FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 1 — Reservas de Hotel

Organizacion de Datos
Curso Rodriguez
Primer cuatrimestre de 2023

Alumno	Padrón	gitHub
Camila Gonzalez	105661	c-gonzalez-a
Eduardo Martín Bocanegra	106028	martinboca
Mateo Cabrera	108118	m-cabrerar

1. Introducción

El objetivo de esta tercer entrega fue el utilizado de modelos de ensamble para la predicción de datos en nuestro data set. Para ello, primero vamos a entrenar modelos para ensamblar, K Nearest Neighbors, Support Vector Machine y Random Forest. Además, entrenaremos los siguientes ensambles:

- XGBoost, ensamble de árboles de decisión.
- Voting, ensamble híbrido que utlizará los modelos entrenados anteriormente.
- Stacking, ensamble híbrido que utiliazará los tres modelos, y regresión logística como meta-modelo.

2. Entrenado de modelos

Durante el desarrollo de esta entrega, exploramos distintas formas de entrenar el modelo con el objetivo de encontrar el algoritmo de entrenamiento que lograra la mayor precisión en nuestras predicciones. Inicialmente, utilizamos el algoritmo de k-Nearest Neighbors (KNN) con los parámetros estándar, lo cual nos proporcionó una precisión del 0.73 en nuestras predicciones. Sin embargo, buscamos mejorar el rendimiento del modelo y optimizamos los hiperparámetros utilizando la técnica de K-fold Cross Validation. Esta optimización nos permitió aumentar la precisión del modelo a un 0.82.
Luego, nos enfocamos en el algoritmo de Support Vector Machines (SVM) y realizamos experimentos con diferentes configuraciones de parámetros. Desafortunadamente, todas las combinaciones probadas tuvieron precisiones bajas y resultados insatisfactorios. Posteriormente exploramos ir variando los kernels, como el lineal, polinómico y sigmoidal. Después de estas pruebas, concluimos que el mejor modelo SVM tenía una precisión del 0.79.
Finalmente, nos adentramos en el algoritmo de Random Forest. Comenzamos utilizando hiperparámetros arbitrarios y obtuvimos resultados decentes. Sin embargo, buscamos mejorar aún más el rendimiento del modelo y optimizamos los hiperparámetros, observando que, aunque la diferencia no era significativa, el modelo con hiperparámetros optimizados mostraba la mayor precisión.

3. Ensamble

3.1. XGBoost

El algoritmo XGBoost crea una secuencia de modelos que se entrenan uno después del otro. Cada modelo aprende y mejora de los errores de sus anteriores.

Para usar XGBoost buscamos los mejores parametros para nuestro dataset, usando HalvingGridSearchCV. Incluimos un amplio rango de parametros con los que probar el modelo, por lo que la busqueda tardo varias horas. Finalmente dimos con los mejores parámetros para XGBoost en nuestro dataset, logrando una precision, un recall y un F1-Score mayores a 0.91

3.2. Modelos Híbridos

En cuanto a los ensambles híbridos, vamos a utilizar Voting y Stacking aprovechando los modelos entrenados.
El modelo de Voting tendrá en cuenta la clasificación de los tres modelos. Con Hard Voting, clasificará la observación en la categoría con más votos, en cuanto a soft voting, le da mayor importancia a los modelos que voten con más seguridad.

Por úlitmo, el clasificador Stacking, este utilizará los mismos modelos como base, pero además, tendrá un meta-modelo, encargado de seleccionar qué modelo base se utilizará para la predicción de cada observación nueva.