INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

INGENIERÍA EN INTELIGENCIA ARTIFICIAL

APLICACIONES DE LENGUAJE NATURAL

6BM1

PRÁCTICA 2: Optimización de Hiperparámetros

BRIONES TAPIA SARAHÍ **2021630135**

LARRAZOLO BARRERA BRUNO **2021630367**

DESCRIPCIÓN DEL ANALÍSIS REALIZADO

- ∇ Análisis del conjunto de datos: “interview_dataset.csv” y preprocesamiento aplicado:

En esta parte del desarrollo de la práctica se realizó la lectura del archivo Word que se nos proporcionó en la asignación de classroom junto al dataset, con la finalidad de la interpretación de las columnas que conforman el conjunto de datos, para ir identificando las covariables que nos resultan candidatos potencialmente útiles para el objetivo que tenemos que cubrir con la práctica.

La primera parte del notebook que se nos compartió (del bloque 3 al 10) se encarga justamente de esta parte de la extracción de los datos, pues se aplica una técnica de “normalización” de valores mediante el cálculo de los percentiles y la reducción de valores atípicos, así como la eliminación de valores nulos.

Posteriormente en el apartado del notebook donde comienza el análisis correlacional también se aplica la matriz de correlación para todas las covariables. Esto se realizó para poder interpretar los datos y ver los coeficientes de correlación entre dos variables, con la finalidad de identificar que variables están más relacionadas y como es que los cambios en una variable pueden afectar a otra.

Posteriormente, al momento de comenzar a entrenar el modelo seleccionamos que nuestros datos de entrenamiento serían todas las columnas de datos, excepto las columnas: “producción”, “mes” y “año”, y seleccionamos como nuestra etiqueta la columna “producción” dado que es el valor que buscamos predecir.

- ∇ Análisis del modelo de Regresión Lineal:

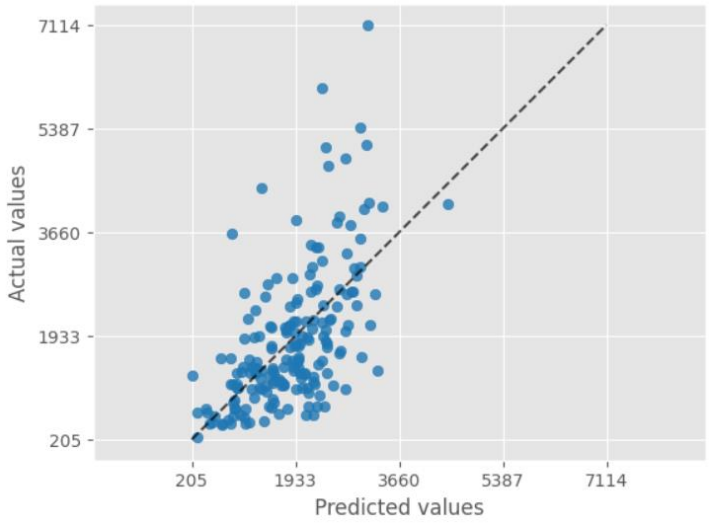
En el entrenamiento de este modelo observamos que son pocos los parámetros que podemos optimizar por medio de GridSearchCV, pues los posibles hiperparámetros a modificar son:

- fit_intercept: Parámetro para calcular la intersección de este modelo.
- copy_X: Si es Verdadero, se copiará X; de lo contrario, es posible que se sobrescriba.
- positive: Cuando se establece en True, fuerza que los coeficientes sean positivos. Esta opción sólo se admite para matrices densas.

Es importante detonar que en este modelo no logramos mejorar la precisión, pues luego de aplicar GridSearchCV y medir el score de la predicción, obtuvimos uno más bajo que el que venía en el notebook proporcionado en la asignación de classroom. En el notebook original se tenía un Testing score: 0.3728 y luego de nuestra implementación de GridSearchCV, nosotros obtuvimos un Testing score: 0.34753.

- Resultados obtenidos:

```
Training score:  0.4126986406813308
{'copy_X': True, 'fit_intercept': True, 'positive': False}
Testing score:  0.34752612082585577
```



∇ Análisis del modelo de Regresión basado en KNN y propuesta de optimización:

En este modelo hicimos la optimización de tres parámetros, dado que analizando la documentación que nos proporciona scikit learn nos dimos cuenta de que estos eran los que posiblemente tendrían más peso y mayor influencia en el rendimiento del modelo.

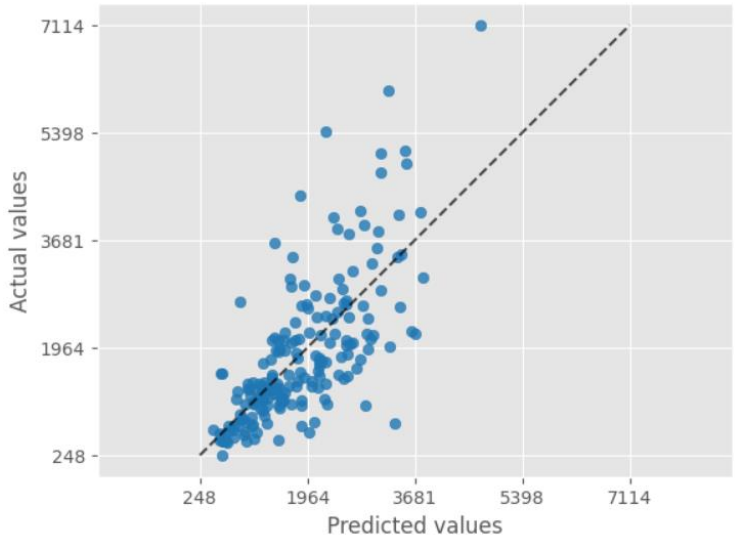
Los hiperparámetros optimizados fueron:

- n_neighbors: Número de vecinos a utilizar.
- weights: Función de peso utilizada en la predicción.
- metric: Métrica que se utilizará para el cálculo de la distancia.

para lo cual creamos una cuadrícula con los posibles valores de estos parámetros y a través de GridSearchCV buscamos la mejor combinación de estos para obtener el mejor resultado de rendimiento del modelo.

- Resultados obtenidos:

```
{'metric': 'manhattan', 'n_neighbors': 7, 'weights': 'distance'}
Testing score:  0.5064866948848388
```



∇ Análisis del modelo de Árboles de Decisión y propuesta de optimización:

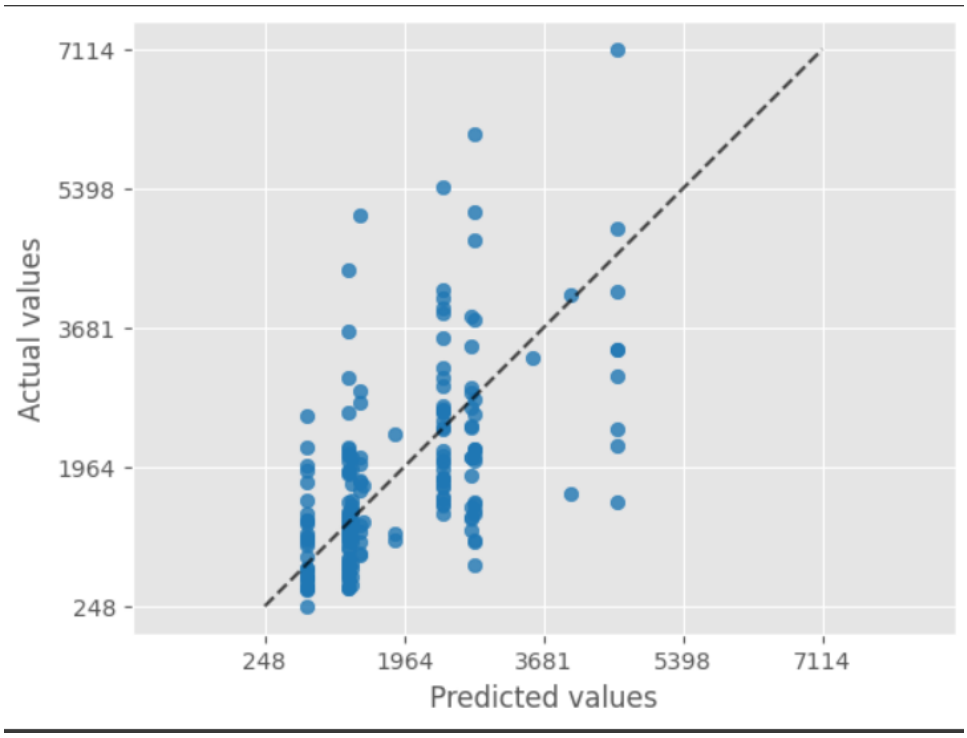
En este modelo ocupamos los siguientes hiperparámetros:

- **Criterion:** Función para medir la calidad de una división.
- **Splitter:** Estrategia utilizada para elegir la división en cada nodo.
- **Max_depth:** Profundidad máxima del árbol.
- **min_samples_split:** Número mínimo de muestras necesarias para dividir un nodo interno
- **min_samples_leaf:** Número mínimo de muestras necesarias para estar en un nodo de hoja.
- **min_weight_fraction_leaf:** Fracción ponderada mínima de la suma total de pesos (de todas las muestras de entrada) requerida para estar en un nodo hoja.
- **max_features:** Cantidad de características a considerar al buscar la mejor división

El método para seleccionar la mejor combinación de hiperparámetros fue atreves de GridSearch, obteniendo los siguientes resultados:

```
R2 Score: 0.2620889020017425
```

```
Mejor combinación de hiperparámetros: {'criterion': 'squared_error', 'max_depth': 4,
'max_features': 11, 'min_samples_leaf': 9, 'min_samples_split': 9, 'min_weight_fraction_leaf': 0.0,
'splitter': 'best'}
```



Conclusión:

En esta práctica pudimos solucionar uno de los problemas más comunes a la hora de utilizar modelos de Machine Learning, el cuál es la selección de hiperparámetros, para el cuál utilizamos GridSearch, además, aplicamos un preprocesamiento a los datos que nos ayudó a mejorar el desempeño del modelo de regresion de KNN, siendo este el que presentó mejores resultados.