# Seoul Bike Sharing Demand: Predecir Número De Bicicletas

María Fernanda Moreno Gómez

A01708653@tec.com

**ABSTRACT** Este documento presenta la implementación y evaluación de un modelo de Inteligencia Artificial para predecir la demanda de bicicletas compartidas en Seúl, utilizando el dataset "Seoul Bike Sharing Demand". El objetivo del modelo es ayudar a optimizar la distribución y disponibilidad de bicicletas en diferentes zonas de la ciudad, mejorando así la eficiencia del sistema de transporte. Se utilizó un modelo de regresión lineal multivariable, que fue entrenado con datos históricos sobre el uso de bicicletas, condiciones meteorológicas y factores estacionales. Los resultados indican que el modelo logra un coeficiente de determinación ($R^2$) de aproximadamente 0.51 en el conjunto de prueba, lo que sugiere que el modelo es capaz de explicar alrededor del 51% de la variabilidad en la demanda de bicicletas. Sin embargo, el error cuadrático medio al final del entrenamiento fue relativamente bajo, pero la precisión del modelo podría mejorarse mediante la implementación de técnicas más avanzadas o la incorporación de características adicionales.

*Keywords: Bike Sharing, Seoul, Demand Prediction, Linear Regression, Data Preprocessing, Gradient Descent, Transportation Systems.*

## 1. INTRODUCCIÓN

En los últimos años, los sistemas de bicicletas compartidas han evolucionado considerablemente, convirtiéndose en una parte fundamental de las redes de transporte urbano. Los sistemas de bicicletas compartidas comenzaron en Ámsterdam en 1965, con un enfoque sencillo y práctico: las personas pintaban sus bicicletas de color blanco, dejándolas libres para ser compartidas y posteriormente devueltas. Con el tiempo, este concepto evolucionó, introduciendo bicicletas diseñadas específicamente para ser utilizadas en áreas determinadas, las cuales podían desbloquearse en un sitio de préstamo de bicicletas y devolverse en otro sitio similar, haciendo que la disponibilidad de las bicicletas dependiera de la cantidad de bicicletas disponibles en cada estación.

A medida que estos sistemas se han expandido globalmente, se han convertido en una solución de movilidad clave en muchas ciudades, abordando problemas como la congestión del tráfico y la contaminación ambiental. De hecho, estudios recientes destacan cómo estos sistemas han sido adoptados en ciudades de todo el mundo, ofreciendo una alternativa ecológica al uso de automóviles privados y mejorando el acceso al transporte público [1]. Este crecimiento ha sido impulsado no solo por su conveniencia, sino también por sus beneficios ambientales, contribuyendo a la reducción de emisiones de carbono y promoviendo un estilo de vida más saludable entre los usuarios [1].

Además, la electrificación de los sistemas de bicicletas compartidas ha demostrado tener un impacto aún mayor en la

sostenibilidad urbana. Según PBSC Urban Solutions, la implementación de bicicletas eléctricas en estos sistemas no solo mejora la accesibilidad y comodidad para los usuarios, sino que también reduce significativamente la huella de carbono de las ciudades al disminuir las emisiones de CO2 [2]. Estos avances subrayan la importancia de seguir innovando en el diseño y la operación de sistemas de bicicletas compartidas para maximizar sus beneficios ambientales y sociales.

Sin embargo, a pesar de sus beneficios, la efectividad de los sistemas de bicicletas compartidas depende en gran medida de la capacidad para predecir la demanda en diferentes áreas y momentos del día. Esto es crucial para garantizar que las bicicletas estén disponibles donde y cuando se necesiten, optimizando así la eficiencia del sistema. Es aquí donde los modelos de inteligencia artificial, como los de regresión lineal multivariable, juegan un papel fundamental. Al analizar datos históricos sobre el uso de bicicletas, condiciones meteorológicas y factores estacionales, estos modelos pueden predecir la demanda con precisión, permitiendo una mejor planificación y gestión de los recursos del sistema.

El objetivo de este trabajo es implementar y evaluar un modelo de inteligencia artificial para predecir la demanda de bicicletas compartidas en Seúl, utilizando el dataset "Seoul Bike Sharing Demand". Este modelo busca mejorar la distribución y disponibilidad de bicicletas, contribuyendo así a un sistema de transporte urbano más eficiente y sostenible.

El dataset "Seoul Bike Sharing Demand" contiene datos registrados por hora sobre la demanda de bicicletas compartidas en Seúl, Corea, desde el 1 de diciembre de 2017 hasta el 30 de noviembre de 2018. Estos datos incluyen variables como las condiciones meteorológicas, la hora del día, y factores estacionales, todos los cuales son cruciales para predecir con precisión la demanda de bicicletas [3]. El propósito de este estudio es utilizar las instancias más relevantes de este dataset para entrenar un modelo de predicción que aproveche estas características, permitiendo una estimación más precisa del número de bicicletas necesarias en diferentes momentos y lugares de la ciudad. Este enfoque busca optimizar la operación del sistema de bicicletas compartidas en Seúl, asegurando que las bicicletas estén disponibles donde y cuando se necesiten, y contribuyendo a un sistema de transporte urbano más eficiente y sostenible.

## 2. DESCRIPCIÓN DEL DATASET

El dataset *"Seoul Bike Sharing Demand"* proporciona un registro detallado de la demanda de bicicletas compartidas en la ciudad de Seúl, Corea del Sur. Los datos abarcan un periodo de un año completo, desde el 1 de diciembre de 2017 hasta el 30 de noviembre de 2018, y están organizados en registros horarios, sumando un total de 8760 instancias.

### *2a.- Features*

El dataset incluye 14 columnas, cada una representando una característica relevante para la predicción de la demanda de bicicletas. A continuación, se describen las características disponibles:

- **Date:** La fecha en formato DD/MM/YYYY, representando el día específico del registro.
- **Rented Bike Count**: El número de bicicletas rentadas durante la hora específica (variable objetivo).
- **Hour**: La hora del día (0-23), representando la hora específica del registro.
- **Temperature(C)**: La temperatura en grados Celsius.
- **Humidity(%)**: El porcentaje de humedad relativa.
- **Wind speed (m/s)**: La velocidad del viento en metros por segundo.
- **Visibility (10m)**: La visibilidad en decenas de metros (por ejemplo, 2000 = 20 km).
- **Dew point temperature(C)**: La temperatura del punto de rocío en grados Celsius.
- **Solar Radiation (MJ/m2)**: La radiación solar medida en megajulios por metro cuadrado.
- **Rainfall(mm)**: La cantidad de precipitación en milímetros.
- **Snowfall (cm)**: La cantidad de nieve en centímetros.
- **Seasons**: La estación del año en la que se registró la instancia, categorizada como: Winter, Spring, Summer, Autumn.
- **Holiday**: Indica si el día del registro fue un día festivo (‘Holiday’) o no (‘No Holiday’).
- **Functioning Day**: Indica si el sistema de bicicletas estaba en funcionamiento (‘Yes’) o no (‘No’) durante esa hora específica.

[3]

## 3. EXTRACCIÓN Y LIMPIEZA DE DATOS

La extracción de datos implica recuperar datos de fuentes con el fin de analizarlos [4]. En este caso, se aplicaron técnicas de procesamiento de datos al dataset *"Seoul Bike Sharing Demand"* con el objetivo de optimizar la precisión del modelo de predicción.

### *3a.- Carga y filtrado de datos*

El primer paso en el preprocesamiento fue la carga del dataset y la revisión de datos faltantes. Se verificó que no había valores nulos en ninguna de las columnas, lo cual simplificó el proceso de limpieza de datos. Además, se verificó de igual manera que no existieran instancias repetidas o instancias con valores anormales (esto se obtuvo obteniendo las estadísticas de los features numéricos), de manera que no sucedió y se tuvo certeza que el dataset estaba correcto a nivel datos. A continuación, se realizó una exploración inicial de las características presentes en el dataset.

Durante esta exploración, se observó que la columna ‘Date’ contenía valores que se repetían cada hora para cada día. Esto implicaba que la fecha no aportaba información adicional para el análisis predictivo, ya que la demanda de bicicletas estaba registrada a nivel de hora y las diferencias entre fechas ya estaban capturadas por la variable ‘Hour’.

Posteriormente, las columnas ‘Functioning Day’ y ‘Holiday’ también fueron eliminadas. La columna ‘Functioning Day’ fue eliminada porque se observó que su valor estaba directamente relacionado con el conteo de bicicletas rentadas (‘Rented Bike Count’). Específicamente, cuando el conteo de bicicletas rentadas era 0, la columna ‘Functioning Day’ siempre

indicaba "No", lo que implicaba una redundancia en los datos. Esta relación directa no aportaba información adicional al modelo, por lo que se decidió prescindir de esta columna para evitar la redundancia y simplificar el modelo.

En cuanto a la columna 'Holiday', se decidió eliminarla porque su impacto podría estar indirectamente representado a través de otras variables, como las estaciones del año ('Seasons'). Además, el número de registros correspondientes a días festivos era significativamente menor en comparación con los días no festivos, lo que podría haber introducido un sesgo en el modelo. Dado que los datos sobre las estaciones ya proporcionan información relevante sobre patrones estacionales, se consideró que la columna 'Holiday' no era esencial para el análisis.

Adicionalmente, se observó que el conteo de bicicletas rentadas ('Rented Bike Count') presentaba una distribución altamente sesgada hacia valores bajos, como se muestra en la Figura 1 (el histograma de la distribución de 'Rented Bike Count'). Para evitar que los pocos valores extremadamente altos sesgaran el modelo, se decidió filtrar el dataset para excluir las instancias donde el conteo de bicicletas rentadas superaba los 2500. Esta decisión se basó en la observación de que las instancias con valores superiores a 2500 eran mínimas y podrían influir desproporcionadamente en el entrenamiento del modelo.

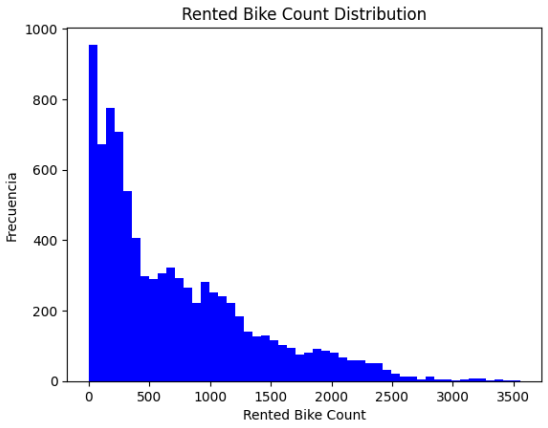



**Figura 1:** Histograma de la distribución de Rented Bike Count

El filtrar estos outliers es una práctica que ayuda a eliminar valores atípicos que influyan de manera negativa al modelo predictivo, pues incrementan la variabilidad de nuestros datos, reduciendo el poder estadístico, es por eso que, filtrar los datos atípicos puede hacer que los resultados se vuelvan estadísticamente significativos. [5].

*3b.- Análisis de Correlación y Selección de Características*

Para identificar las variables que tienen una relación lineal más fuerte con la variable objetivo 'Rented Bike Count', se calculó una matriz de correlación. Esta matriz muestra los coeficientes de correlación de Pearson entre cada par de variables, lo que indica el grado de asociación lineal entre ellas.

El coeficiente de correlación de Pearson varía entre -1 y 1. Un valor de 1 indica una correlación positiva perfecta, es decir, cuando una variable aumenta, la otra también lo hace de manera proporcional. Un valor de -1 indica una correlación negativa perfecta, donde un aumento en una variable corresponde a una disminución en la otra. Un valor cercano a

0 sugiere que no existe una relación lineal significativa entre las variables [6].

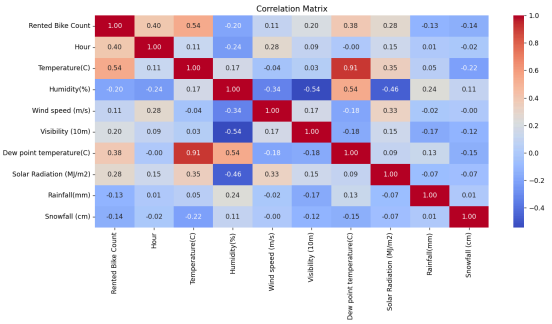



**Figura 2:** Matriz de correlación entre los features del dataset

Al observar la matriz de correlación generada (ver Figura 2), se identificaron las siguientes relaciones clave:

- 'Humidity(%)': Muestra una correlación negativa débil con Rented Bike Count (-0.20), lo que sugiere que la humedad no tiene un impacto significativo en la demanda de bicicletas. Además, esta variable está altamente correlacionada con otras como 'Dew point temperature(C)' (0.91), lo que podría introducir redundancia en el modelo.
- 'Wind speed (m/s)': Tiene una correlación muy baja con Rented Bike Count (0.11), lo que indica que no es un factor relevante en la predicción de la demanda de bicicletas.
- 'Rainfall(mm)' y 'Snowfall(cm)': Ambas variables muestran correlaciones negativas débiles con Rented Bike Count (-0.13 y -0.14, respectivamente), lo que indica que, aunque podrían tener algún impacto en el uso de bicicletas, no son factores determinantes en el contexto de este análisis.
- 'Visibility (10m)': Aunque tiene una correlación ligeramente positiva (0.20), su efecto es menor en comparación con otras variables seleccionadas, lo que llevó a su exclusión para simplificar el modelo.

## 4. TRANSFORMACIÓN DE DATOS

### *4a.- Normalización de Datos*

Normalización en el contexto de Machine Learning es una manera de escalar el rango de ciertos features a una escala estándar. Sirve cuando se tienen rangos de datos variables de los features, dado que, se asegura que todas las variables estén en una escala comparable y así evitar que aquellas que sean más grandes dominen el proceso de entrenamiento [7].

En este sentido, la fórmula de la normalización es la siguiente:

$$X_{\text{norm}} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

**Figura 3:** Fórmula para normalizar rangos de datos

Las columnas que se normalizaron fueron 'Rented Bike Count', 'Hour', 'Temperature(C)', 'Visibility (10m)', 'Dew point temperature(C)', 'Solar Radiation (MJ/m2)', y 'Rainfall(mm)'. Estas columnas fueron seleccionadas para normalización debido a que, cuando se trabajan algoritmos de regresión lineal en Machine Learning, al normalizar las variables se mejora la estabilidad numérica del modelo al contar con una escala estándar, más cuando se trabaja con diferentes unidades de medidas, como es el caso del dataset.

*4b.- Codificación de Variables Categóricas*

Se utilizó la técnica de “One-Hot Encoding”, que es cuando se presentan datos categóricos (como en el caso de ‘Season’) y mapear esos datos a enteros [8].

En este contexto, se transformó el feature ‘Seasons’ en variables dummy, con el fin de que se hicieran 4 columnas ‘Spring’, ‘Summer’, ‘Autumn’, ‘Winter’ con “False’s” si el valor en X instancia no es igual a dicha columna o con “True’s” cuando coincide el nombre de la columna con el contenido de X instancia en la columna de ‘Season’ y, posteriormente convirtiendo los “False” en 0 y los “True” en 1 para darles datos numéricos a estas columnas.

*4c.- División del Dataset en Conjuntos de Entrenamiento y Prueba*

Una vez limpiados y transformados los datos, se dividió el dataset en conjuntos de entrenamiento y prueba. La importancia de esta división es evaluar la capacidad del modelo para generalizar en datos no vistos. El dataset fue aleatorizado para evitar sesgos que pudieran existir por el orden original de las instancias.

Con eso en mente, el 80% de los datos se asignó a entrenamiento y el otro 20% se asignó a pruebas, siguiendo el principio de Pareto (regla 80-20) que dice que el 80% del efecto es impulsado por el 20% de las causas (y viceversa) [9].

## 5. CONSTRUCCIÓN Y EVALUACIÓN DEL MODELO

*5a.- Selección de Features y Variables Objetivo*

Los features utilizados para entrenar al modelo son: 'Hour', 'Temperature(C)', 'Dew point temperature(C)','Spring', 'Summer', 'Autumn', 'Winter','Solar Radiation (MJ/m2)', 'Rainfall(mm)', esto debido al análisis previo. Mientras que, el valor que queremos predecir (o nuestra variable objetivo) es ‘Rented Bike Count’, ya que deseamos saber la cantidad de bicicletas compartidas de acuerdo a los valores de los features.

*5b.- Sesgo en Features*

Un algoritmo linear suele tener un sesgo alto, lo que hace que aprenda rápido. En un análisis de regresión lineal, sesgo es el error introducido al acercarse a un problema de la vida real. Un modelo con un alto nivel de sesgo suele hacer más suposiciones sobre la relación entre las variables y el resultado final. Por otro lado, un modelo con un bajo nivel de sesgo hace menos suposiciones sobre cómo se comporta la función que relaciona esas variables con el resultado [10]. Este sesgo se implementó añadiendo una columna de unos a las matrices de características de entrenamiento y prueba.

*5c.- Inicialización de Parámetros*

Los parámetros del modelo se inicializan de manera aleatoria, con el fin de que el modelo no caiga en patrones repetitivos o en simetrías que puedan causar un mal ajuste [11].

*5d.- Cálculo de la Hipótesis*

En el contexto de la regresión lineal, la hipótesis se refiere a la función que utiliza

el modelo para predecir el valor esperado de la variable dependiente (en este caso, el conteo de bicicletas rentadas) en función de las características de entrada. La fórmula básica de la hipótesis en una regresión lineal simple es la siguiente [12]:

$$E(Y|x) = \beta_0 + \beta_1 x$$

**Figura 4:** Función de hipótesis para regresión lineal

En mi implementación, esta idea se extiende a múltiples variables explicativas (regresión lineal múltiple), y la función de hipótesis se calcula como una suma ponderada de todas las características de entrada multiplicadas por sus respectivos parámetros (o coeficientes). En el código, esto se realiza en la función h(params, sample), donde el producto punto entre los parámetros del modelo (params) y las características de entrada de una muestra específica (sample) da como resultado una predicción de la cantidad de bicicletas rentadas para ese conjunto de características.

*5e.- Cálculo del error*

El error de modelo (o también conocido como función de pérdida) mide la discrepancia entre las predicciones del modelo y los valores reales, es decir, mide la precisión de nuestro modelo [13].

En mi caso, utilicé el Mean Square Error (MSE), que es la diferencia cuadrática entre el valor predicho y el real [14]. Este error se calcula tomando la diferencia entre las predicciones del modelo y los valores reales (y), elevando al cuadrado estas diferencias, sumándolas y, finalmente, dividiéndolas por el doble del número de muestras (m) para obtener un promedio. En el código, esta operación se implementa en la función mse(params, samples, y), donde "samples" son las características de entrada, "params" son los parámetros actuales del modelo y "y" son los valores reales de la variable objetivo (Rented Bike Count). El objetivo del entrenamiento es minimizar este error, lo que se logra ajustando los parámetros del modelo a lo largo de las iteraciones.

$$mse = \frac{1}{n}\sum(y_n - \bar{y})^2$$

**Figura 5:** Fórmula del MSE

*5f.- Descenso de Gradiente*

El descenso del gradiente es un algoritmo de optimización usada para entrenar modelos de aprendizaje automático y redes neuronales [15]. Se usa para minimizar la función de pérdida (que, en el caso de mi implementación, es el MSE). Es una función iterativa de ajustar los parámetros del modelo para minimizar la diferencia entre la predicción y los valores reales.

En el contexto del dataset, En cada iteración, calcula la pendiente (o gradiente) de la función de pérdida con respecto a los parámetros y ajusta los parámetros en la dirección opuesta a esta pendiente. El tamaño de este ajuste está controlado por la tasa de aprendizaje "alfa", que determina qué tan grandes son los pasos que el algoritmo da hacia el mínimo de la función de pérdida.

$$\Theta_j := \Theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}[(h_\Theta(x_i) - y)x_i]$$

**Figura 6:** Actualización de parámetros para el descenso del gradiente para la regresión lineal.

Primero, se calcula la predicción del modelo (h(Θ(x))) multiplicando las características de las muestras (samples) por los parámetros actuales del modelo (params). Luego, se obtiene el error restando las predicciones obtenidas del valor real (y). Este error representa la diferencia entre lo que el modelo predice y los datos reales.

A continuación, se calcula el gradiente, que mide la dirección y magnitud en la que deben ajustarse los parámetros para reducir el error. En mi código, esto se logra multiplicando la transpuesta de la matriz de características por el vector de errores y dividiendo por el número de muestras. Finalmente, los parámetros se actualizan restando este gradiente multiplicado por la tasa de aprendizaje (alfa). Este proceso se repite en cada iteración (o época), lo que permite que el modelo se ajuste de manera progresiva y aprenda a hacer predicciones más precisas.

*5g.- Evaluación del Modelo*

Para evaluar el rendimiento del modelo, se utilizó la métrica de R² (coeficiente de determinación), que es una medida de correlación de los features con el objetivo. Va desde -∞ a ∞, Cuanto más se acerque a 1, más varianza del valor objetivo pueden explicar las variables de las características. En otras palabras, cuanto más probable sea que haya variables importantes que conduzcan a predicciones precisas [14].

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

**Figura 7:** Maneras de calcular la R²

*5h.- Test, train y validation*

El dataset, con el propósito de evaluar (y mejorar en caso de ser necesario) la capacidad del modelo para predecir datos no vistos, se dividió en tres partes: train (entrenamiento) con 60% del dataset, test (prueba) con 20% del dataset y validation (validación) con 20% del dataset. En Machine Learning, se dice que no hay que usar el mismo conjunto de datos para el entrenamiento y la evaluación del modelo, pues al hacerlo, el modelo se puede sesgar al aprenderse de memoria los datos existentes con la incapacidad de poder predecir datos nuevos [16].

El conjunto de train se usa para entrenar y hacer que el modelo se aprenda los patrones que se encuentran en los datos. Durante el entrenamiento, el modelo ajusta sus parámetros para minimizar el error entre las predicciones y los valores reales.

El conjunto de validation se usa para validar el rendimiento del modelo durante el entrenamiento. Este modelo evalúa la capacidad del modelo de generalizar a datos no vistos, pues realiza la evaluación del modelo en el conjunto de validación después de cada época.

El conjunto de test se usa para probar el modelo después de su entrenamiento, pues da una métrica imparcial del rendimiento del modelo en términos de exactitud y precisión.

Durante el proceso de entrenamiento, se usan las métricas de MSE (error cuadrático medio) y el R² (coeficiente de determinación) para los tres conjuntos, con

el fin de evaluar la precisión del modelo en cada etapa.

Es importante mantener un equilibrio entre los conjuntos de entrenamiento y validación para evitar el sobreajuste (overfitting). Si el modelo se ajusta demasiado bien a los datos de entrenamiento, podría perder la capacidad de generalizar a nuevos datos, mostrando un alto rendimiento en el entrenamiento, pero un bajo rendimiento en los conjuntos de validación y prueba. Por ello, el uso adecuado de los tres conjuntos garantiza que el modelo no solo aprenda los patrones de los datos, sino que también sea capaz de aplicarlos a situaciones reales.

*5i.- Underfitting, Fitting y Overfitting*

En el entrenamiento de modelos en Machine Learning, hay tres escenarios que afectan al rendimiento de un modelo: underfitting (subajuste), fitting (ajuste adecuado) y overfitting (sobreajuste).

Underfitting ocurre cuando el modelo es demasiado simple para capturar la complejidad de los datos. El modelo no logra aprender bien los patrones de los datos de entrenamiento, lo que resulta en un mal desempeño tanto en los datos de entrenamiento como en los de prueba. Un indicativo de underfitting es un $R^2$ bajo y un error alto en ambos conjuntos de datos, ya que se caracteriza por tener un alto bias (sesgo) y baja varianza [17].

Sucede cuando:

- El modelo es demasiado simple para representar datos con relaciones complejas.
- Los features usados en el modelo no son los adecuados para predecir la variable objetivo.
- El tamaño del conjunto de train no es lo suficientemente grande.
- Los datos no están correctamente limpiados.

Se puede evitar con:

- Aumentar complejidad del modelo.
- Cambiar o aumentar los features.
- Limpiar los datos.
- Incrementar el número de epochs o el tamaño del conjunto de train.

Overfitting ocurre cuando el modelo no hace predicciones precisas sobre los datos de prueba, ya que captura el ruido y patrones específicos que no son aplicables a otros datos. Podemos identificar el overfitting cuando hay un rendimiento alto en el conjunto de train, pero es deficiente en test o validation. Esto se debe a que el modelo "memoriza" los datos en lugar de generalizar patrones, lo que resulta en un $R^2$ alto en train y bajo en test [17].

Sucede cuando:

- Hay una alta varianza y bajo sesgo.
- El modelo es muy complejo.

Se puede evitar con:

- Mejorar la calidad del conjunto de train.
- Aumentar datos en train (para que no generalice).
- Reducir complejidad del modelo.
- Regulaciones o modificar los hiperparámetros del modelo.

Fitting adecuado es cuando un modelo puede capturar los patrones de los datos sin ajustarse demasiado al ruido o a

patrones específicos. En este caso, se muestra un balance en el desempeño entre train, test y validation, reflejándose en un $R^2$ similar en los tres conjuntos de datos [17].

*5j.- Bias y Varianza*

En Machine Learning, bias (sesgo) y varianza nos dicen qué tan preciso es el modelo y qué tan buna es su capacidad de predecir datos nuevos con base a un conjunto de datos. Tanto el bias y la varianza describen el error del modelo y su relación con los datos.

Bias es la incapacidad del modelo para hacer que se produzca alguna diferencia o error entre el valor predicho del modelo y el valor real. Un bias alto indica que el modelo hace muchas suposiciones, por lo que lleva que el modelo sea muy simple y no se puedan capturar relaciones complejas entre las variables [18].

- Un bias bajo indica que se toman menos suposiciones para construir la predicción, por lo que el modelo va a coincidir bastante con el conjunto de datos de train.
- Un bias alto indica que se toman más suposiciones para contruir la suposición. Por ello, el modelo no va a coincidir tanto con el conjunto de datos de train.

Varianza es la cantidad de cambio que se da en el rendimiento de un modelo cuando se entrena con distintos subconjuntos de entrenamiento, es decir, cuánto puede ajustarse al nuevo subconjunto de datos [18].

- Una baja varianza indica que el modelo es menos sensible a los datos en el conjunto de train y por ende, dar predicciones más consistentes en los distintos conjuntos de datos.
- Un bias alto indica que el modelo es muy sensible a los cambios en los datos de train y por ende, dar predicciones menos consistentes en los distintos conjuntos de datos.

Un modelo ideal tiene un bias y una varianza moderados, lo que le permite aprender lo suficiente de los datos de entrenamiento sin sobreajustarse a ellos.

## 6. VISUALIZACIÓN DE RESULTADOS DEL CASO BASE

Después de entrenar el modelo y realizar las pruebas pertinentes, los resultados muestran que el modelo alcanzó una exactitud en términos del coeficiente de determinación ($R^2$) con los siguientes valores:

- Train $R^2$: 0.4705
- Validation $R^2$: 0.4657
- Test $R^2$: 0.4382


**Figura 8:** Datos finales del modelo para train, test y validation.

Estos resultados sugieren que el modelo fue capaz de capturar aproximadamente el 47.05% de la variabilidad en los datos de train, el 46.57% en los datos de validation, y el 43.82% en los datos de test.

Los valores de $R^2$ en los conjuntos de entrenamiento (0.4705), validación (0.4657) y prueba (0.4382) son relativamente similares, lo que indica que el modelo no está experimentando overfitting. En un escenario de overfitting,

se hubiera observado un $R^2$ significativamente mayor en el conjunto de entrenamiento y un valor considerablemente más bajo en los conjuntos de validación y prueba, lo que no es el caso aquí. Esta consistencia en los valores entre los tres conjuntos de datos sugiere que el modelo es estable y no está reaccionando de manera excesiva a las particularidades del conjunto de entrenamiento, lo que indica una varianza baja.

Sin embargo, a pesar de esta estabilidad, los valores moderadamente bajos de $R^2$ en los tres conjuntos sugieren que el modelo está enfrentando un problema de underfitting. La incapacidad del modelo para lograr un $R^2$ más alto, incluso en el conjunto de entrenamiento, implica que el modelo es demasiado simple para capturar las relaciones complejas en los datos. Un $R^2$ del 43.82% en el conjunto de prueba indica que el modelo solo puede explicar una porción moderada de la variabilidad en los datos, lo que puede señalar que no se han utilizado suficientes características relevantes o que las relaciones entre las variables son más complejas de lo que el modelo actual puede manejar.

Este underfitting es un signo de un bias alto. El modelo está haciendo suposiciones simplificadas sobre las relaciones entre los datos, lo que lleva a un ajuste ineficiente. La solución para reducir este bias sería aumentar la complejidad del modelo, ya sea utilizando más características o probando con modelos más avanzados que puedan capturar mejor las interacciones entre las variables.

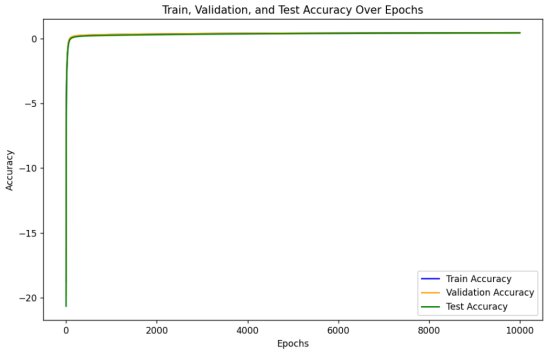



**Figura 9:** Precisión de entrenamiento, validación y prueba a lo largo de las épocas.

La gráfica muestra cómo la precisión se estabiliza después de un número determinado de iteraciones, con valores similares para los tres conjuntos de datos, lo que sugiere que el modelo ha encontrado un buen balance sin overfitting a los datos de entrenamiento.

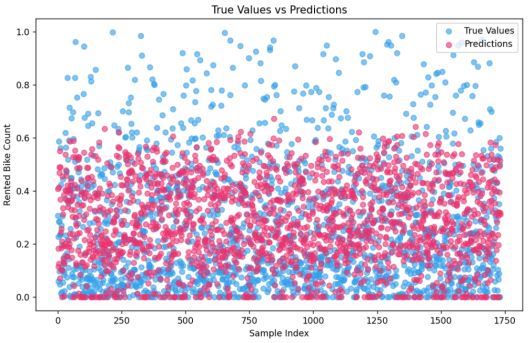



**Figura 10:** Comparación entre los valores reales y predichos para el conteo de bicicletas rentadas.

La dispersión de los puntos muestra que, si bien el modelo es capaz de predecir ciertos patrones generales, existe un margen considerable de error en las predicciones individuales.

## 7. IMPLEMENTACIÓN DE FRAMEWORKS

En esta sección se presenta la implementación del modelo de Random Forest Regressor utilizando Python y las librerías pandas, numpy, scikit-learn y

matplotlib. El objetivo es predecir la demanda de bicicletas en Seúl, aplicando técnicas de preprocesamiento y entrenamiento de modelos, además de analizar los resultados obtenidos.

*7a.- Carga y preprocesamiento de datos*

El proceso comenzó cargando el dataset SeoulBikeData.csv con la librería pandas, eliminando las instancias en las que el conteo de bicicletas superaba las 2500 para evitar sesgos derivados de outliers. Posteriormente, se normalizaron varias columnas relevantes, incluidas Rented Bike Count, Temperature(C) y Solar Radiation (MJ/m2), utilizando la normalización mínima-máxima. Este paso es fundamental para que todas las características estén en la misma escala y evitar que los rangos de valores más grandes dominen el proceso de entrenamiento del modelo.

Además, se realizó la codificación one-hot encoding para la columna categórica Seasons, creando columnas individuales para Spring, Summer, Autumn y Winter. Este enfoque permite que el modelo pueda interpretar adecuadamente las variables categóricas durante el entrenamiento.

*7b.- División del conjunto de datos*

Como se mostró en el caso base, el dataframe se dividió en tres conjuntos: entrenamiento (60%), validación (20%) y prueba (20%). Esta división es importante para garantizar que el modelo pueda generalizar bien y no sufra de overfitting. El conjunto de validación se utiliza para ajustar el modelo durante el entrenamiento, mientras que el conjunto de prueba sirve para evaluar su rendimiento final en datos no vistos.

*7c.- Selección de features para el modelo*

Los features seleccionados para entrenar el modelo son: ‘Hour’, ‘Temperature(C)’, ‘Dew point temperature(C)’, y las columnas resultantes de la codificación de ‘Seasons’. Estas variables fueron seleccionadas con base en el análisis de correlación del caso base y relevancia para la predicción de la demanda de bicicletas.

*7d.- Creación del modelo*

Se utilizó el algoritmo Random Forest, con 100 árboles de decisión y una semilla aleatoria, para garantizar la reproducibilidad de los resultados. El Random Forest es adecuado para este tipo de problemas debido a su capacidad para manejar tanto características continuas como categóricas, y su robustez frente a outliers, pues la lógica dice que las observaciones que se ajusten a los criterios seguirán la rama "Sí" y las que no lo hagan seguirán la ruta alternativa, por lo que buscarán la mejor división para crear subconjuntos de datos [19].

Su enfoque consiste en la creación de múltiples árboles de decisión, donde cada uno de estos árboles es entrenado con diferentes subconjuntos de los datos, utilizando una técnica llamada bagging (bootstrap aggregating). Esto permite al modelo reducir la varianza y aumentar su capacidad para generalizar mejor a datos no vistos.

Funciona de la siguiente manera:

- Se construyen distintos árboles de decisión con subconjuntos randoms de los datos de entrenamiento (para que el árbol se entrene con

posiciones distintas del dataset y pueda dar mejores predicciones).

- El Random Forest toma la media de las predicciones generadas por cada árbol del problema de regresión.
- Calcula la importancia de los features con los que fue entrenado, de manera que se pueda identificar cuantitativamente los features que aportan más al modelo.

Si bien son un método muy usado en Machine Learning, pueden tener problemas con bias y de overfitting, Sin embargo, cuando varios árboles de decisión forman un conjunto en el algoritmo de random forest, predicen resultados más precisos, especialmente cuando los árboles individuales no están correlacionados entre sí [19].

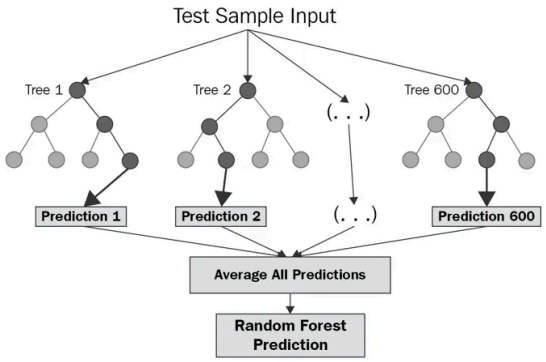



**Figura 11:** Diagrama del funcionamiento del Random Forest.

*7e.- Resultados para el modelo de Random Forest*

Después de haber dividido el dataset en los conjuntos de entrenamiento, validación y prueba, se entrenó el modelo Random Forest y se obtuvieron los siguientes resultados:

- Train:
  - MSE: 0.0022
  - $R^2$: 0.9616
- Validation:
  - MSE: 0.0138
  - $R^2$: 0.7704
- Test:
  - MSE: 0.0135
  - $R^2$: 0.7756

```
Train MSE: 0.0022, Train R²: 0.9616
Validation MSE: 0.0138, Validation R²: 0.7704
R² en el conjunto de prueba: 0.7756
Mean Squared Error en el conjunto de prueba: 0.0135
```

**Figura 12:** Resultados del modelo de regresión de Random Forest.

Estos resultados sugieren que el modelo Random Forest se ajustó muy bien a los datos de entrenamiento, explicando el 96.16% de la variabilidad en el conjunto de entrenamiento. Sin embargo, al evaluar el modelo con los datos de validación y prueba, la precisión disminuye, con un $R^2$ de 0.7704 y 0.7756 respectivamente. Esto indica que, aunque el modelo generaliza de manera adecuada, aún hay margen para mejorar la capacidad de predicción, especialmente en datos no vistos.

La diferencia observada entre el conjunto de entrenamiento y los de validación y prueba puede indicar una ligera tendencia al overfitting, donde el modelo se ajusta mejor a los datos de entrenamiento en comparación con los nuevos datos.

Un MSE bajo es indicativo de que el modelo está logrando minimizar el error promedio cuadrático entre los valores predichos y los reales, lo que sugiere que las predicciones del modelo son, en general, precisas. En términos de bias y varianza, esto significa que el modelo tiene un bias relativamente bajo, ya que no está cometiendo errores en sus predicciones, y

una varianza controlada, dado que los resultados de validación y prueba no difieren drásticamente de los del conjunto de entrenamiento. Esto sugiere que el modelo tiene una buena capacidad de generalización, aunque la ligera diferencia entre los conjuntos indica que aún podría beneficiarse de técnicas adicionales para reducir la varianza y evitar el overfitting.

El modelo también calculó la importancia de cada feature:

- 'Hour': 0.2890
- 'Temperature(C)': 0.3687
- 'Dew point temperature(C)': 0.1043
- 'Spring': 0.0067
- 'Summer': 0.0031
- 'Autumn': 0.0229
- 'Winter': 0.0164
- 'Solar Radiation (MJ/m2)': 0.1088
- 'Rainfall(mm)': 0.0801

Donde, la característica más importante para el modelo fue 'Temperature(C)', con una importancia del 36.87%, seguida de la Hour con un 28.90%. Esto es consistente con la expectativa de que la temperatura y la hora del día son factores clave para predecir la demanda de bicicletas. 'El Dew point temperature(C)' y la 'Solar Radiation' también mostraron importancia moderada, mientras que las estaciones del año tuvieron una contribución menor al modelo.

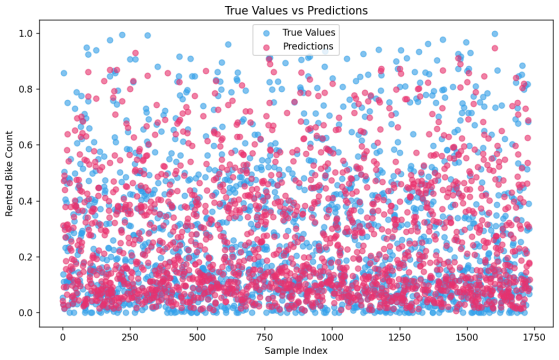



**Figura 13:** Comparación entre los valores reales y las predicciones.

La Figura 13 muestra que aunque el modelo es capaz de capturar tendencias generales, persisten errores en algunas predicciones, lo que indica que el modelo aún podría beneficiarse de mejoras adicionales, como la inclusión de características adicionales o el ajuste de hiperparámetros.

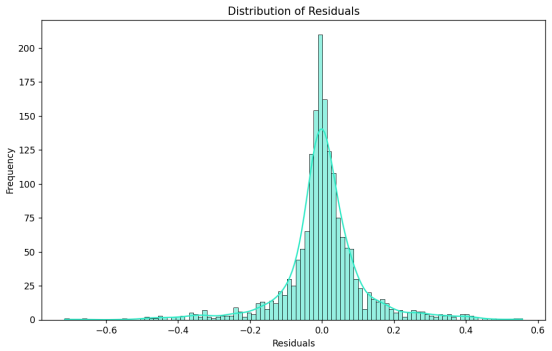



**Figura 14:** Análisis de los residuos: valores reales menos valores predichos.

La Figura 14 muestra que los errores entre los valores predichos y los valores reales están distribuidos de manera simétrica alrededor de 0, lo que es un buen indicativo de que el modelo no está sesgado hacia predicciones altas o bajas. Sin embargo, se observa una ligera concentración de residuos cercanos a 0, lo que sugiere que el modelo tiende a predecir más correctamente los valores

promedio, pero podría tener dificultades con valores extremos.

## 8. MEJORA DEL MODELO CON FRAMEWORK

Al implementar el modelo Random Forest, se observan mejoras notables (en relación con el modelo base) en términos de bias y varianza, que son esenciales para entender el comportamiento del modelo en diferentes escenarios de ajuste.

El bias mide el error debido a suposiciones excesivamente simples en el modelo. Un modelo con alto bias puede no capturar las relaciones complejas en los datos, resultando en underfitting. Al contrario, un modelo con alta varianza se ajusta demasiado bien a los datos de entrenamiento y puede fallar en generalizar adecuadamente a nuevos datos, llevando al overfitting.

El modelo de Random Forest es útil para controlar la varianza, ya que utiliza múltiples árboles de decisión entrenados en subconjuntos aleatorios de los datos. Esto permite que el modelo reduzca la sensibilidad a los cambios en los datos, disminuyendo el riesgo de overfitting, que se evidenció en el modelo base. En términos del MSE y $R^2$, el modelo con framework presentaba una mayor tendencia a overfitting debido a la disparidad entre los valores obtenidos en el conjunto de entrenamiento y los de prueba.

Para reducir esta tendencia a overfitting, se regularizó el modelo ajustando los hiperparámetros del Random Forest, donde la implementación de parámetros como *max_depth=7* y *min_samples_leaf=5* ayudó a controlar la profundidad de los árboles y evitar que el modelo se ajuste demasiado a los datos de entrenamiento.

Los resultados del modelo mejorado reflejan estas modificaciones, con valores de MSE y $R^2$ que muestran un mejor equilibrio entre los conjuntos de entrenamiento, validación y prueba:

- Train MSE: 0.0136
- Train $R^2$: 0.7626
- Validation MSE: 0.0155
- Validation $R^2$: 0.7380
- Test MSE: 0.0150
- Test $R^2$: 0.7609

```
Train MSE: 0.0136, Train R²: 0.7626
Validation MSE: 0.0155, Validation R²: 0.7380
R² en el conjunto de prueba: 0.7609
Mean Squared Error en el conjunto de prueba: 0.0150
```

**Figura 15:** Resultados del modelo de regresión de Random Forest mejorado.

Estos resultados reflejan que el modelo no está sufriendo de overfitting, ya que no hay una disparidad significativa entre los resultados del conjunto de entrenamiento y los conjuntos de validación y prueba. El $R^2$ relativamente cercano entre estos conjuntos sugiere que el modelo ha logrado un buen equilibrio entre bias y varianza, lo que permite generalizar mejor a nuevos datos. En otras palabras, el modelo no está "memorizando" los datos de entrenamiento, sino que ha aprendido patrones.

Además, el análisis de importancia de las características del modelo muestra cómo el modelo prioriza ciertas variables para hacer sus predicciones. Las características más influyentes en el modelo fueron:

- Temperature(C): 38.51%
- Hour: 33.11%
- Solar Radiation (MJ/m2): 8.50%

Esto es consistente con la lógica de que la temperatura y la hora del día son factores clave para predecir la demanda de bicicletas. Otras variables, como las estaciones del año, tienen una importancia menor, lo que refleja que su impacto en la predicción es más reducido.

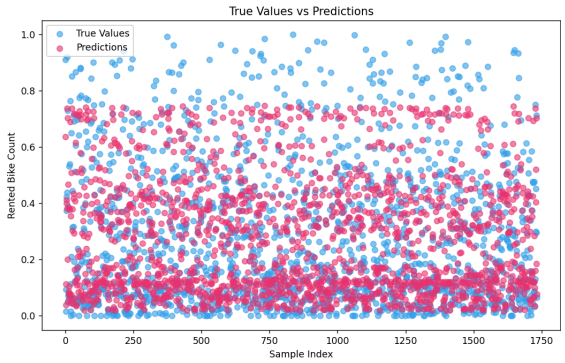



**Figura 16:** Comparación entre los valores reales y las predicciones del framework mejorado.

Esta gráfica muestra la relación entre los valores reales (en azul) y las predicciones (en rosa) obtenidas por el modelo Random Forest mejorado. La cercanía entre los puntos de ambos colores a lo largo del gráfico sugiere que el modelo está logrando capturar bien las tendencias generales de los datos. Sin embargo, se observan algunas diferencias en predicciones individuales, lo que indica que aún hay margen para mejorar en la precisión de ciertos valores extremos.

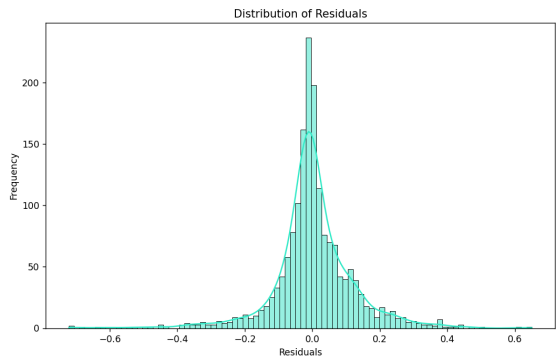



**Figura 17:** Análisis de los residuos: valores reales menos valores predichos del framework mejorado.

El gráfico de distribución de residuos ilustra cómo los errores de predicción están distribuidos. La mayoría de los residuos están centrados cerca de 0, lo que indica que el modelo no está presentando un sesgo significativo en las predicciones. La simetría de la gráfica refuerza que el modelo Random Forest está haciendo predicciones de manera equilibrada, sin sobreestimar ni subestimar sistemáticamente los valores de la demanda de bicicletas.

*8a.- Análisis comparativo entre el modelo base, el modelo con framework y el modelo mejorado del framework*

A continuación, se realiza una comparación entre los tres modelos utilizados: el modelo base de regresión lineal, el modelo de Random Forest con los parámetros por defecto, y el modelo de Random Forest mejorado con ajustes de hiperparámetros. Esta comparación se centra en dos métricas clave: el coeficiente de determinación $R^2$ y el Mean Squared Error (MSE) para los conjuntos de datos de entrenamiento, validación y prueba.

| Modelo | Train MSE | Validation MSE | Test MSE | Train $R^2$ | Validation $R^2$ | Test $R^2$ |
|---|---|---|---|---|---|---|
| Regresión lineal | 0.0245 | 0.0257 | 0.0261 | 0.4705 | 0.4657 | 0.4382 |
| Framework | 0.0022 | 0.0138 | 0.0135 | 0.9616 | 0.7704 | 0.7756 |
| Framework mejorado | 0.0136 | 0.0155 | 0.0150 | 0.7626 | 0.7380 | 0.7609 |

**Análisis del Modelo Base (Regresión Lineal)**

- Underfitting: El modelo de regresión lineal presenta un coeficiente $R^2$ de 0.4705 en el conjunto de entrenamiento, lo que indica que el modelo no está capturando suficientemente bien la relación entre las variables. El hecho de que el valor de $R^2$ en el conjunto de prueba sea

relativamente cercano al del entrenamiento (0.4382) indica que el modelo está generalizando bien, pero a costa de un ajuste insuficiente.

- Bias: El modelo muestra un alto bias, lo que sugiere que está haciendo suposiciones demasiado simples sobre la relación entre las características y la demanda de bicicletas. Este alto sesgo se manifiesta en los valores bajos de $R^2$ en todos los conjuntos, lo que implica que el modelo no está capturando adecuadamente las complejidades de los datos.
- Varianza: La varianza es baja, ya que el rendimiento entre los conjuntos de entrenamiento y prueba es bastante similar. Sin embargo, esto es resultado del underfitting, lo que impide que el modelo aprenda lo suficiente de los datos de entrenamiento.

**Análisis del Modelo con Framework (Random Forest)**

- Fitting: El modelo de Random Forest mejora considerablemente el ajuste en comparación con el modelo base. El $R^2$ en el conjunto de entrenamiento es de 0.9616, lo que indica un excelente ajuste a los datos de entrenamiento. Sin embargo, la caída en el $R^2$ en el conjunto de prueba a 0.7756 sugiere que el modelo está empezando a mostrar una ligera tendencia al overfitting.
- Bias: El modelo tiene un bias bajo en el conjunto de entrenamiento, lo que sugiere que está capturando correctamente las relaciones en los datos. Sin embargo, el bajo sesgo en el entrenamiento, combinado con la caída en el conjunto de prueba, sugiere que el modelo está memorizando los datos del entrenamiento.
- Varianza: La varianza es alta, lo que se refleja en la diferencia notable entre los valores de $R^2$ en el conjunto de entrenamiento y los de prueba. Este alto nivel de varianza es un signo de overfitting.

**Análisis del Modelo Mejorado (Random Forest con Hiperparámetros Ajustados)**

- Fitting: El modelo mejorado logra un buen fitting con un $R^2$ de 0.7626 en el conjunto de entrenamiento, que es más bajo que el de la versión por defecto del modelo de Random Forest, pero más equilibrado. El $R^2$ en los conjuntos de validación y prueba es cercano (0.7380 y 0.7609 respectivamente), lo que indica que el modelo generaliza mejor que en la versión por defecto.
- Bias y Varianza: El modelo mejorado muestra un bias más equilibrado y una varianza más controlada en comparación con el modelo por defecto. Los valores de MSE y $R^2$ en los tres conjuntos de datos son más cercanos, lo que sugiere que el modelo ha reducido la varianza sin incrementar significativamente el sesgo. Esto sugiere que el modelo no está sobreajustando, lo que lo convierte en una solución más robusta.

Por ello, el modelo base de regresión lineal muestra claramente un caso de

underfitting. Aunque tiene la capacidad de generalizar de manera consistente entre los conjuntos de entrenamiento y prueba, su ajuste es insuficiente para capturar las relaciones complejas entre las variables. Esto se debe a que el modelo es demasiado simple y no logra adaptarse adecuadamente a los patrones subyacentes en los datos.

Por otro lado, el modelo con Random Forest sin ajustes de hiperparámetros presenta una ligera tendencia al overfitting. Si bien el modelo tiene un bias bajo en el conjunto de entrenamiento, lo que significa que está capturando bien las relaciones en ese conjunto, la alta varianza entre los conjuntos de entrenamiento y prueba indica que el modelo está aprendiendo demasiado de los datos de entrenamiento, capturando incluso el ruido presente en ellos. Esto le impide generalizar correctamente a datos nuevos.

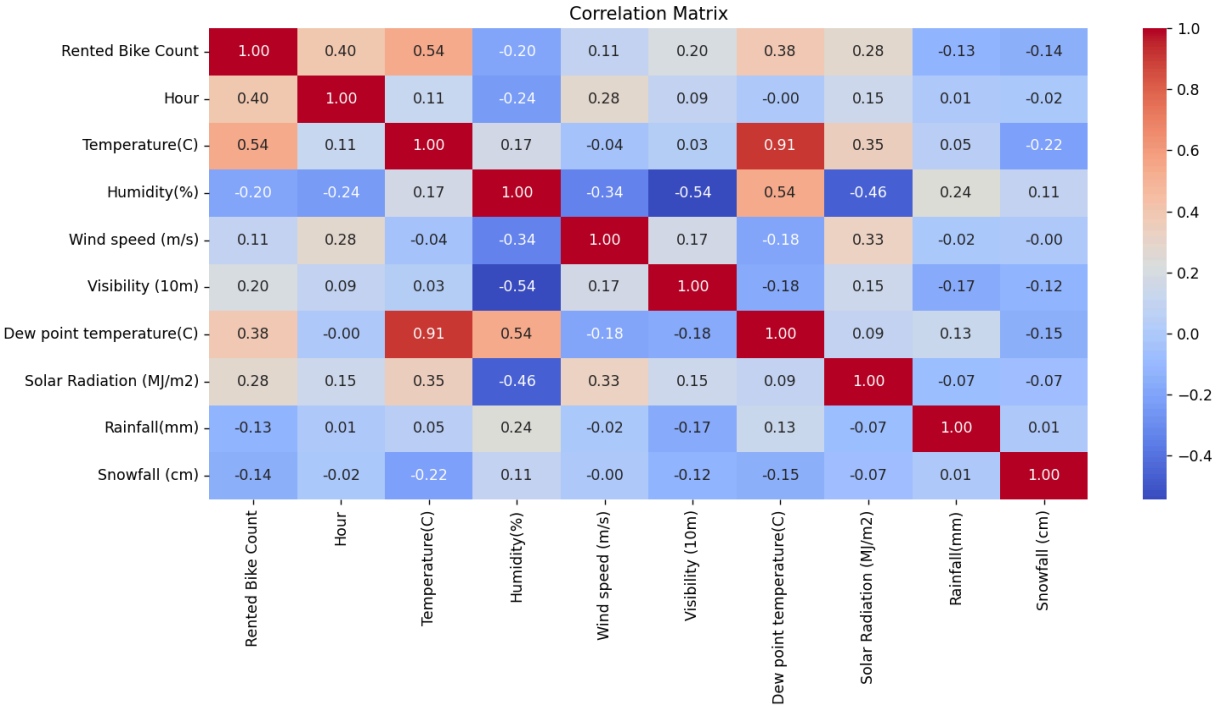
Finalmente, el modelo mejorado de Random Forest, con ajustes en los hiperparámetros, logra un buen equilibrio. Al reducir la varianza que se observaba en la versión sin ajustes, el modelo mejora su capacidad de generalización sin aumentar de forma significativa el bias. Esto se traduce en un mejor desempeño tanto en los conjuntos de validación como en los de prueba, lo que sugiere que el modelo ha encontrado un buen equilibrio entre bias y varianza y puede hacer predicciones más confiables en datos no vistos.

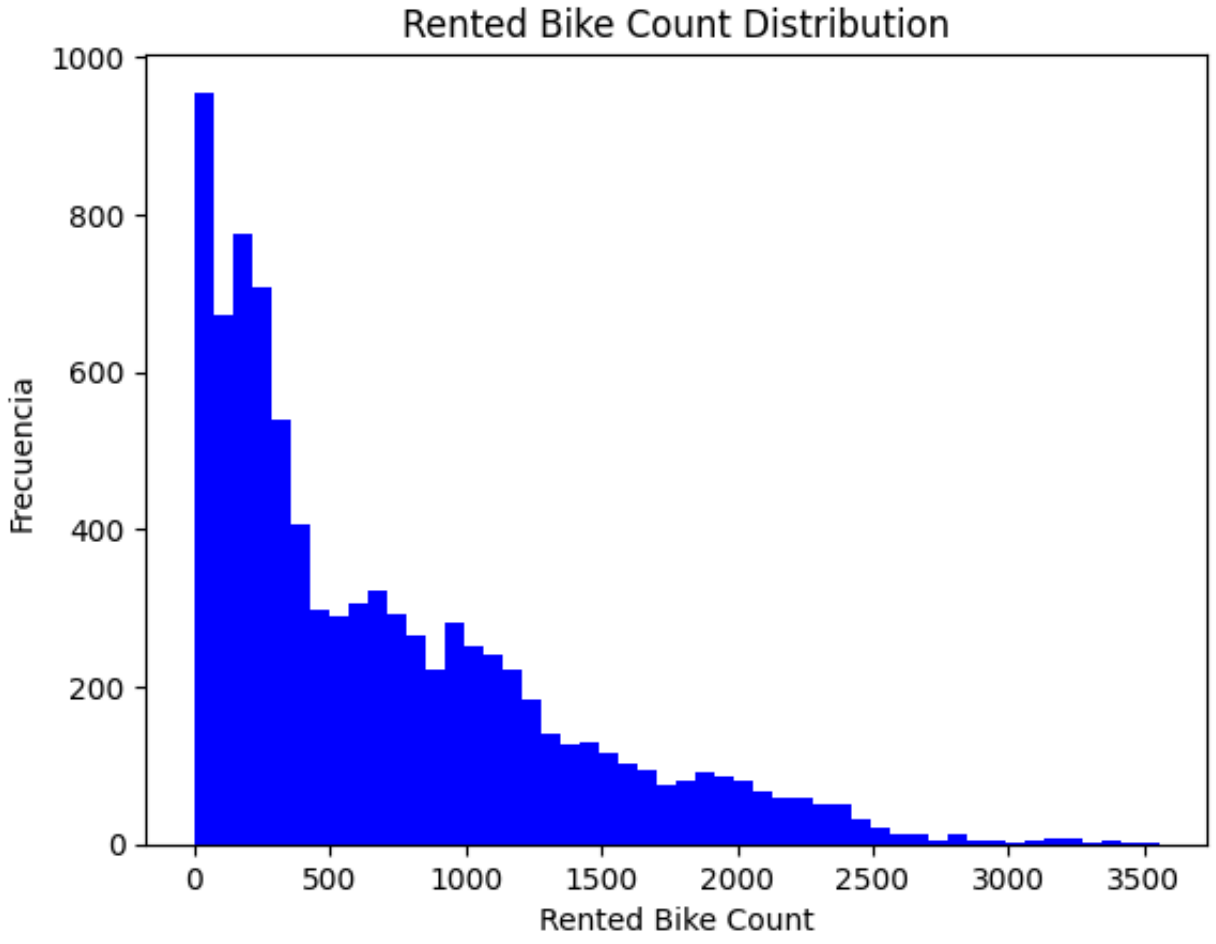
## 7. REFERENCIAS

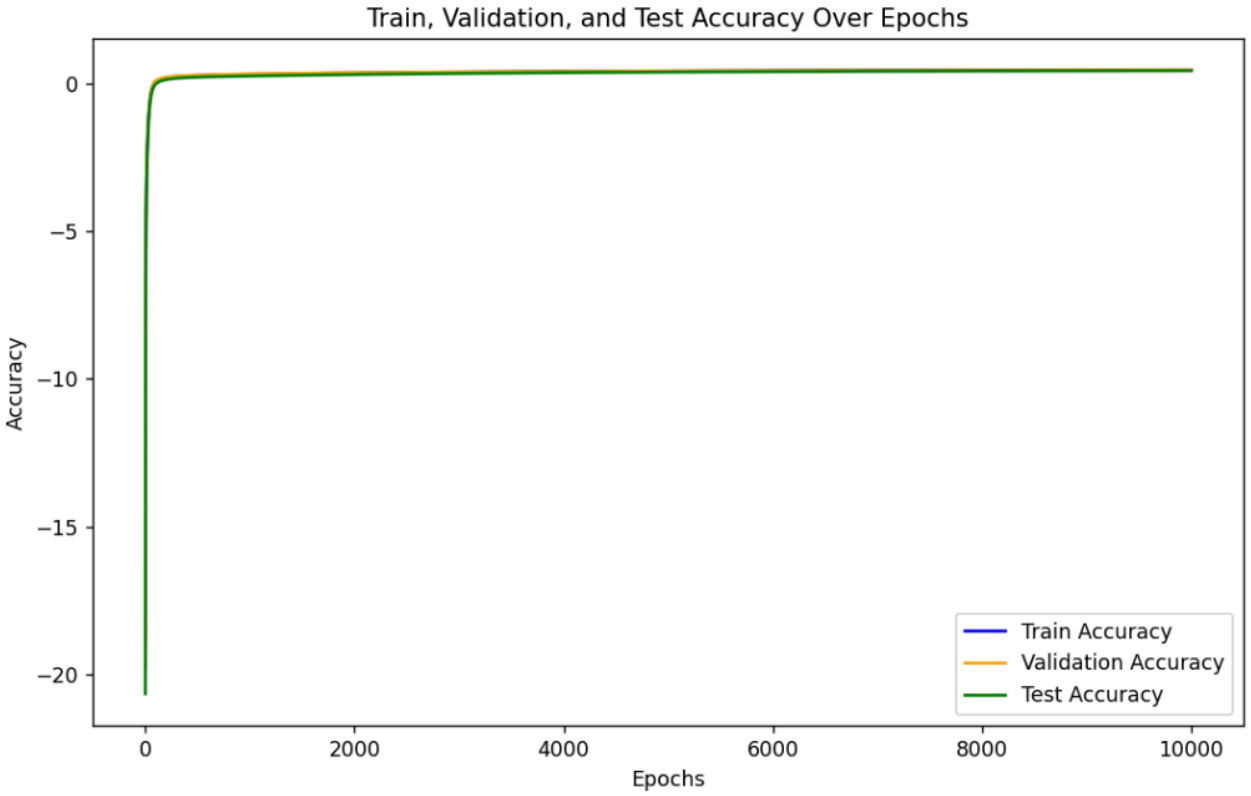

1. Shaheen, S., Guzman, S., & Zhang, H. (2020). The Development, Characteristics, and Impact of Bike Sharing Systems: A Literature Review. Retrieved from ResearchGate.
2. PBSC Urban Solutions. (2023). Electrifying Bike Sharing Systems: A Positive Impact on the Environment and Our Society. Retrieved from PBSC.
3. Seoul Bike Sharing Demand Dataset. (n.d.). Retrieved from https://archive.ics.uci.edu/dataset/560/seoul+bike+sharing+demand
4. Docsumo. (n.d.). Data Extraction with Machine Learning. Retrieved from https://www.docsumo.com/blogs/data-extraction/machine-learning
5. Statistics by Jim. (n.d.). How to Remove Outliers in Data. Retrieved from https://statisticsbyjim.com/basics/remove-outliers/
6. DATAtab. (n.d.). Pearson Correlation Tutorial. Retrieved from https://datatab.es/tutorial/pearson-correlation
7. DataCamp. (n.d.). Normalization in Machine Learning. Retrieved from https://www.datacamp.com/tutorial/normalization-in-machine-learning
8. Educative.io. (n.d.). One-Hot Encoding in Machine Learning. Retrieved from https://www.educative.io/blog/one-hot-encoding
9. Ahmed, N. (n.d.). The Motivation for Train-Test Split. Medium. Retrieved from

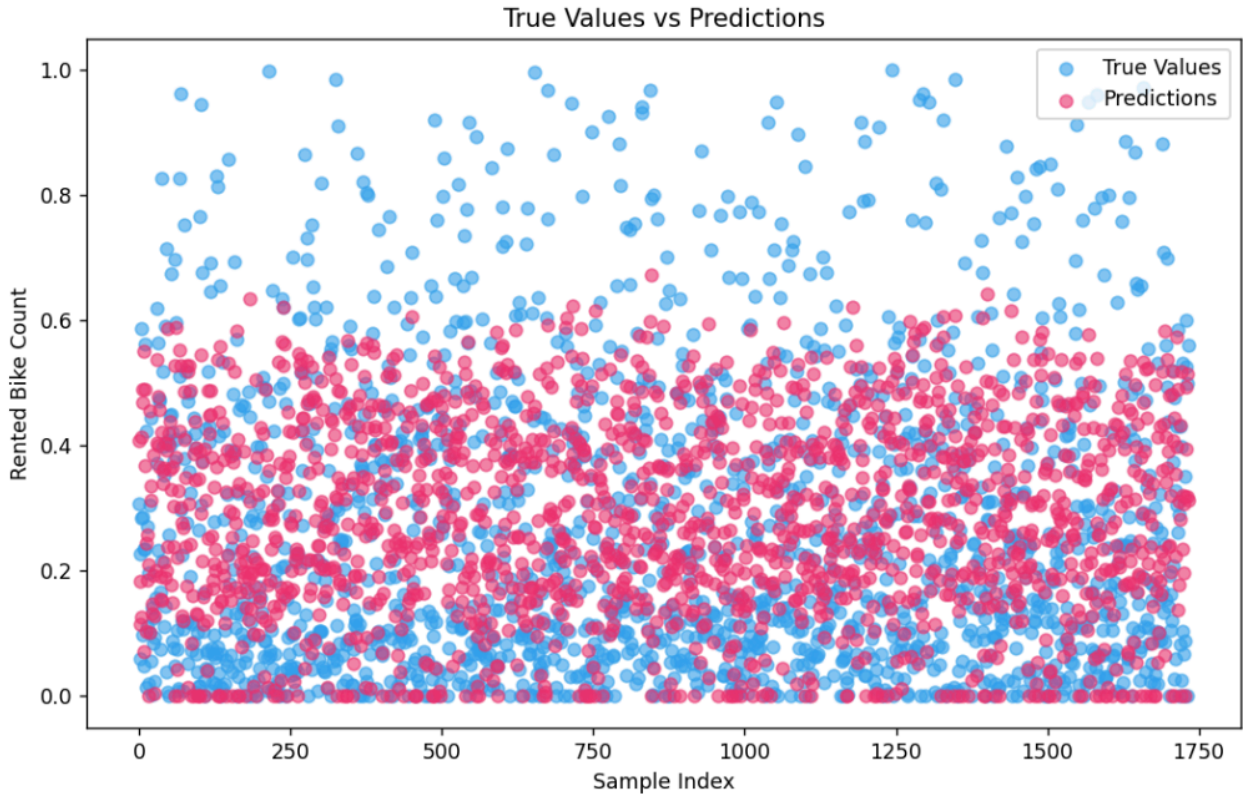
https://medium.com/@nahmed3536/the-motivation-for-train-test-split-2b1837f596c3
10. Masters in Data Science. (n.d.). Difference Between Bias and Variance. Retrieved from https://www.mastersindatascience.org/learning/difference-between-bias-and-variance/#:~:text=In%20linear%20regression%20analysis%2C%20bias,their%20output%20easier%20to%20understand.
11. Shukla, P. (n.d.). Random Initialization in Deep Learning. Medium. Retrieved from https://medium.com/@Coursesteach/deep-learning-part-27-random-initialization-b25ef8df8334#:~:text=Random%20initialization%20is%20crucial%20for,network%20architecture%20and%20activation%20functions.
12. Seltman, H. J. (n.d.). The Model Behind Linear Regression. Carnegie Mellon University. Retrieved from https://www.stat.cmu.edu/~hseltman/309/Book/chapter9.pdf
13. Shukla, P. (n.d.). Error Calculation Techniques for Linear Regression. Medium. Retrieved from https://medium.com/@shuklapratik22/error-calculation-techniques-for-linear-regression-ae436b682f90
14. Qlik Help. (n.d.). Scoring Regression Models. Retrieved from https://help.qlik.com/es-ES/cloud-services/Subsystems/Hub/Content/Sense_Hub/AutoML/scoring-regression.htm
15. IBM. (n.d.). Gradient Descent. Retrieved from https://www.ibm.com/topics/gradient-descent
16. V7 Labs. (n.d.). Train, Validation, and Test Set in Machine Learning. Retrieved from https://www.v7labs.com/blog/train-validation-test-set
17. GeeksforGeeks. (n.d.). Underfitting and Overfitting in Machine Learning. Retrieved from https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/
18. GeeksforGeeks. (n.d.). Bias vs Variance in Machine Learning. Retrieved from https://www.geeksforgeeks.org/bias-vs-variance-in-machine-learning/
19. IBM. (n.d.). Random Forest. Retrieved from https://www.ibm.com/mx-es/topics/random-forest

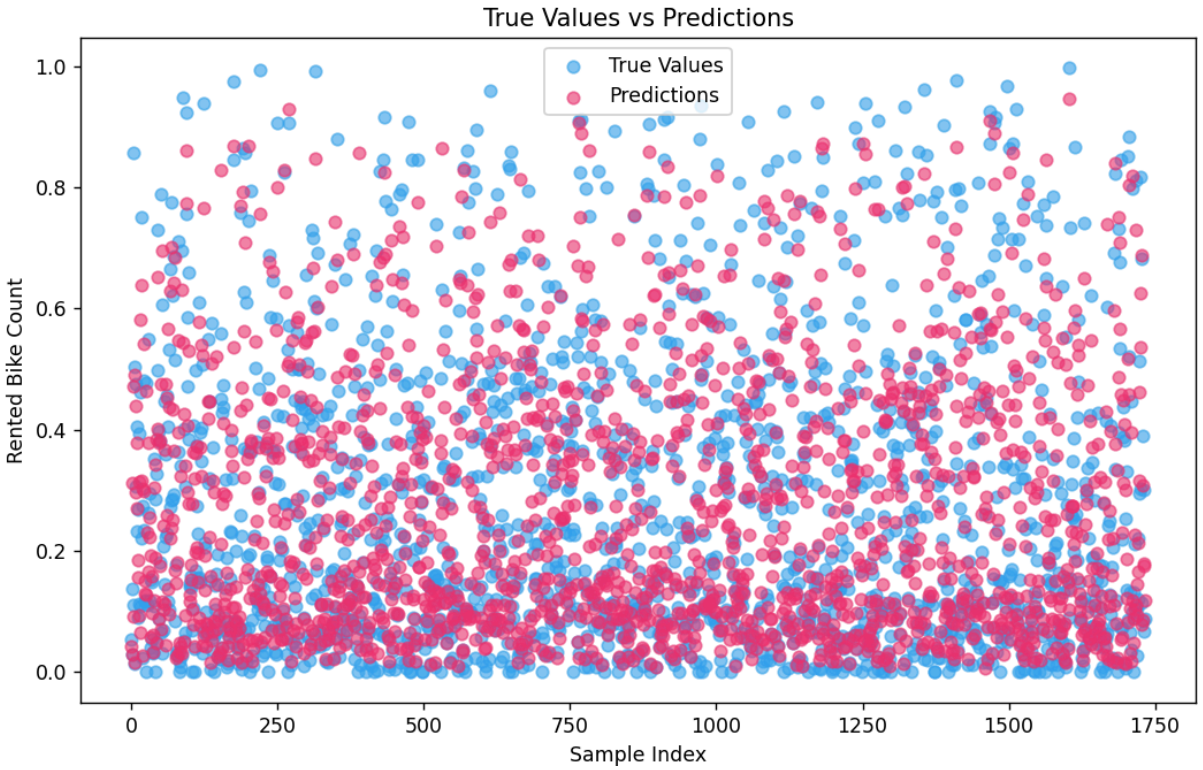
## 8. ANEXOS

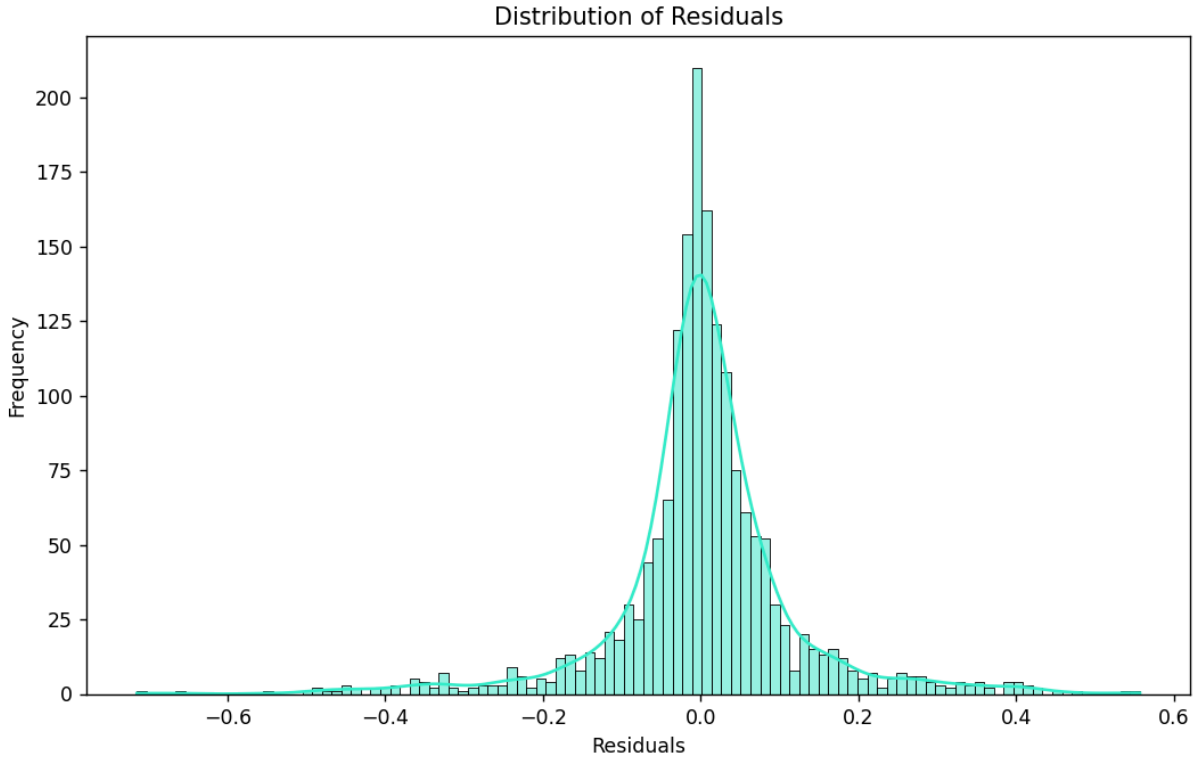

Correlation Matrix
Rented Bike Count
Hour
Temperature(C)
Humidity(%)
Wind speed (m/s)
Visibility (10m)
Dew point temperature(C)
Solar Radiation (MJ/m2)
Rainfall(mm)
Snowfall (cm)
1.00 0.40 0.54 -0.20 0.11 0.20 0.38 0.28 -0.13 -0.14
0.40 1.00 0.11 -0.24 0.28 0.09 -0.00 0.15 0.01 -0.02
0.54 0.11 1.00 0.17 -0.04 0.03 0.91 0.35 0.05 -0.22
-0.20 -0.24 0.17 1.00 -0.34 -0.54 0.54 -0.46 0.24 0.11
0.11 0.28 -0.04 -0.34 1.00 0.17 -0.18 0.33 -0.02 -0.00
0.20 0.09 0.03 -0.54 0.17 1.00 -0.18 0.15 -0.17 -0.12
0.38 -0.00 0.91 0.54 -0.18 -0.18 1.00 0.09 0.13 -0.15
0.28 0.15 0.35 -0.46 0.33 0.15 0.09 1.00 -0.07 -0.07
-0.13 0.01 0.05 0.24 -0.02 -0.17 0.13 -0.07 1.00 0.01
-0.14 -0.02 -0.22 0.11 -0.00 -0.12 -0.15 -0.07 0.01 1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

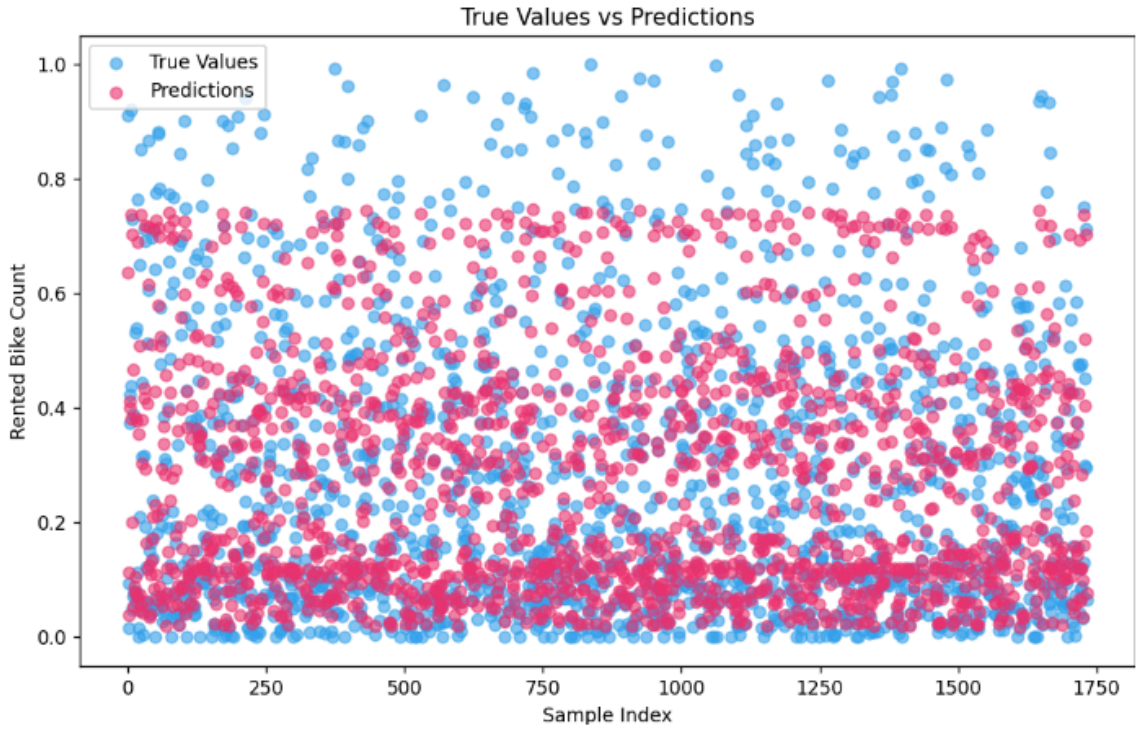

Rented Bike Count Distribution
Frecuencia
Rented Bike Count
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000
3500

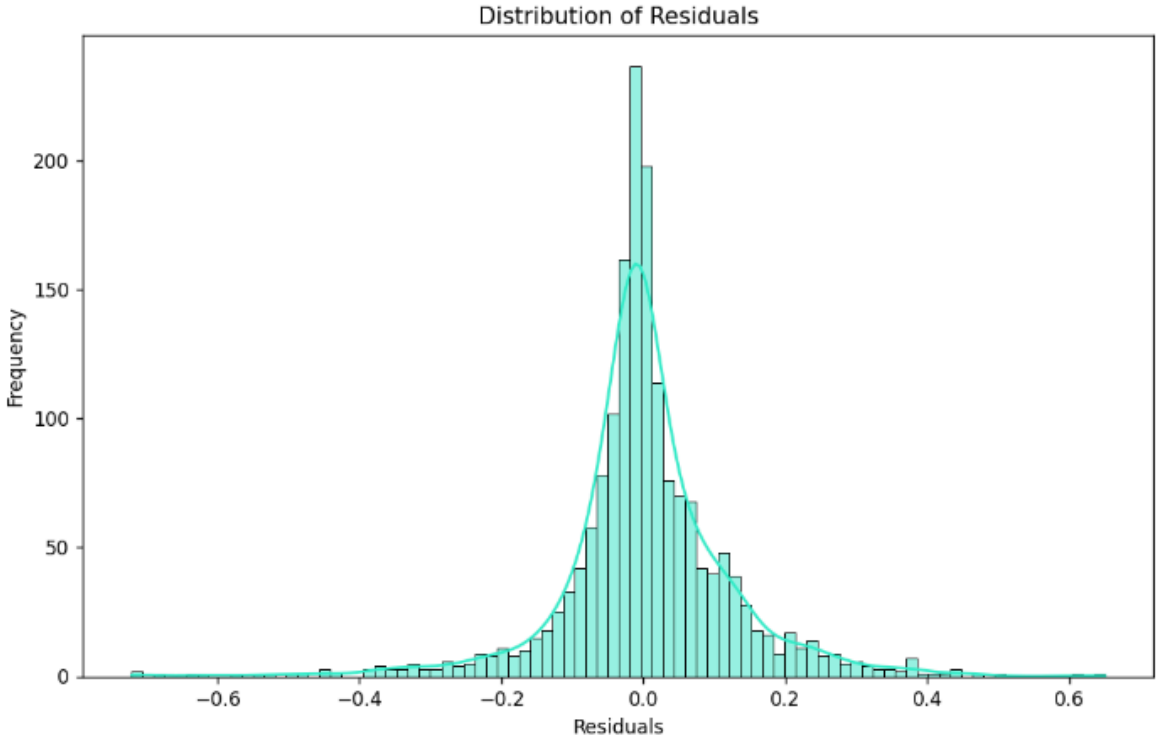

Train, Validation, and Test Accuracy Over Epochs
Accuracy
Epochs
0
−5
−10
−15
−20
0
2000
4000
6000
8000
10000
Train Accuracy
Validation Accuracy
Test Accuracy


True Values vs Predictions
Rented Bike Count
Sample Index
1.0
0.8
0.6
0.4
0.2
0.0
0
250
500
750
1000
1250
1500
1750
True Values
Predictions


True Values vs Predictions
True Values
Predictions
Rented Bike Count
Sample Index
0.0
0.2
0.4
0.6
0.8
1.0
0
250
500
750
1000
1250
1500
1750


Distribution of Residuals
Frequency
Residuals
0
25
50
75
100
125
150
175
200
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6


True Values vs Predictions
True Values
Predictions
Rented Bike Count
Sample Index
0.0
0.2
0.4
0.6
0.8
1.0
0
250
500
750
1000
1250
1500
1750


Distribution of Residuals
Frequency
Residuals
0
50
100
150
200
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6