Instituto Tecnológico y de Estudios Superiores de Monterrey
GDL Campus

Inteligencia artificial avanzada para la ciencia de datos I

Módulo 2: Aprendizaje automático

Miguel Angel Barrientos Ballesteros (A01637150)

Group 101

14 sept 2025

Análisis y Reporte sobre el desempeño del modelo KNN

Índice

Instituto Tecnológico y de Estudios Superiores de Monterrey..... 1
Análisis y Reporte sobre el desempeño del modelo KNN..... 2
Índice.....2
Introducción.....2
Sobre el modelo y el proyecto.....2
Sobre el dataset..... 3
Preparación de los datos, entrenamiento, y evaluación.....3
Inputs.....3
Ejemplo de inputs:.....3
Separación del conjunto de datos.....3
Estandarización.....3
Resultados..... 4
Explicación de outputs..... 4
Durante entrenamiento.....4
Durante predicción..... 4
Análisis..... 4
Bias o sesgo..... 5
Varianza.....5
Nivel del ajuste..... 6
Técnicas de ajuste de parámetros..... 6
Bibliografía..... 6

Introducción

El presente documento es un breve análisis del desempeño del modelo de K-Vecinos más cercanos (KNN) para regresión, implementado sin el uso de librerías o frameworks existentes de Machine Learning.

Dicha implementación se encuentra en el repositorio de Github público:
https://github.com/mrcolor-blind/ML_ModelAlgorithmImplementation

Sobre el modelo y el proyecto

Como se mencionó anteriormente, el modelo es una implementación simple de un algoritmo KNN, con el parámetro **k.** Actualmente el modelo solo soporta distancia euclidiana.

El repositorio incluye un archivo con la clase del modelo, otro archivo para correrlo y guardar sus resultados, así como un archivo haciendo lo mismo, pero utilizando **KNeighborsRegressor** de sklearn para hacer una comparativa. Además, se incluye un archivo **run.py** que orquestra la preparación de los datos y corre los modelos, con el objetivo de que el uso del proyecto sea intuitivo para el usuario, con un solo comando.

El proyecto permite entrenar al modelo utilizando una búsqueda grid para encontrar el mejor valor del hiperparámetro **k,** así como la utilización del modelo para predecir variables target dada una serie de inputs.

Sobre el dataset

El dataset utilizado en el proyecto es **California Housing Prices,** modificado para eliminar la columna categórica **ocean_proximity** por simplicidad. Los datos representan casas encontradas en California, junto con estadísticas obtenidas sobre ellas en un censo de 1990. La variable target del dataset es **median_house_value,** representando un aproximado del precio de la casa en cuestión.

En este proyecto se utiliza el modelo de KNN implementado para predecir el precio de una casa, dadas el resto de las variables en un csv.

Preparación de los datos, entrenamiento, y evaluación

Inputs

Al entrenar, el algoritmo recibe como input el dataset **housing.csv**, así como una lista de valores de **k** con el objetivo de hacer la búsqueda de rejilla.

Al utilizar el modelo, el algoritmo recibe como input el archivo **predictionInputs.csv,** conteniendo datos en el mismo formato que el dataset de entrenamiento, pero sin la columna target, con el objetivo de que el modelo regrese la predicción de los precios de estas entradas.

Separación del conjunto de datos

El proyecto, previo a el uso de la clase implementada **KNNRegressor,** hace una separación de los datos de entrada de la siguiente manera:

- **Train** (70%): usado para entrenar el modelo y almacenar las instancias.
- **Validation** (15%): usado para probar diferentes valores de **k** y, utilizando la búsqueda de rejilla con los valores obtenidos del input, seleccionar el mejor hiperparámetro, para posteriormente ser usado en para predicción.
- **Test** (15%): usado únicamente al final para evaluar el rendimiento real del modelo con datos nunca vistos, así como proporcionar los valores de MAE, RMSE, y R^2.

Estandarización

Ya que KNN se basa en distancias, se realiza una estandarización de los datos previo al entrenamiento con el objetivo de que todas las variables tengan la misma escala.

Para esto se calcula la media y desviación estándar del conjunto de entrenamiento, y con esos valores se transforman los tres conjuntos divididos (**train, validation, test**) para evitar fugas de información.

Resultados

Explicación de outputs

Durante entrenamiento

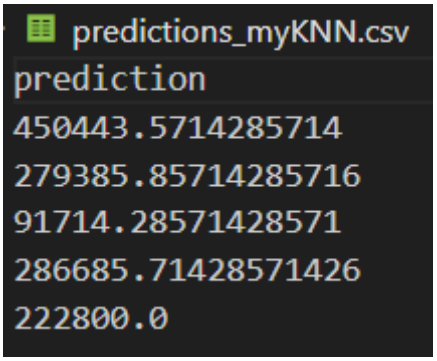
El algoritmo KNN implementado, durante la fase de **entrenamiento y evaluación**, devuelve en consola:

- **Mejor k**: valor de vecinos encontrado en la búsqueda en rejilla que ofrece el menor error en el conjunto de validación.
- **RMSE (Root Mean Squared Error)**: mide el error promedio, penalizando más los errores grandes.
- **MAE (Mean Absolute Error)**: mide el error promedio en las predicciones sin importar la dirección.
- **R^2 (Coeficiente de determinación):** indica qué proporción de la variabilidad del valor objetivo es explicada por el modelo (1 = perfecto, 0 = aleatorio).

Además, el algoritmo devuelve una imagen png, con un gráfico representando la curva de validación, con los ejes MSE y k, con el objetivo de comparar la curva de los conjuntos de entrenamiento y validación, y visualizar de manera más fácil el sesgo y la varianza del modelo.

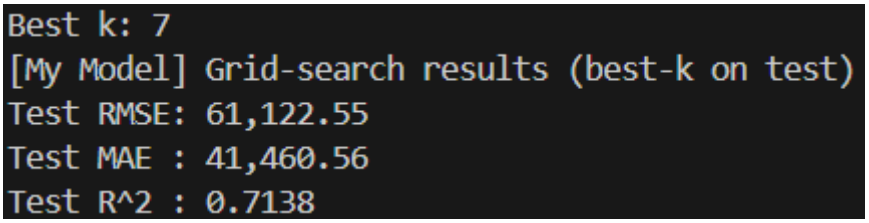
Durante predicción

El modelo, al usarlo para predecir los datos incluidos en **predictionInputs.csv** devuelve como salida el archivo **predictions_myKNN.csv,** el cual incluye una fila por cada fila en el input, representando la variable target **median_house_value,** es decir, la predicción del precio de las casas con los valores proporcionados.



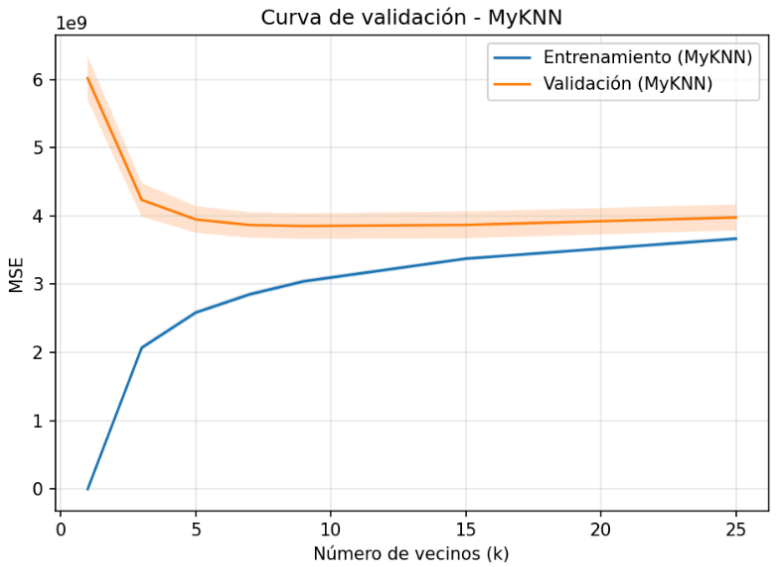
Análisis

A continuación se presenta el análisis de un ejemplo de los resultados obtenidos con el modelo, desde entrenamiento y evaluación, hasta las predicciones.



En este caso, se obtuvo que el mejor valor para **k** es 7. Utilizando este valor con el conjunto de **test**, se obtuvieron los valores mostrados, que pueden ser interpretados de la siguiente manera:

- **RMSE**: Significa que, en promedio, las predicciones del modelo se desvían del valor real en alrededor de 61 mil dólares. Como es una métrica cuadrática, penaliza más los errores grandes, lo que indica que puede haber casos aislados donde el modelo se equivocó bastante.
- **MAE**: Aquí el error promedio absoluto es de 41 mil dólares. Como esta métrica no está elevada al cuadrado, no se ve tan afectada por outliers como el RMSE. Lo que nos dice que si el modelo es utilizado con un **k** igual a 7, en promedio se va a equivocar por 41 mil dólares, arriba o abajo, en la predicción. En el contexto del dataset, es un número alto, pero esperable, y dado que los precios de las casas son altos en california, el modelo sigue siendo funcional con este promedio de error.
- **R^2:** El modelo explica alrededor del 71% de la variabilidad en los precios de las casas. Al estar por encima del 70%, el valor representa un buen nivel de ajuste, aunque lo deseado quizá sería que esté por encima de 80%, en el contexto de las casas sigue siendo útil para predecir datos.



El gráfico muestra el ajuste del modelo a medida que el valor de **k** aumenta, durante el uso de la búsqueda de rejilla.

Bias o sesgo

Primeramente, el coeficiente de determinación **R^2** muestra que el 29% de la variación de los precios no es explicada por el modelo. Esto indica que el sesgo del modelo es moderado, pues idealmente debería ajustarse un poco más.

Esta interpretación es apoyada por el gráfico de Curva de Validación, en el que, si se observan los valores de entrenamiento y validación cuando **k** es igual a 7 (el valor ideal, con el que se probó el modelo usando el conjunto de test), se identifica que se obtiene un valor de MSE ligeramente algo, indicando un sesgo un poco mayor del deseado.

Sin embargo, la curva también muestra que el error (y por ende el sesgo) tiene una tendencia a aumentar a medida que **k** es mayor, de manera que 7 es el mejor valor encontrado.

Varianza

La manera más fácil de observar un aproximado del nivel de varianza es mediante el gráfico. Se observa que el valor ideal encontrado de **k,** 7, proporciona un modelo en el que la distancia entre el MSE de entrenamiento y el MSE de validación no es tan pequeño como podría desearse, sugiriendo que el grado de varianza es relativamente alto.

Esto se confirma con que la diferencia entre el RMSE y el MAE obtenidos es alta, indicando que el modelo predice bien en promedio, pero es posible que en casos de outliers se equivoque mucho.

Nivel del ajuste

Tras haber analizado los grados de sesgo y varianza, se puede decir que el modelo no se subajusta, dado el sesgo moderado y el **R^2** es relativamente alto. De la misma manera, tampoco podría decirse que se sobreajusta, pues los errores siguen estando dentro de un rango esperado, y la varianza no es lo suficientemente alta como para representar un problema constante en las predicciones.

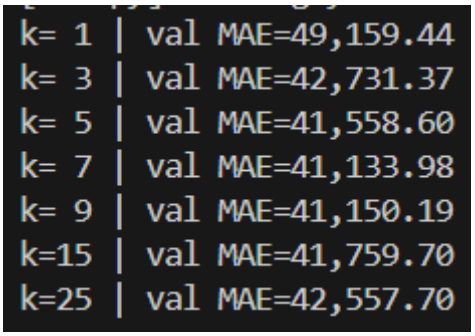
Esto nos lleva a concluir que el nivel de ajuste del modelo implementado de KNN es **fitt,** aunque no perfecto, pues lo ideal sería reducir tanto el sesgo como la varianza un poco más.

Técnicas de ajuste de parámetros

Debido a que la implementación de KNN para regresión fue simple, como se mencionó anteriormente, solo se implementó un tipo de distancia (euclidiana). Esto deja al modelo con un solo hiper parámetro: **k**.

La técnica de ajuste de parámetros que se realizó en este proyecto fue la búsqueda de rejilla. Esta corre el modelo múltiples veces, con diferentes valores de k, utilizando el conjunto de validación, para obtener los errores de cada iteración, con el objetivo de encontrar el mejor valor de **k**. Tras hacer esto, se evalúa el modelo una vez más, con el valor encontrado y con el conjunto de **test**, para obtener el error del modelo ya ajustado.

Para identificar cómo esta técnica mejora el desempeño del modelo, se imprimieron los errores encontrados con cada valor de **k** probado. Estos fueron los resultados en consola:



Observando esto, es evidente que el error más bajo obtenido con los valores del grid proporcionados en el input, es de **41,133.98,** cuando el valor de **k** es 7. De la misma manera, se puede visualizar cómo dicho desempeño es mejor que con valores no ajustados, en donde el error es mayor.

Bibliografía

Nugent, C. (2017). *California housing prices*. Kaggle. Recuperado de: https://www.kaggle.com/datasets/camnugent/california-housing-prices

mrcolor-blind. (n.d.). *ML_ModelAlgorithmImplementation* [Repositorio de GitHub]. GitHub. https://github.com/mrcolor-blind/ML_ModelAlgorithmImplementation

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830. Recuperado de: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html