distintas formas.

El modelo de **Random Forest** combina varios modelos (en este caso arboles de decision) para mejorar el rendimiento general y reducir el sobreajuste. Esto se hace mediante el proceso de promedio o votación de los resultados de los árboles individuales.

El model de **XGBoost** que significa *Extreme Gradient Boosting* se basa en el algoritmo de Gradient Boosting y es conocido por su capacidad para generar modelos altamente precisos en una variedad de tareas de clasificación y regresión. XGBoost combina eficazmente múltiples árboles de

decisión para crear un modelo conjunto robusto y optimizado. EL *Boosting* se refiere a una familia de algoritmos de aprendizaje automático que combinan múltiples modelos base más débiles (por ejemplo, árboles de decisión simples) en un modelo fuerte y más preciso. El objetivo es corregir los errores de los modelos base anteriores al dar más peso a las instancias clasificadas incorrectamente.

En el **Voting** la idea principal detrás del método de votación es que, al combinar múltiples modelos, es más probable que se obtenga una predicción precisa, ya que los errores individuales de cada modelo tienden a compensarse entre sí. El método de votación es especialmente útil cuando se utilizan diversos tipos de modelos o algoritmos, ya que cada uno puede capturar diferentes aspectos de los datos o tener fortalezas en diferentes áreas. Al combinar sus predicciones, se puede lograr un rendimiento general mejorado.

El **Stacking** es una técnica de ensamblado avanzada en el aprendizaje automático que se utiliza para combinar las predicciones de múltiples modelos en un nivel superior llamado "meta-modelo". A diferencia de la votación, donde los modelos base emiten predicciones y se realiza una elección mayoritaria o ponderada, en el stacking, se entrena otro modelo (el meta-modelo) para aprender a combinar las predicciones de los modelos base.

4. Matriz de Confusión

A continuación podemos ver la matriz de confusión de nuestro mejor predictor

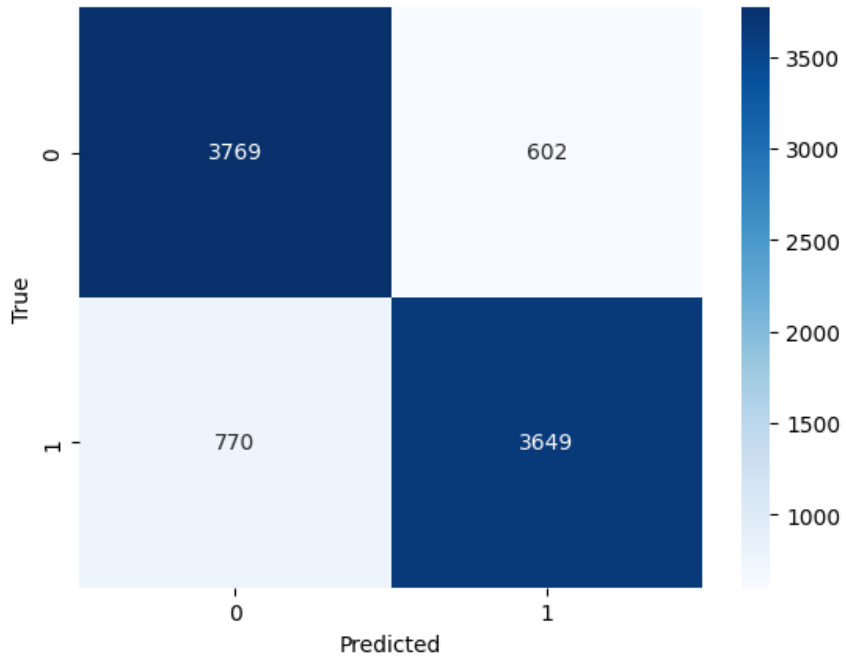

Figura 1: Matriz de predicción del dataset real del train-test split y las predicciones de nuestro mejor modelo entrenado, el Random Forest.

Como podemos observar, y como da a entender nuestro score, cerca de un 83 por ciento de la predicciones son correctas, no podemos observar una mayor o menor tendencia a hacia uno u otro parametro, tanto en los falsos positivos y/o negativos, asi como en los verdaderos positivos y/o negativos.

5. Tareas realizadas

Integrante	Tarea
Tomas Gabriel Ayala	Armado de Reporte y Aramado y entrenamiento de SVM
Sebastian Olaran	Aramado y entrenamiento de XGBoost y Voting y Stacking
Juan Ignacio Giacobbe	Aramado y entrenamiento de KNN y RF