Room Occupancy Stimation

Daniel Alejandro Yepes Mesa alejandro.yepes@udea.edu.co
Juan Felipe Santa Ospina juan.santa@udea.edu.co
Cristian David Tamayo Espinosa cristian.tamayoe@udea.edu.co
Dpto. of Systems Engineering and Computer Science
Universidad de Antioquia

Descripción del Contexto del Problema y Utilidad de la Solución Basada en ML

Estimar con precisión el número de ocupantes en una sala se vuelve imprescindible, ya que el consumo energético ejerce un papel determinante en el gasto global, siendo los sistemas de calefacción, ventilación, aire acondicionado e iluminación sus principales responsables. Por lo tanto, es primordial definir una estrategia efectiva para optimizar el uso de estos sistemas y ajustarlos dinámicamente al número de ocupantes en un espacio, para de este modo además de optimizar el uso de estos, se logre estimar correctamente en función de las necesidades del espacio y así obtener un ahorro energético. Esto se aborda utilizando sensores ambientales no intrusivos, estos sensores ofrecen una alternativa que respeta la privacidad para recopilar datos relevantes sobre el entorno de una sala, y así evitar acudir a otros metodos intrusivos como cámaras, que pueden generar preocupaciones con respecto a la privacidad. Por lo anterior, la utilidad de desarrollar una solución de Machine Learning para este problema es multifacética, ya que al ajustar dinámicamente los sistemas según la presencia real se permite una considerable eficiencia energética lo que mejora simultáneamente el confort de los ocupantes, esto a su vez también impulsa la automatización inteligente de edificios y facilita un valioso análisis para la optimización del uso del espacio.

Descripción de la Composición de la Base de Datos

El dataset utilizado para este proyecto es el "Room Occupancy Estimation Dataset" del UCI Machine Learning Repository [1].

Número de Muestras: El dataset contiene 10,129 instancias. Cada instancia representa un conjunto de lecturas de sensores tomadas en un intervalo de 30 segundos durante un período de 4 días.

Número de Variables: Hay un total de 19 variables, de estas, 18 son características (features) y 1 es la variable objetivo (target).

Significado de las Variables:

Date (Fecha): Fecha de la medición (YYYY/MM/DD). Tipo: Fecha.

Time (Hora): Hora de la medición (HH:MM:SS). Tipo: Fecha/Hora.

S1_Temp a S4_Temp: Lecturas de temperatura de 4 sensores distintos (S1 a S4). Tipo: Continuo. Unidades: Grados Celsius (°C).

S1_Light a S4_Light: Lecturas de intensidad lumínica de los mismos 4 sensores. Tipo: Entero. Unidades: Lux.

S1_Sound a S4_Sound: Lecturas del nivel de sonido (salida del amplificador leída por ADC) de los mismos 4 sensores. Tipo: Continuo. Unidades: Volts.

S5_CO2: Nivel de dióxido de carbono medido por el sensor S5. Tipo: Entero. Unidades: Partes por millón (PPM).

S5_CO2_Slope: Mide qué tan rápido está cambiando el nivel de CO2. Ayuda a detectar más rápidamente la presencia de personas. Es un valor numérico continuo.

S6_PIR, S7_PIR: Valores de los sensores infrarrojos pasivos que detectan movimiento. Tipo: Binario (0 o 1, para detección de movimiento). Aunque S7_PIR está listado como entero en UCI su función es binaria.

Room_Occupancy_Count (Variable Objetivo): Número real de ocupantes en la sala. Rango: 0 a 3 personas. Tipo: Entero.

Según la descripción del dataset en UCI, no hay valores faltantes, y dado que no hay datos faltantes, no se requiere una estrategia de imputación en esta etapa.

Adicionalmente cabe destacar el desbalance notorio de clases, lo cual se tendra en cuenta en las implementaciones posteriores:

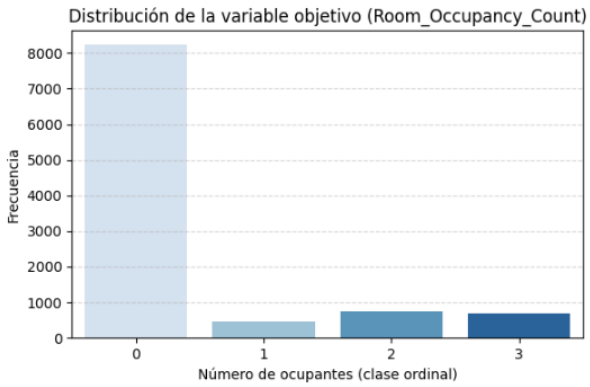

Imagen 1: Distribución de clases

Codificación de Variables:

Date y Time: Actualmente están como cadenas de texto o tipos de fecha/hora. Para su uso en modelos, podrían transformarse en características numéricas más útiles como por ejemplo hora del día, día de la semana, etc.

Variables de Sensores (Temp, Light, Sound, CO2, CO2_Slope): Son numéricas, y pueden usarse directamente, aunque la normalización o estandarización será considerada.

Variables PIR (S6_PIR, S7_PIR): Son inherentemente binarias (0 para no movimiento, 1 para movimiento).

Variable Objetivo (Room_Occupancy_Count): Es una variable entera que representa categorías ordenadas (0, 1, 2, 3).

Tipo de Configuración o Paradigma de Aprendizaje y su Justificación

Para abordar el problema de estimación de ocupantes en la sala, el equipo ha decidido que el paradigma de aprendizaje más apropiado es el de regresión ordinal con datos temporales , la justificación de esta elección es debido a la naturaleza ordenada de la variable a predecir, Room_Occupancy_Count, ésta variable representa etiquetas con un orden intrínseco y significativo (0, 1, 2, 3). por lo que en este problema no es lo mismo equivocarse entre dos niveles consecutivos que entre niveles alejados, por lo que la regresión ordinal es el paradigma diseñado específicamente para este escenario. Adicional se debe tener en cuenta la dependencia temporal de los datos, ya que el dataset proporciona mediciones tomadas cada 30 segundos y esto implica que las observaciones no son independientes entre sí, ya que el número de ocupantes en un momento dado está probablemente relacionado con las mediciones de los momentos anteriores.

Estado del arte

Empezamos analizando el artículo, **Room Occupancy Prediction: Exploring the Power of Machine Learning and Temporal Insights [2]**. Este trabajo propone una solución al problema de estimar la ocupación de habitaciones utilizando diversos modelos de clasificación. Los autores trabajan directamente con el conjunto de datos "Room Occupancy Estimation" del UCI [1] y evalúan una serie de algoritmos de aprendizaje automático para predecir la cantidad de ocupantes en una habitación, a partir de sensores de temperatura, luz, sonido, CO_2 y movimiento. Paradigma de aprendizaje: Supervisado, clasificación multiclase. Las técnicas utilizadas fueron: Regresión logística multinomial, Análisis Discriminante Lineal (LDA), Máquinas de Vectores de Soporte (SVM), Random Forest, XGBoost, LightGBM y Perceptrón Multicapa (MLP). Las metodología de validación fueron: validación cruzada de 5 pliegues junto con ajuste de hiperparámetros utilizando búsqueda en malla. En el apartado de métricas de evaluación tenemos Precisión (accuracy), AUC ponderado y análisis SHAP para interpretación de resultados. El modelo que obtuvo el mejor desempeño fue Random Forest, alcanzando una precisión del 98.7%. Los autores destacan que las variables más relevantes fueron las de iluminación (especialmente S1_Light y S2_Light), lo que sugiere que la intensidad lumínica está estrechamente relacionada con la presencia de personas. Además, el análisis SHAP permitió comprender cómo cada variable contribuía a la predicción, facilitando la interpretación del modelo y resaltando la importancia de ciertos sensores ambientales. A pesar de no modelar explícitamente la temporalidad, se observó que la alta frecuencia de muestreo de los datos capturaba implícitamente patrones temporales, mejorando la capacidad predictiva de modelos tradicionalmente no secuenciales.

Continuamos con **Ubiquitous Multi-Occupant Detection in Smart Environments [3].** Este estudio se enfoca en la detección de múltiples ocupantes en entornos inteligentes utilizando sensores no invasivos y modelos de aprendizaje profundo que capturan la dinámica temporal de las señales. Aunque no se menciona explícitamente que se utilizó la misma base de datos del UCI [1], la estructura de datos, tipo de sensores y objetivo del estudio son altamente similares. El paradigma de aprendizaje predominante es Supervisado, con enfoque secuencial (serie de tiempo). Las técnicas utilizadas fueron los modelos tradicionales como SVM, QDA y k-NN, y modelos de aprendizaje profundo como MLP, LSTM, GRU y BiGRU.

Ahora, las metodologías de validación utilizadas son las de separación entre conjuntos de entrenamiento y prueba mediante ventanas deslizantes, permitiendo evaluar el rendimiento en secuencias temporales.Ahora para evaluar el modelo usaron métricas como: Precisión, F1-score, recall y exactitud. Las métricas F1-score y recall permiten evaluar el rendimiento de clasificación considerando el balance entre clases. El modelo BiGRU fue el más destacado, obteniendo un F1-score superior al 90% en la tarea de estimar múltiples ocupantes. Este resultado demuestra la ventaja de utilizar modelos secuenciales en problemas donde los datos presentan dependencia temporal. El trabajo resalta que, al modelar la información contextual a través del tiempo, se mejora significativamente la capacidad del sistema para detectar no solo la presencia, sino también el número exacto de personas en una habitación. Los modelos tradicionales mostraron buen desempeño, pero fueron superados por las redes neuronales recurrentes en tareas de mayor complejidad.

Dentro de las propuestas más interesantes encontradas en la literatura, se destaca el trabajo titulado Room Occupancy Prediction from Temperature Data with Deep Convolutional Neural Networks publicado en IJAMEC (2025). Este estudio resulta especialmente atractivo por su enfoque minimalista y eficiente: predecir la cantidad de personas en una habitación utilizando únicamente sensores de temperatura, sin necesidad de cámaras ni sensores intrusivos. Esta propuesta se alinea directamente con aplicaciones reales como la activación inteligente de sistemas de climatización, donde conocer la ocupación permite optimizar el encendido o apagado de aires acondicionados para ahorrar energía y mejorar el confort.

Para ello, los autores transformaron las lecturas de cuatro sensores de temperatura en imágenes RGB de 28×28 píxeles, donde cada sensor representa una zona de la imagen. Estas imágenes fueron la entrada para una red neuronal convolucional profunda (CNN), que fue entrenada para identificar patrones térmicos que correspondieran a diferentes niveles de ocupación.

El conjunto de datos empleado proviene de la base abierta de la UCI, y fue preprocesado cuidadosamente para eliminar días sin ocupación y suavizar el ruido temporal. La validación del modelo se realizó con un esquema hold-out, separando los datos en entrenamiento, validación y prueba, sin mezclar temporalidades, lo que garantizó una evaluación realista del rendimiento.

La métrica principal fue la accuracy, logrando una precisión del 93,33% en el conjunto de prueba. El modelo fue especialmente preciso para detectar habitaciones vacías o con baja ocupación, aunque mostró más errores en transiciones abruptas —cuando el número de personas cambiaba rápidamente—, probablemente porque los cambios térmicos no siempre se reflejan de forma inmediata.

Este estudio demuestra cómo es posible aprovechar datos simples y económicos, como los de temperatura ambiente, para realizar predicciones útiles en contextos reales de automatización y eficiencia energética. Su enfoque ofrece un punto de partida valioso para futuros sistemas de conteo de personas que busquen ser económicos, no intrusivos y fáciles de implementar en entornos reales.

Otro artículo que realmente nos llamó la atención fue el publicado por Kumari en 2023, titulado Indoor Occupancy Detection and Counting via Boosting and Sensor Fusion. Lo interesante de este trabajo no es solo que lograron saber si una habitación estaba ocupada, sino que también pudieron estimar cuántas personas había dentro. Y todo eso sin usar cámaras ni micrófonos. Solo sensores ambientales. Nada invasivo.

Trabajaron con datos reales recolectados a partir de sensores de temperatura, luz, dióxido de carbono, sonido y movimiento (PIR). La clave estuvo en cómo integraron todos esos datos. En vez de depender de un solo tipo de sensor, fusionaron la información para tener una imagen más completa de lo que ocurre dentro del espacio. A esto sumaron algoritmos de aprendizaje automático, desde técnicas más tradicionales como regresión logística hasta modelos más sofisticados como redes neuronales, además de métodos de boosting como LightGBM y XGBoost.

Pero no se quedaron ahí. También aplicaron una técnica inspirada en la inteligencia colectiva de los enjambres: el Particle Swarm Optimization (PSO). Esta herramienta les permitió ajustar los parámetros de sus modelos y encontrar las combinaciones de sensores más efectivas. O sea, el sistema no solo clasificaba, también aprendía a optimizarse con el tiempo.

¿Y los resultados? Bastante impresionantes. En la tarea de detectar si una habitación estaba ocupada o no, algunos modelos alcanzaron casi un 99% de precisión y F1-score. Es decir, se equivocaban muy poco. Incluso al contar cuántas personas había (lo cual es mucho más complicado), lograron superar el 95% de precisión. Y todo esto con sensores accesibles y económicos.

Este tipo de desarrollos abre un montón de posibilidades para los edificios inteligentes. Por ejemplo, se podría activar la climatización o las luces solo cuando haya gente realmente en el espacio. Esto no solo optimiza el consumo energético, también mejora la experiencia de quienes lo habitan. En definitiva, el estudio demuestra que al combinar múltiples sensores con modelos bien afinados, se pueden lograr resultados sorprendentemente buenos en contextos cotidianos.

4.1 Configuración experimental

Metodología de Validación:

El conjunto de datos Occupancy_Estimation.csv posee una naturaleza temporal. Para asegurar una evaluación robusta que evite la fuga de información del futuro al pasado, se empleó una estrategia de validación basada en series temporales.

Se utilizó TimeSeriesSplit de Scikit-learn para dividir los datos secuencialmente. Se iteró sobre los folds generados hasta encontrar uno específico que cumpliera con la condición crítica de tener todas las clases objetivo (0, 1, 2 y 3 ocupantes) presentes tanto en el conjunto de entrenamiento como en el de prueba. El fold seleccionado se utilizó como la división final entre entrenamiento (X_train, y_train) y prueba (X_test, y_test).

Para la selección de características (usando SFS) y la optimización de hiperparámetros (usando GridSearchCV), se aplicó una validación cruzada interna sobre el conjunto de entrenamiento (X_train). Se utilizó TimeSeriesSplit(n_splits=3) para mantener la integridad temporal durante la optimización, asegurando que en cada iteración de la validación cruzada, los datos de entrenamiento precedieran temporalmente a los de validación.

6. Modelos Evaluados y Malla de Hiperparámetros:

Se evaluaron seis modelos de aprendizaje automático, cubriendo diferentes paradigmas:

- Regresión Logística (Paramétrico/Regresión):
- K-Nearest Neighbors (k-NN) (No paramétrico):
- Random Forest
- XGBoost
- Support Vector Classifier (SVC)
- Multi-Layer Perceptron (MLP) (Red Neuronal)

Se detalla la malla de hiperparámetros utilizada:

Modelo	Hiperparámetro	Valores Analizados
LogisticRegression	C (Inverso de la fuerza de regularización)	[0.01, 0.1, 1, 10]
KNeighborsClassifier	n_neighbors	[3, 5, 7, 9]
RandomForestClassifier	n_estimators	[50, 100, 200]
	max_depth	[None, 5, 10]
XGBClassifier	n_estimators	[50, 100, 200]
	learning_rate	[0.01, 0.1]
	max_depth	[3, 5, 7]
MLPClassifier	hidden_layer_sizes	[(50,), (100,), (50, 50)]
	alpha (Regularización L2)	[0.0001, 0.001]
SVC	C	[0.1, 1, 10]
	gamma	['scale', 'auto']

tabla 1: malla de hiperparámetros utilizada

7. Métricas de desempeño:

Se implementaron las siguientes métricas:

Accuracy: Aunque no es la métrica principal debido a su sensibilidad al desbalance de clases se incluye como un indicador general sobre la proporción de predicciones correctas totales.

F1 Ponderado: Proporciona un promedio del F1-Score por clase, ponderado por el número de muestras en cada una. Es ideal para problemas con desbalance de clases como el presente, ya que ofrece una medida equilibrada de la precisión y la exhaustividad del modelo en su conjunto.

F1 Macro: Calcula el F1-Score para cada clase de forma independiente y luego promedia los resultados, dando el mismo peso a cada clase sin importar su tamaño. Es útil para detectar si el modelo tiene un bajo rendimiento en las clases minoritarias.

Mean Absolute Error (MAE): Al ser un problema ordinal, el MAE es una métrica de regresión muy informativa. Mide la distancia promedio entre la clase predicha y la clase real, penalizando los errores en función de su magnitud.

Adjacent Accuracy: Es una métrica personalizada

que evalúa la proporción de predicciones que son correctas o se desvían en una sola clase.

Kappa de Cohen: Mide el acuerdo entre las predicciones y los valores reales, corrigiendo el efecto del azar.

Kendall Tau: esta sería la métrica más adecuada por diversas razones, una de ellas es que mide la correlación ordinal entre las predicciones y las verdaderas etiquetas y penaliza más los errores que rompen el orden relativo entre clases. Además es compatible con datos temporales, al basarse en rangos y concordancias evalúa si el modelo mantiene una secuencia ordinal consistente a través del tiempo. Adicionalmente, a diferencia de métricas como f1_macro, accuracy o kappa, no depende directamente de la frecuencia de cada clase.

Resultados sólidos: en el fold seleccionado (con todas las clases presentes), se obtuvo un valor promedio de Kendall Tau ≈ 0.985, lo cual refleja una alta concordancia ordinal entre predicción y realidad.

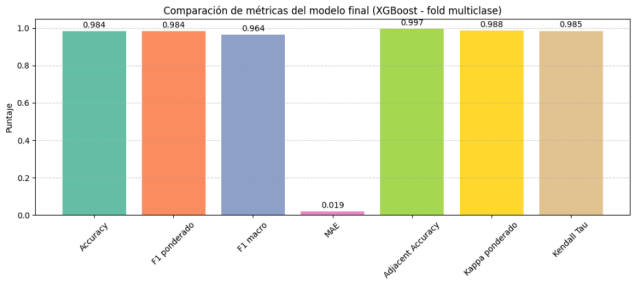

imagen 1: Comparación de métricas del modelo final (XGBoost - fold multiclase)

La métrica Kendall Tau proporciona una evaluación coherente con la naturaleza ordinal y secuencial de este problema. Será adoptada como métrica principal en las fases posteriores de validación y selección de modelos, asegurando que no solo se prediga correctamente, sino que también se respete el patrón evolutivo en el tiempo.

También es de utilidad analizar la siguiente matriz de confusión

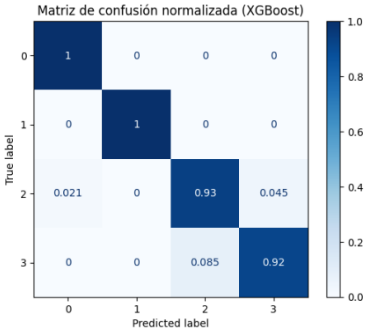

imagen 2: matriz de confusión normalizada XGBoost

Clases 0 y 1: 100 % de acierto.

Clase 2: 93 % de predicciones correctas; 4.5 % se confunden con 3 y 2.1 % con 0.

Clase 3: 92 % de acierto; 8.5 % se confunden con 2.

La normalización elimina el sesgo de frecuencia y muestra que todas las confusiones ocurren entre clases adyacentes, es decir, el modelo tiende a cometer "errores cercanos" al número real de ocupantes.

4.2 Resultados del entrenamiento de los modelos

8.

Con todas las variables:

	Kendall Tau	MAE	F1 ponderado
MLP	1.000000	0.000000	1.000000
SVM (RBF)	1.000000	0.000000	1.000000
XGBoost	0.985035	0.018957	0.983924
Random Forest	0.976383	0.023104	0.984378
k-NN (k=5)	0.793589	0.162322	0.904844

tabla: 1 desempeño de cada modelo

Esta evaluación inicial establece a XGBoost y Random Forest como los candidatos más prometedores, ya que combinan un rendimiento excepcional con una mayor probabilidad de generalización. Además, justifica la necesidad de una fase de selección de características (como se realiza en los puntos 9 y 10 del notebook) para eliminar la redundancia, reducir la complejidad y construir un modelo final mas robusto.

Optimización de Hiperparámetros:

Siguiendo la metodología descrita en la sección 4.1, se aplicó una búsqueda de hiperparámetros mediante GridSearchCV sobre el subconjunto de 8 características identificado por Selección Secuencial Ascendente (este método de Selección Secuencial Ascendente, se explicará en mas detalle en el numeral 9). La evaluación se realizó utilizando validación cruzada temporal (TimeSeriesSplit con 3 splits) y se optimizó para la métrica Kendall Tau.

La Tabla resume el mejor desempeño de cada modelo durante esta fase de validación cruzada, mostrando el puntaje promedio de Kendall Tau y los mejores hiperparámetros encontrados.

	Modelo	Mejor_Params	Mejor_KendallTau_CV
3	XGBClassifier	{'model__learning_rate': 0.1, 'model__max_dept...	0.6624
2	RandomForestClassifier	{'model__max_depth': 5, 'model__n_estimators':...	0.6604
5	SVC	{'model__C': 10, 'model__gamma': 'scale'}	0.6598
1	KNeighborsClassifier	{'model__n_neighbors': 3}	0.6102
0	LogisticRegression	{'model__C': 10}	0.5344
4	MLPClassifier	{'model__alpha': 0.001, 'model__hidden_layer_s...	0.5094

tabla 2: Resumen de la búsqueda de hiperparámetros con validación cruzada temporal

Durante la validación cruzada, XGBoost, Random Forest y SVC mostraron el mejor rendimiento, con puntajes de Kendall Tau muy cercanos entre sí. Esto indica que estos modelos, con los hiperparámetros adecuados, son capaces de aprender patrones ordinales de manera robusta incluso en los folds internos más desafiantes de la serie temporal.

Evaluación Final en el Conjunto de Prueba (Holdout)

Los modelos con los mejores hiperparámetros identificados en la sección anterior se evaluaron finalmente en el conjunto de prueba (holdout).

	Modelo	KendallTau	MAE	AdjacentAccuracy
0	LogisticRegression	0.9665	0.0806	0.9994
4	MLPClassifier	0.9650	0.1102	0.9994
3	XGBClassifier	0.9644	0.0782	1.0000
2	RandomForestClassifier	0.9608	0.1191	1.0000
5	SVC	0.9549	0.1037	0.9988
1	KNeighborsClassifier	0.9251	0.1321	0.9953

tabla 3: Desempeño final de los modelos optimizados en el conjunto de prueba.

En la evaluación final, todos los modelos mostraron un rendimiento excepcional. Notablemente XGBoost mantuvo su posición de liderazgo, y también logró el Error Absoluto Medio (MAE) más bajo (0.0782) y una Adjacent Accuracy perfecta (1.0000).

5. Reducción de dimensión:

Para abordar la reducción de dimensión se realizó un análisis en dos etapas, donde primero se hizo un examen individual de cada característica para identificar candidatas a ser eliminadas, y luego una selección automática utilizando un método de búsqueda secuencial para encontrar el subconjunto óptimo de variables.

5.1. Selección de características

9. Tras combinar las métricas univariantes (ANOVA_F, Mutual_Info, Kendall_Tau), la importancia del modelo XGBoost y el VIF, se obtienen las siguientes conclusiones:

Variables más relevantes:

S1_Light y S3_Light lideran todas las métricas: son los predictores univariantes más discriminativos y aportan > 60 % de la importancia en XGBoost.

S7_PIR y S2_Light ocupan el tercer y cuarto puesto, mostrando buena correlación ordinal (Tau > 0.57) y moderada contribución al modelo.

Variables de relevancia media:

Sensores de temperatura (S2_Temp, S1_Temp, S5_CO2_Slope, S5_CO2) y sonido (S2_Sound, S4_Sound, S1_Sound, S3_Sound), con Kendall Tau entre 0.39 y 0.67, aportan información complementaria pero menor.

Colinealidad (VIF):

Aunque S1_Temp y S3_Temp muestran VIF altos (> 10), indican redundancia entre sensores de temperatura.

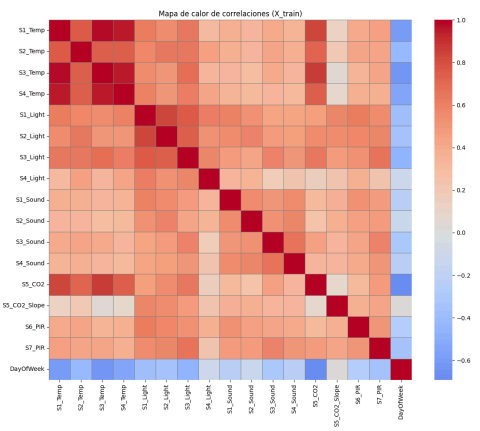

imagen 2: correlación de variables

Candidatas a eliminación por baja información:

S4_Temp: baja en ANOVA_F, Mutual_Info y Kendall_Tau, aporta < 0.2 % en XGBoost.

DayOfWeek: correlación ordinal nula (Tau < 0), importancia en el modelo prácticamente cero.

10. Para la **selección de características** se aplicó el método de Selección Secuencial Ascendente .

Se eligió la correlación de **Kendall Tau** como la función criterio para guiar el algoritmo de selección. Esta métrica es la más adecuada para este problema por tres razones clave:

- Evalúa si las predicciones mantienen el orden correcto de las clases (ej., 0 < 1 < 2), lo cual es fundamental en un problema de regresión ordinal.
- Es menos sensible al desbalance de clases presente en el dataset en comparación con métricas como F1-score o Accuracy.
- Mide la consistencia del orden a lo largo del tiempo, alineándose con la naturaleza secuencial de los datos.

Evaluación y Resultados:

Se aplicó SFS a los dos mejores modelos identificados previamente (XGBoost y Random Forest), utilizando el fold temporal validado que contenía todas las clases. Los resultados se resumen en la siguiente tabla:

	Modelo	N° Features	Reducción (%)	Kendall Tau	Features
1	8	XG Boost	52.9	1.0000	S1_Temp, S1_Light, S2_Light, S3_Light, S1_Sound, S2_Sound, S6_PIR, DayOfWeek
0	8	Random Forest	52.9	0.9969	S1_Temp, S2_Temp, S4_Temp, S1_Light, S3_Light, S4_Light, S3_Sound, DayOfWeek

tabla 3: Comparación entre subconjuntos seleccionados (SFS)

Ambos modelos logran una reducción de características del 52.9%, pasando de 17 a solo 8 variables.

El modelo XGBoost, utilizando el subconjunto seleccionado por SFS, alcanzó un Kendall Tau perfecto de 1.0, superando ligeramente a Random Forest (0.9969). Esto demuestra que no solo no se perdió rendimiento, sino que se eliminó el ruido de tal manera que el modelo pudo capturar perfectamente la estructura ordinal en el conjunto de prueba.

5.2. Extraccion de caracteristicas:

En esta etapa se aplicó Análisis de Componentes Principales (PCA) con el objetivo de reducir la complejidad del sistema predictivo sin comprometer su capacidad de generalización. Se aplicó StandardScaler a las 18 variables predictoras y posteriormente se utilizó PCA con un umbral del 95% de varianza explicada acumulada, lo que resultó en la selección de 10 componentes principales, equivalentes a una reducción del 44.44% en la dimensionalidad.

Para evaluar el impacto de esta transformación, se entrenaron nuevamente los dos modelos con mejor rendimiento en la sección 4: Random Forest y XGBoost. Se utilizó TimeSeriesSplit para conservar la secuencia temporal de los datos, seleccionando un fold de prueba que incluyera todas las clases posibles.

Los resultados obtenidos se resumen a continuación:

Modelo	Variables Originales	Componentes PCA	Accuracy (%)	F1-score (%)	Reducción
Random Forest	18	10	97.99	92.28	44.44%
XGBoost	18	10	93.48	78.65	44.44%

tabla 4: Resultados de la transformación y uso de PCA

Los resultados evidencian que Random Forest mantuvo un rendimiento excelente a pesar de la reducción de variables, mientras que XGBoost experimentó una caída significativa en su capacidad de generalización, particularmente reflejada en la métrica F1-score. Esto puede deberse a que XGBoost depende más de la estructura específica de las variables originales, mientras que Random Forest parece más robusto frente a transformaciones lineales como las que genera PCA.

Se concluye que, en este caso, la aplicación de PCA es favorable para modelos como Random Forest, ya que logra reducir la dimensionalidad y el costo computacional sin sacrificar rendimiento. Sin embargo, no todos los algoritmos se benefician por igual, lo que sugiere la importancia de evaluar esta técnica de forma diferenciada por modelo.

Con el fin de comprender de manera más detallada el rendimiento y el comportamiento de los modelos tras la transformación de los datos mediante PCA, se incorporaron matrices de confusión como herramienta complementaria de evaluación.
Estas matrices permiten visualizar no solo el nivel de aciertos globales, sino también los patrones específicos de error, es decir, cuáles clases fueron confundidas entre sí y con qué frecuencia.

A continuación, se presentan las matrices correspondientes a los dos modelos con mejor desempeño en este trabajo: Random Forest y XGBoost, evaluados sobre el conjunto de prueba con los datos reducidos mediante PCA (conservando el 95% de la varianza original).

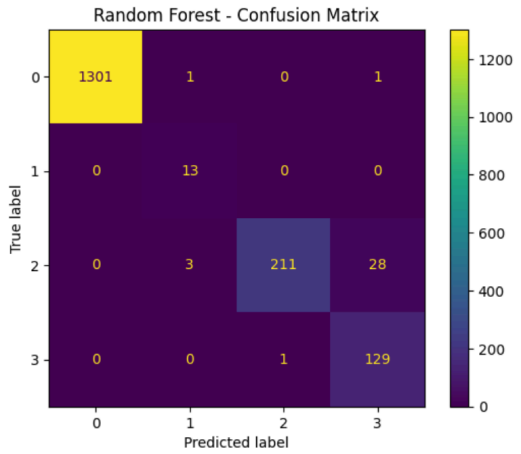

imagen 3: matriz de confusión del random forest en PCA.

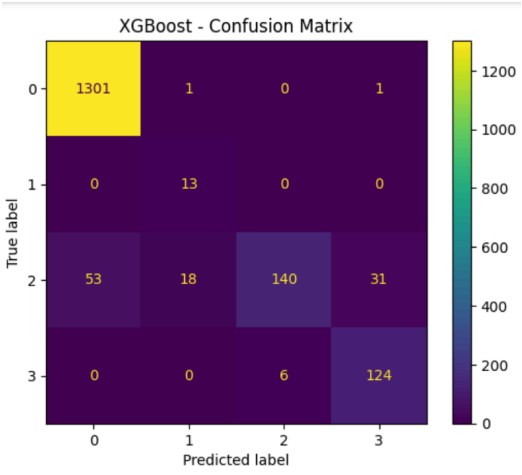

imagen 4: matriz de confusión de XGBoost en PCA.

Las anteriores figuras muestran las matrices de confusión correspondientes a los modelos Random Forest y XGBoost utilizando los datos transformados por PCA.

En el caso de Random Forest, se observa un comportamiento altamente preciso en las clases 0, 1 y 3. La única fuente significativa de error se encuentra en la clase 2, la cual es ocasionalmente confundida con la clase 3 (28 instancias). Sin embargo, todos los errores ocurren entre clases adyacentes, lo cual es coherente con la naturaleza del problema y evidencia una clasificación estable y razonable.

En contraste, el modelo XGBoost muestra una notable confusión en la clase 2, que fue incorrectamente clasificada como 0 (53 veces), 1 (18 veces) o 3 (31 veces). Esto indica una mayor dependencia del modelo a la estructura original de las variables, lo cual explicaría la caída en su desempeño (F1-score) tras aplicar PCA.

La comparación visual refuerza la conclusión de que Random Forest no solo mantuvo su rendimiento tras la reducción de dimensionalidad, sino que también logró una clasificación limpia y precisa, mientras que XGBoost perdió capacidad de discriminación especialmente en clases medias.

Discusión y conclusiones:

Durante el desarrollo de este proyecto, buscamos construir una solución que no solo fuera precisa, sino también eficiente y comprensible desde el punto de vista computacional. A lo largo del proceso, aplicamos distintos modelos de clasificación, evaluamos métricas clave y exploramos técnicas como la selección de características y la reducción de dimensionalidad con PCA. Todo esto nos permitió tener una visión más completa del problema de estimación de ocupación en ambientes interiores.

Después de probar diferentes algoritmos, decidimos enfocarnos en los dos modelos que obtuvieron los mejores resultados: Random Forest y XGBoost. Al aplicar PCA, logramos reducir el número de variables de 18 a 10 componentes principales, lo que representa una reducción del 44.44% en la dimensionalidad. A pesar de esta reducción, el modelo Random Forest mantuvo un rendimiento muy alto, con una precisión del 97.99% y un F1-score macro de 92.28%. Por otro lado, XGBoost sufrió una mayor disminución en su desempeño, lo cual evidencia que no todos los modelos toleran igual la transformación de variables que propone PCA.

Al analizar las matrices de confusión, encontramos que la clase 2 fue la más difícil de predecir para XGBoost, posiblemente porque sus características se parecen a las de otras

clases, especialmente en escenarios intermedios de ocupación. Este tipo de errores no se evidencian fácilmente con una sola métrica global como el accuracy, por eso consideramos valioso haber incluido este tipo de análisis visual. En cambio, Random Forest logró un desempeño mucho más equilibrado entre clases, lo que refuerza su robustez.

También vale la pena reconocer algunas limitaciones de nuestro trabajo. Por ejemplo, no ajustamos los hiperparámetros de los modelos con técnicas más exhaustivas como GridSearchCV, y trabajamos con un dataset que proviene de un entorno físico específico. Esto puede afectar la generalización del modelo en otros contextos. Además, PCA, al ser una técnica no supervisada, podría haber eliminado variables importantes para la clasificación sin que lo notáramos directamente.

A pesar de esto, creemos que el sistema propuesto tiene aplicaciones muy interesantes. Con una buena calibración y reentrenamiento, este tipo de solución puede servir para automatizar iluminación, ventilación o control de aforo en espacios cerrados como oficinas, aulas, edificios inteligentes o centros comerciales. Nos parece muy valioso que un modelo relativamente simple, alimentado por sensores básicos, pueda dar resultados tan sólidos en una tarea como esta.

En conclusión, este proyecto nos permitió aplicar conocimientos clave de aprendizaje automático y análisis de datos, y además nos ayudó a entender mejor cómo las decisiones que tomamos sobre los datos (como reducir características) impactan directamente en el comportamiento de los modelos. Quedan muchas puertas abiertas para seguir optimizando, pero los resultados alcanzados hasta ahora nos dejan satisfechos y motivados para seguir aprendiendo.

Bibliografía

[1] A. Singh and S. Chaudhari. "Room Occupancy Estimation," UCI Machine Learning Repository, 2018. [Online]. Disponible en: https://doi.org/10.24432/C5P605.

[2] S. Mao, Y. Yuan, Y. Li, Z. Wang, Y. Yao y Y. Kang, "Room Occupancy Prediction: Exploring the Power of Machine Learning and Temporal Insights," *American Journal of Applied Mathematics and Statistics*, vol. 12, no. 1, pp. 1–9, 2024. [En línea]. Disponible en: https://pubs.sciepub.com/ajams/12/1/1/index.html

[3] D. Fährmann, F. Boutros, P. Kubon, F. Kirchbuchner, A. Kuijper y N. Damer, "Ubiquitous Multi-Occupant Detection in Smart Environments," *Neural Computing and Applications*, vol. 36, no. 6, pp. 2941–2960, 2024. [En línea]. Disponible en: https://link.springer.com/article/10.1007/s00521-023-09162-z

[4] Ö. Çataltaş, "Room Occupancy Prediction from Temperature Data with Deep Convolutional Neural Networks," Journal of Applied Methods in Electronics and Computers, vol. 13, no. 2, pp. 37–43, 2025. [En línea]. Disponible en: https://ijamec.org/index.php/ijamec/article/view/446

[5] Kumari, P., Reddy, S. R. N., & Yadav, R. (2023). Indoor occupancy detection and counting system based on boosting algorithm using different sensor data. Building Research & Information, 52(1–2), 87–106. https://doi.org/10.1080/09613218.2023.2206090