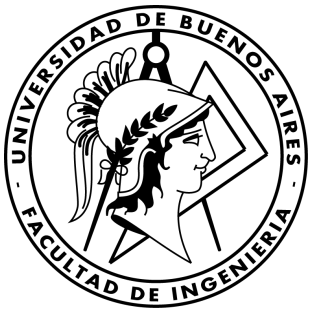


Trabajo Práctico I — Checkpoint III

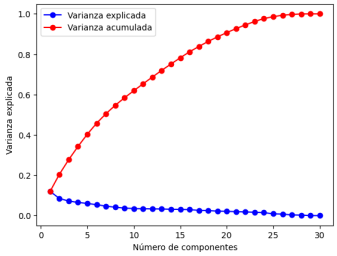
[75.06/95.58] Organizacion de Datos
Primer cuatrimestre de 2023

Alumno	Número de padrón	Email
Masri, Noah	108814	noahmasri19@gmail.com
Ayala, Camila	107440	cayala@fi.uba.ar
Loscalzo, Melina	106571	mloscalzo@fi.uba.ar

En este checkpoint, el objetivo era el de implementar una gran variedad de ensambles para predecir nuestra variable target. Llevamos a cabo ensambles homogéneos, Random Forest (bagging) y XGBoost (boosting), como también ensambles híbridos de Votación y de Stacking.

En primer lugar, creamos un modelo de KNN, al cual le optimizamos los parámetros con Kfolds. Observando las matrices de confusión, pudimos ver que el modelo estaba sobreajustando los datos, ya que era mucho mejor prediciendo sobre los datos de entrenamiento. Es por esto que luego intentamos mejorar las métricas cambiando algunos rangos de parámetros usados a la hora de hacer validación cruzada. Esto dio resultados, pero seguia sobreajustando, por lo que quisimos hacer un ensamble usando la funcion BaggingRegressor, pero esta en mas de 30 minutos no nos devolvio una prediccion.

El segundo tipo de modelo probado fue SVM. Empezamos teniendo dificultades con este, ya que no lograbamos hacer que converja. Luego nos dimos cuenta de que esto sucedia ya que no estabamos escalando los datos; la desviacion estandar era muy alta, ademas de que teniamos demasiadas columnas. Es por esto que realizamos un grafico de varianza explicada para poder realizar una reduccion de la dimensionalidad con PCA.



Creamos entonces, una vez definido el número de componentes principales a mantener luego de usar PCA en base al gráfico, creamos un pipeline que primero estandariza, luego aplica PCA y finalmente nuestro modelo. Comenzamos realizando un modelo básico, con kernel lineal que no dio buenos resultados, y luego obtuvimos mejoras con el kernel polinómico y radial. No pudimos usar Cross Validation para optimizar los hiperparametros óptimos de estos ya que nos seguía diciendo que no convergía, pero aun así probando distintas combinaciones obtuvimos métricas decentes.

El modelo Random Forest no presentó grandes complicaciones a la hora de crearlo, ni aportó métricas mucho mejores que las obtenidas con el único árbol en el checkpoint anterior, que además era de menor profundidad. Luego intentamos mejorar el árbol intentando tener en cuenta más de un hiper parámetro a optimizar, pero esto no dio grandes resultados.

El modelo XGBoost, el cual cuenta con varias métricas de regulación que evitan el overfitting, fue nuestro mejor modelo. Estos son propios de los modelos que utilizan boosting. Comparando los defaults con los obtenidos de la optimización de hiperparametros vemos que se decrementa: el learning rate, por lo que se toman pasos más pequeños y se aumentan: el gamma,por lo cual la poda será más estricta y el lambda,por lo que se regula más el overfitting. Que al optimizar el modelo se cambien de esta manera los hiperparametros nos indica que nuestro dataset es susceptible a el overfitting. Es evidente que estos cambios ayudaron ya que las métricas obtenidas fueron de las más altas, que a su vez no estaban sobreajustando.

Los modelos de ensamble híbrido consisten en crear un estimador fuerte en base a varios débiles. Los modelos base usados son random forest, support vector machine y k-nearest neighboors.

En el modelo de stacking los modelos base hacen una estimación relativamente pobre, es decir, no estan mejorados, con el que se crea un nuevo dataset y luego por medio del meta-modelo se produce la predicción final. Como meta-modelo usamos un árbol, ya que nuestro problema lo

requeria, y el modelo de redes neuronales no estaba a nuestro alcance. Las metricas resultantes fueron buenas pero la diferencia entre las predicciones sobre los set de test y train fue bastante marcada, indicando overfitting. Este modelo se tomo un largo rato en correr a su vez.

En el modelo de voting elegimos que el tipo de votación sea soft, ya que esperamos que los votos no valgan para todos los modelos y todas las predicciones lo mismo, ya que sabemos que hay modelos mas precisos que otros. Los resultados fueron similares a stacking, buenas metricas pero una gran brecha entre test y train, indicando overfitting.