UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MECÁNICA

APRENDIZAJE DE MÁQUINAS PARA LA PREDICCIÓN DEL BAND GAP EN PANELES TIPO SÁNDWICH

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MECÁNICO

VICENTE JOEL QUEZADA COFRÉ

PROFESORA GUÍA:
VIVIANA MERUANE NARANJO

PROFESOR CO-GUÍA:
RAFAEL RUIZ GARCÍA

COMISIÓN:
RUBÉN FERNÁNDEZ URRUTIA

SANTIAGO DE CHILE
2022

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL MECÁNICO
POR: VICENTE JOEL QUEZADA COFRÉ
FECHA: 2022
PROF. GUÍA: VIVIANA MERUANE NARANJO

APRENDIZAJE DE MÁQUINAS PARA LA PREDICCIÓN DEL BAND GAP EN PANELES TIPO SÁNDWICH

En la naturaleza nos podemos encontrar con estructuras de baja densidad, pero con una alta rigidez y resistencia. Las nuevas técnicas de manufactura han permitido replicar estas estructuras, fabricando así paneles tipo sándwich al unir dos capas externas con un núcleo ultraliviano entre ellas. Esto le permite al panel soportar altas cargas en relación a su densidad.

Otra importante propiedad que se puede dar bajo ciertas condiciones en estos paneles es la capacidad de reducir la propagación de ondas mecánicas en el material para cierto rango de frecuencias, más conocido como band gap. Es esta la propiedad que se desea predecir utilizando algoritmos de aprendizaje de máquinas.

Primero se debe generar una base de datos a partir de un modelo de elementos finitos, en el cual se utilizan distintas topologías. Dicha base de datos se divide en dos datasets; entrenamiento y pruebas. Luego se deben entrenar los algoritmos, siendo este un problema de regresión se implementa *Support vector regression* y *Gaussian process regression*, cuyos hiperparámetros y funciones son modificados usando una estrategia de grid search. El dataset de validación es utilizado para evaluar las distintas combinaciones de dichos parámetros. Mientras que el dataset de entrenamiento es utilizado para entrenar los modelos durante la selección de estos. Finalmente se realiza una evaluación utilizando el dataset de pruebas.

Para el cumplimiento de este trabajo se cuenta con *Python* como lenguaje de programación. *SkLearn* como librería de modelos de aprendizaje de máquinas. *Numpy* y *Pandas* para ordenar y procesar los datos.

En conclusión, se logra desarrollar modelos con ambos algoritmos capaces de predecir el band gap y sus características en paneles de tipo sándwich. Luego de analizar distintos enfoques, se determina que una predicción directa de ancho de banda y frecuencia media del band gap obtiene mejores resultados. Finalmente, se realiza un análisis sobre los datasets utilizados y la cantidad de casos con band gap, del cual se recomienda para trabajos futuros investigar el desempeño de los modelos frente a datasets con mayor cantidad de casos con band gap.

A mi mamá y papá,
por apoyarme en todo.

Ama

Agradecimientos

Agradezco a mi familia que me acompañó durante estos seis años de estudio. Sin duda me ayudaron a no perder de vista lo realmente importante, su apoyo y mi gratitud va más allá de mi capacidad de expresarlo de forma escrita.

Agradezco a mis amigos que me han acompañado y estado presente desde el colegio, con ellos mi estadía en una ciudad foránea no habría sido tan amena y llena de buenos recuerdos.

Agradezco a mis nuevos amigos y compañeros que en el futuro serán mis colegas. Las infinitas horas de estudio, las ventanas y tiempos de ocio no habrían sido lo mismo sin su compañía y amistad.

Agradezco al Estado de Chile el cual ha financiado la totalidad de mi carrera en la Universidad de Chile.

Gracias.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

La manufactura aditiva ha permitido la creación de estructuras cada vez más complejas, empleando a su vez una gran variedad de materiales, tanto cerámicos como metálicos y polímeros [1]. Esto permite la confección de materiales, además de estructuras y moldes. Entre estos destacan los paneles tipo sándwich, los cuales son creados a partir de una delgada capa y un núcleo liviano más grueso, el cual es encuentra entre ambas capas.

Sus propiedades son extremadamente deseables en aplicaciones que requieren materiales multifuncionales como, por ejemplo, material estructural ultraliviano en aplicaciones aeroespaciales [2]. Además, en algunos casos estos poseen propiedades vibro acústicas únicas, ya que son capaces de inhibir la propagación de ondas mecánicas en el material para cierto rango de frecuencias, más conocido como band gap [3].

1.1. Motivación

El diseño de paneles tipo sándwich con núcleos ultralivianos es particularmente complejo y requiere de gran poder computacional. Es por esto que se opta por la construcción de modelos sustitutos a partir del aprendizaje de máquinas, utilizando diferentes algoritmos capaces de resolver el problema de regresión. En particular se trabaja con un modelo de elementos finitos, para generar una basta base de datos, con las cuales se entrenan los algoritmos. Estos reciben parámetros relacionados con la topología, tales como la geometría y el material del núcleo. Para finalmente entregar la frecuencia promedio y el ancho del band gap.

1.2. Objetivos

Objetivo general

Proponer y entrenar algoritmos de aprendizaje de máquinas capaces de predecir el band gap y sus características en paneles tipo sándwich con núcleos ultralivianos.

Objetivos específicos

- Procesar la base de datos de estructuras de paneles tipo sándwich, de tal forma que sea apta para el entrenamiento de los algoritmos de aprendizaje de máquinas.
- Implementar algoritmos de aprendizaje de máquinas, estos son: Support vector regression y Gaussian process regression.
- Determinar la metodología y configuración idónea para la predicción de band gaps utilizando los algoritmos anteriormente mencionados.
- Evaluar el rendimiento de los algoritmos de aprendizaje de máquinas para la predicción de band gaps.

1.3. Alcances

El presente trabajo comprende la creación de una base de datos a partir de un modelo parametrizado preexistente. Luego, se implementan algoritmos de aprendizaje de máquinas capaces de resolver el problema de regresión. En específico *Support vector regression* y *Gaussian process regression*, cabe mencionar que el análisis está enfocado en el desempeño y resultados obtenidos de dichos modelos. Por lo tanto, el trasfondo matemático y computacional de estos no serán abordados.

1.4. Recursos disponibles

Para desarrollar y trabajar en este título se utiliza el computador de escritorio personal con las siguientes especificaciones:

- Windows 10 Home.
- AMD Ryzen 7 3700X 8-Core Processor 3.260 GHz.
- 32Gb RAM.
- NVIDEA GeForce GTX 1660 Ti

Además, se utiliza el siguiente software y librerías:

- **Python 3.7** como lenguaje de programación.
- **Scikit-Learn** como librería de algoritmos de aprendizaje de máquinas.
- **Numpy y Pandas** orden y procesamiento de los datos.
- **Matplotlib** para visualizar los datos y resultados.

Capítulo 2

Antecedentes

2.1. Materiales Celulares

Los materiales celulares reciben su nombre al ser formados por arreglos periódicos de celdas, tanto cerradas como abiertas. Estos son comunes en la naturaleza y se caracterizan por su baja densidad con una alta resistencia y rigidez. [1]. Como por ejemplo en los huesos de las alas de un ave observable en la figura 2.1.1.

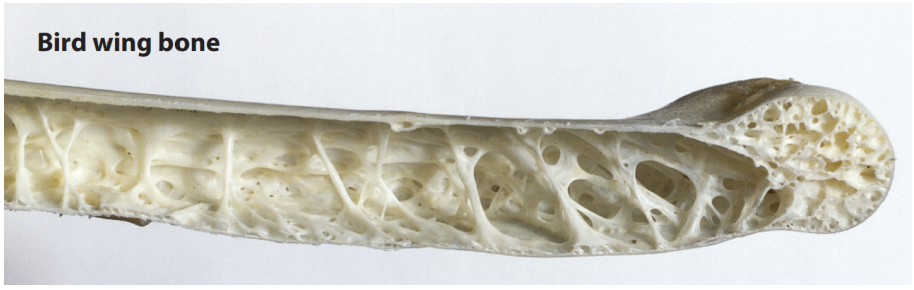


Figura 2.1.1: Materiales celulares en la naturaleza [1].

Las propiedades de estos están dadas principalmente por el sólido constituyente y la arquitectura celular, es decir, su configuración espacial tanto de sólidos como de vacíos. Hoy en día los avances en la manufactura aditiva han permitido la creación de arquitecturas celulares de mayor complejidad.

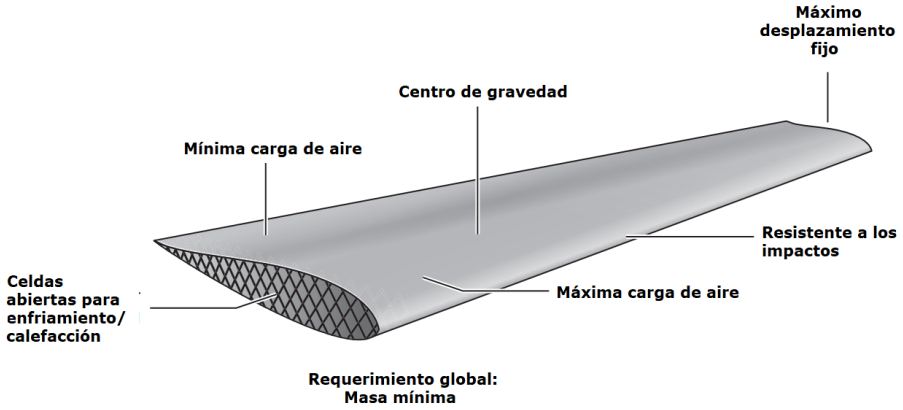


Figura 2.1.2: Múltiples requerimientos logrados vía material celular [1].

Adicionalmente los avances y estudios en la optimización de la topología han permitido que la arquitectura de materiales celulares logre entregar un material con las características deseadas, esto es, materiales ligeros con alta rigidez y resistencia a la flexión. Además de otros requerimientos como se observa en la figura 2.1.2.

Entre las aplicaciones más importantes de estos materiales se encuentran los paneles tipo sándwich con núcleo ultraliviano.

2.2. Paneles Sándwich

Los paneles tipo sándwich son estructuras formadas por múltiples capas. Estas generalmente son dos láminas delgadas de alta rigidez y densidad en el exterior, las cuales rodean un núcleo ultraliviano de baja densidad.

Los parámetros más importantes de diseño de un panel son los materiales del núcleo y las láminas, sus espesores, la topología del núcleo. Además, el espesor relativo y la distribución de volúmenes [3].

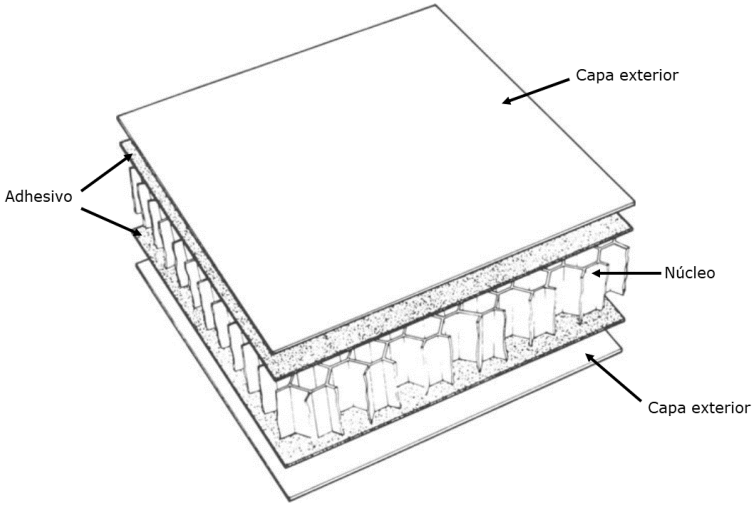


Figura 2.2.1: Esquema de panel tipo sándwich [4].

Gracias a este arreglo de materiales se logra una mayor resistencia a la flexión, en comparación a utilizar una sola placa del mismo material y masa. Esto se debe a que las capas exteriores soportan las cargas de flexión y compresión.

En cuanto al núcleo es lo suficientemente rígido a lo largo como para mantener la distancia entre capas constante. Además, la rigidez en corte es tal que, al doblarse el panel, no ocurran deslizamientos de las placas.

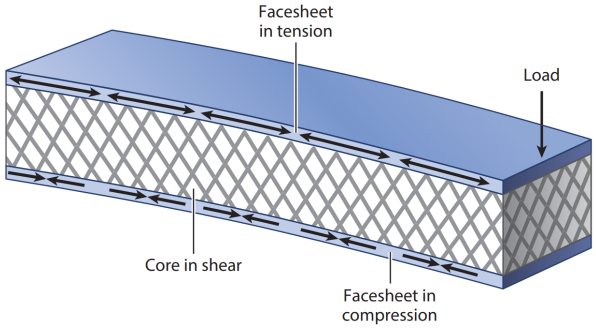


Figura 2.2.2: Distribución de esfuerzos en un panel [1].

2.3. Cristales Fonónicos

Los cristales fonónicos son metamateriales de arquitectura periódica, cuya principal propiedad es impedir la propagación tanto de ondas mecánicas como acústicas, determinado principalmente por la dirección y frecuencia de onda [5]. Factores como el material, tamaño y espaciado, determinan el comportamiento de dichas ondas al interactuar con los cristales [6].

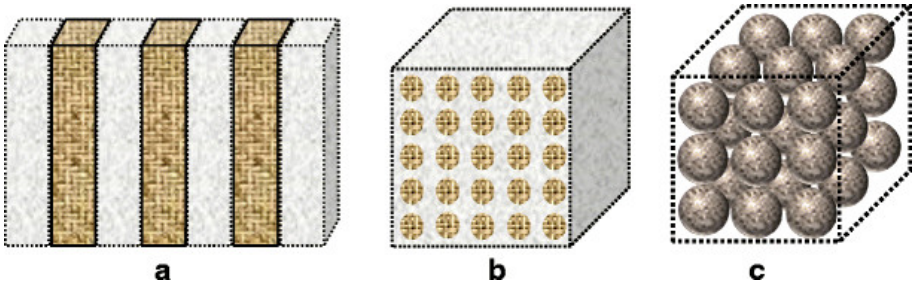


Figura 2.3.1: Cristales fonónicos con periodicidad en (a) una, (b) dos, y (c) tres dimensiones [6].

La periodicidad de estas estructuras como muestra la figura 2.3.1, es capaz de llegar a tres dimensiones. Luego para determinar las bandas de frecuencias en las cuales se impide la transmisión de ondas mecánicas, se debe modelar los cristales como una red periódica infinita.

Esta particular propiedad permite su aplicación como aislantes acústicos y la supresión de vibraciones. Por ejemplo, en sistemas de alta precisión, para guiar y filtrar frecuencias [7].

2.4. Band Gaps Fonónicos

Los band gaps fonónicos son las bandas de frecuencia en la cual las ondas mecánicas son completamente suprimidas. Estas son observadas en los cristales fonónicos mencionados anteriormente.

El efecto se produce por la combinación de la difracción de *Bragg* y el *Scattering* de las ondas, ambos se producen debido a la geometría y su periodicidad [8], incapacitando la propagación de la onda en cualquiera de las direcciones.

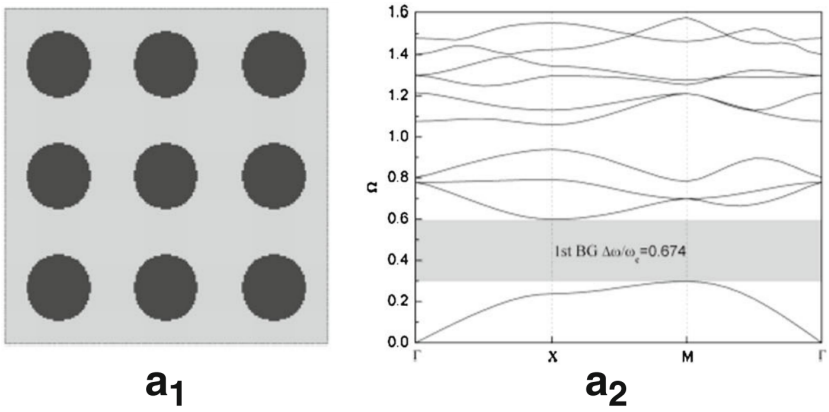


Figura 2.4.1: Estructura compuesta de Pb/Epoxy (a_1), junto con su diagrama de banda (a_2) [9].

En la figura 2.4.1, se observa como la estructura periódica (a_1) compuesta de plomo y epoxy posee un diagrama de bandas (a_2), donde el eje de las ordenadas muestra la frecuencia normalizada y el eje de las abscisas muestra el vector de ondas. Entre las dos primeras bandas en el diagrama existe una región donde no se presentan frecuencias, esto corresponde al band gap. Su ancho es la diferencia entre el valor máximo de la banda inferior y el mínimo de la banda superior. Luego su frecuencia media es igual al máximo de la banda inferior más la mitad del ancho. Este diagrama presenta un band gap enmarcado en gris entre los valores 0.3 y 0.6 de frecuencia, por lo tanto posee un ancho de band gap igual a 0.3 y su frecuencia media es de 0.45.

El ancho en frecuencias observado es modificado a medida que se ajusta la periodicidad y la proporción de las fases en la estructura. Hao-Wen Dong [9] al optimizar la topología logra un band gap mayor utilizando los mismos materiales.

2.5. Aprendizaje de Máquinas

El aprendizaje de máquinas es un campo de la inteligencia artificial, cuya motivación es generar programación inteligente. Lo cual significa que las tareas resueltas no se encuentran plasmadas directamente en el código.

2.5.1. Aprendizaje supervisado

El aprendizaje supervisado es una técnica del aprendizaje de máquinas, en la cual se entrena un algoritmo a través de ejemplos, donde por cada uno de estos ejemplos el algoritmo recibe como entrada un vector de valores. Estos son caracterizados junto con su valor esperado de salida, mientras que el proceso de aprendizaje encuentra la relación entre estas variables.

En el fondo, el algoritmo cuantifica la diferencia entre la predicción y el valor real. Dado lo anterior, se genera un mapa desde la base de datos a su etiqueta, luego la función de pérdida cambia dependiendo de los datos entregados y la tarea a resolver [10].

2.5.2. Métodos de Regresión

En todo modelo de aprendizaje de máquinas se debe definir el tipo de resultado o respuesta que se espera de este. Los ejemplos clásicos de tipos de respuesta son 3: clasificación binaria, clasificación multiclase y de regresión.

Para el caso de estudio de este trabajo se requieren modelos de regresión, ya que estos entregan una predicción o valor real como su respuesta.

2.5.2.1. Support Vector Regression

Regresión de soporte vectorial o **SVR** por sus siglas en inglés (Support Vector Regression) es un modelo de aprendizaje de máquinas que implementa el principio de minimización de riesgo inductivo para obtener una buena generalización en un número limitado de patrones de aprendizaje [11].

Al tener un set de entrenamiento $\mathcal{L}$, la meta del método es predecir el valor de resultado $\hat{Y}$ de cada elemento del set de entrenamiento, el cual posee a lo más una desviación ϵ del valor real Y [11]. Finalmente la predicción se calcula como:

$$\hat{Y}(X) = K(w, X) + b \tag{2.1}$$

Donde b es una constante, K es una función de kernel y w es un vector que comparte el mismo espacio dimensional de X.

Existen distintos tipos de SVR, estos pueden ser de regresión lineal y no lineares. Esto hace referencia a la función kernel utilizada, en la primera es un kernel lineal y en los del último tipo pueden utilizar distintas funciones, entre las más comunes está la Gaussian Radial Basis o RBF.

El modelo de **SVR** es implementado en python utilizando la librería de *Scikit-Learn*[12], la

cual se especializa en aprendizaje de máquinas. Este se define por múltiples hiperparámetros, tales como el mencionado anteriormente kernel, además de gamma, C y épsilon. Son estos parámetros los que se deben optimizar a la hora de encontrar la configuración óptima del modelo y así obtener una regresión con mejor desempeño.

2.5.2.2. Gaussian Process Regression

Un proceso gausiano o Gaussian Process (GP) es una colección de variables aleatorias donde un número finito de ellas posee una distribución gausiana. Un GP está definido por $K(x, x')(y \sim GP(m, K))$ donde $m(x)$ es el promedio [13].

Al considerar un set de entrenamiento $\mathcal{L}$ con X e Y como sets de input y output respectivamente, mientras que los X_* y Y_* son los sets de input y output de validación. Luego se asume que los datos están representados por una distribución gausiana, por lo que se representa como sigue:

$$\begin{bmatrix} Y \\ Y_* \end{bmatrix} = N\left(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} \sum & \sum_* \\ \sum_*^T & \sum_{**} \end{bmatrix} \right) \tag{2.2}$$

Donde μ y μ_* son los promedios del set de entrenamiento y validación. Mientras que $\sum$, $\sum_*$ y $\sum_{**}$ son la covarianza de los set de entrenamiento, validación y entrenamiento-validacion respectivamente. Luego se define el GP a continuación:

$$\begin{gathered} Y|L = GP(m_L, K_L) \\ m_L(x) = m(x) + \sum(X, x)^T \sigma^{-1}(Y - m) \\ K_L(x, x') = K(x, x') - \sum(X, x)^T \sum^{-1} \sum(X, x') \end{gathered} \tag{2.3}$$

Donde $\sum(X, x)$ es la covarianza entre cada muestra del entrenamiento y x. Finalmente el output se puede predecir como:

$$Y(x) = g(x) + \varepsilon \tag{2.4}$$

Con $\varepsilon \sim N(0, \sigma_n^2)$, $g \sim GP(m, K)$, además, $Y \sim GP(m, K + \sigma_n^2 \delta_{ii'})$. Donde $\delta_{ii'} = 1$ si $i = i'$.

Al igual que el modelo anterior, **GPR** es implementado utilizando la misma librería de python[14]. No obstante, este posee una menor cantidad de hiperparámetros contando solo con el kernel, cuya primordial tarea es computar la covarianza del *GP* entre los puntos de datos.

2.5.3. Métricas de evaluación

2.5.3.1. Coeficiente de determinación

El *coeficiente de determinación* o mejor conocido como R^2 representa la proporción de varianza de los datos y su predicción, con esto se tiene un indicador de que tan buena es la regresión y también proporciona una medida de que tan bien un caso nunca antes visto será predicho. A continuación se define R^2 tal como es utilizado en la misma librería de python anteriormente mencionada[15]:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{2.5}$$

Para n casos, se tiene que y_i es el valor real y que $\hat{y}_i$ es el valor predicho del i-esimo caso. Además, $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ y $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \epsilon_i^2$. El mejor valor posible de R^2 es 1, mientras que una predicción de valor constante obtiene un valor igual a 0, por otro lado, este puede obtener un valor negativo, ya que, la predicción puede ser arbitrariamente peor.

2.5.3.2. Raíz del error cuadrático medio

La raíz del error cuadrático medio o **RMSE** por sus siglas en inglés, mide el error entre la predicción y los valores reales. Este se encuentra en las mismas unidades, por lo que en general un menor valor RMSE significa una buena predicción.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.6}$$

Donde y_i es el valor real y que $\hat{y}_i$ es el valor predicho del i-esimo caso, mientras que n es la cantidad total de casos.

2.5.4. Selección de hiperparámetros

Tanto **SVR** como **GPR** requieren una selección exhaustiva de sus parámetros con tal de obtener el desempeño óptimo, no obstante, existe una gran cantidad de combinaciones posibles además de un vasto rango para cada uno. Por lo tanto, se opta por utilizar una estrategia de automatización de este proceso, la cual se llama *Gridsearch*[16]

Esta consiste en evaluar todas las combinaciones posibles dentro de un cierto rango para cada parámetro. Por ejemplo, para el caso de SVR se tienen 4 hiperparámetros que se desean optimizar; C, gamma, épsilon y kernel. Los primeros tres poseen un valor numérico perteneciente a los reales, mientras que el kernel es determinando por distintas funciones.

En cuanto a GPR, este posee la ventaja de utilizar solo el kernel como hiperparámetro. Sin embargo, existen múltiples combinaciones, por lo que la búsqueda se centra en los provistos por la librería y sus combinaciones más comunes dentro de la literatura.

Capítulo 3

Formulación

En la primera sección de este capítulo se presentan y describen las estructuras utilizadas en este trabajo, estas corresponden a paneles tipo sándwich con núcleos ultralivianos que han sido definidas y estudiadas en el trabajo de título de *Vicente Gálvez S.* titulado *"Desarrollo de modelos en elementos finitos parametrizados de paneles tipo sándwich con núcleos ultralivianos para predecir band gaps"*. En donde se encarga de la modelación FEM y parametrización de las distintas topologías a estudiar, estas han sido creadas utilizando el software matemático *Matlab*. Además, cabe mencionar que la base de datos es creada a partir de estos modelos y posteriormente proporcionada.

Finalmente, en la segunda sección del capítulo se presenta el cálculo de band gap y su ancho a partir del diagrama de bandas, cuya obtención es transversal a las tres estructuras presentadas.

3.1. Estructuras

Se consta de tres estructuras de panel tipo sándwich que serán principalmente denominadas por el enrejado del núcleo, estas son; pirámide cuadrada, Kagome tridimensional y pirámide central. La materialidad y por ende propiedades de estas son las mismas y corresponden a las del acero, por lo que su *Modulo de Young* es $E = 2,1 \cdot 10^{11}[Pa]$, *Coeficiente de Poisson* es $\nu = 0,3$ y su *Densidad* es $\rho = 7800[\frac{Kg}{m^3}]$.

Otra característica compartida por las tres estructuras es la geometría de celdas base, estas se componen de 9 celdas base repartidas tanto en el eje de abscisas X como el de ordenadas Y, siendo cada una de ellas cúbica o tridimensional también se ubican en el eje Z.

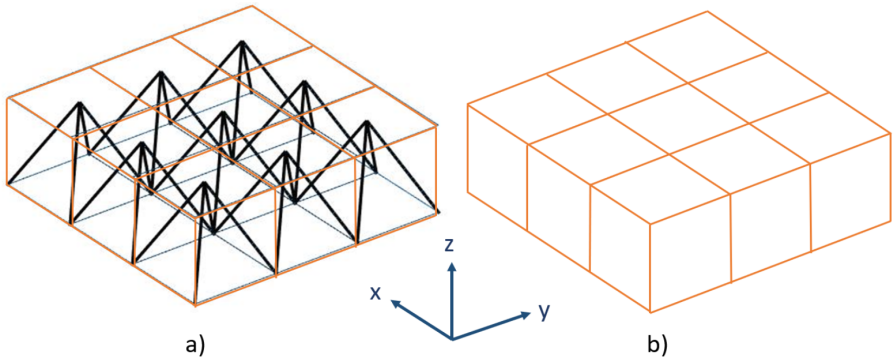


Figura 3.1.1: a) Visualización estructura de pirámides en celdas base. b) Vista tridimensional de celdas base.

Tal como se observa en la figura 3.1.1 cada celda en particular alberga la geometría principal, por ejemplo, en la primera estructura se utiliza una pirámide cuadrada. Además, el número de celdas base en cada eje se determina por n_x, n_y y n_z, donde $n_c = n_x = n_y$ conformando así un plano X-Y cuadrado, mientras que $n_z = 1$ lo que significa que solo se utiliza una única capa de núcleo. Por otra parte, se tiene L_x, L_y y L_z representando estos el ancho, largo y alto respectivamente. Igual que para la cantidad de celdas base, se conserva la igualdad $L_c = L_x = L_y$ y $L_z = L_c/3$ de tal forma de obtener una celda base cúbica.

3.1.1. Pirámide Cuadrada

La estructura de pirámide cuadrada utiliza la geometría anteriormente mencionada de celdas base, en la figura 3.1.2 se observa la vista del plano XY con $n_c = n_x = n_y = 3$, por ende se cuenta con 9 celdas base cúbicas. Además, se observan cruces diagonales dentro de los cuadrados o celdas base, estos representan a las pirámides.

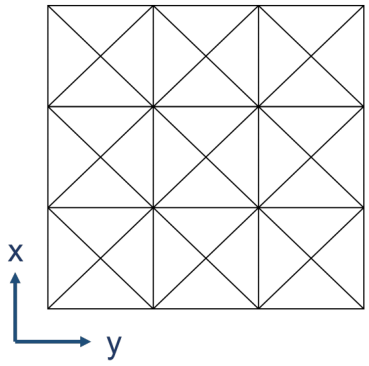


Figura 3.1.2: Vista celdas base en el plano XY de una estructura tipo pirámide cuadrada.

Es en este plano donde se impone la simetría en 3 ejes; vertical, horizontal y diagonal en 45°. A partir de esta simetría se genera la *zona irreducible de Brillouin* o *IBZ* por sus siglas en inglés. Ella consiste en 6 elementos que se repiten mediante las simetrías mencionadas y se distinguen a través de distintos colores, lo que significa que los elementos con el mismo color en las celdas base poseen las mismas propiedades a través de la estructura. Finalmente

la estructura se define con 36 elementos, es decir, las 9 celdas bases poseen 4 elementos cada una.

A continuación, en la figura 3.1.3 se representan las simetrías generadoras de la IBZ destacada en el triángulo gris en conjunto con cada elemento del núcleo y su esquema de color.

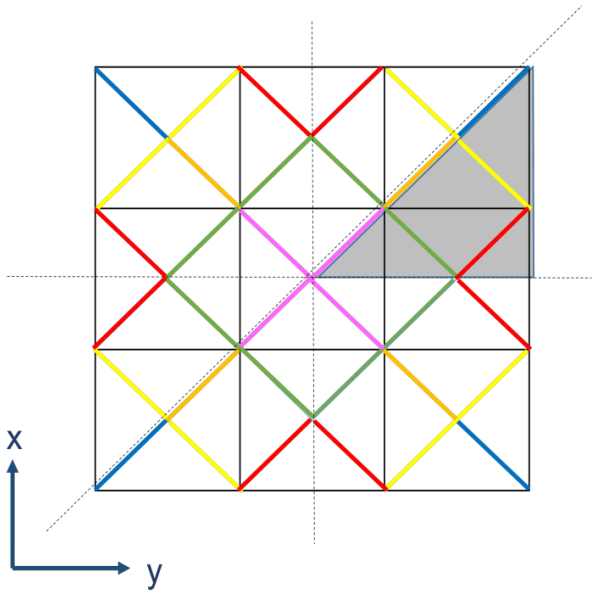


Figura 3.1.3: Vista celdas base en el plano XY de una estructura tipo pirámide cuadrada con líneas de simetría y zona de IBZ marcadas.

En base a esto la estructura es representada finalmente por los 6 elementos de la IBZ con su factor de radio de la sección transversal, estos factores se denotan como f_1, f_2, f_3, f_4, f_5 y f_6. Además, se utiliza un factor altura representado por f_L, el cual modifica la distancia o altura entre las placas exteriores del panel, por ende, modifica a su vez la altura de las pirámides. Finalmente, los rangos de valores de dichas variables y factores son:

- **Materialidad:** Los núcleos son de acero, por lo que su *Modulo de Young* es $E = 2,1 \cdot 10^{11}[Pa]$, *Coeficiente de Poisson* es $\nu = 0,3$ y su *Densidad* es $\rho = 7800[\frac{Kg}{m^3}]$.
- **Factores de radio f_i:** Cada factor posee un valor $f_i \in [0,5\ ,\ 2]$ modificando así los radios de la IBZ, tal que, el radio del i-esimo elemento de la estructura es $r_i \in [0,4\ ,\ 1,6]$ $[mm]$.
- **Largo y ancho de celda L_c:** Posee un valor de $L_c \in [0,015\ ,\ 0,030]$ $[m]$.
- **Factor de altura f_L:** Con un valor de $f_L \in [0,5\ ,\ 2]$ modifica la distancia entre placas o altura L_z.

3.1.2. Kagome Tridimensional

Esta estructura utiliza la misma geometría mencionada anteriormente con 9 celdas cúbicas base distribuidas de igual forma en el plano XY y Z, por lo tanto, n_c y sus igualdades se mantienen y también $n_z = 1$. La gran diferencia recae en el núcleo que se forma al unir un vértice de la celda base con otro diagonalmente opuesto, esto crea dos pirámides de base cuadrada una invertida sobre la otra dentro de la celda base.

A continuación, en la figura 3.1.4 logra visualizar la celda base en un plano XY (a) y su núcleo (b). Además, cabe destacar que la visualización de las celdas base en el plano XY (c) se representa de igual forma que para la estructura anterior, sin embargo, la IBZ no es igual.

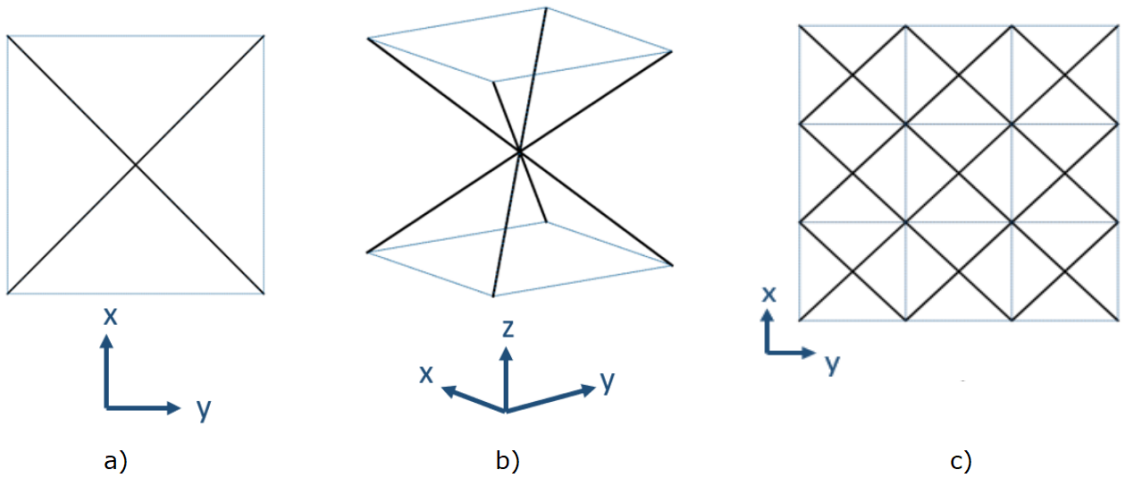


Figura 3.1.4: a) Visualización plano XY celda base. b) Vista isométrica celda base y núcleo Kagome. c) Vista plano XY de celdas base de una estructura tipo Kagome.

La IBZ se genera con 3 simetrías iguales al caso anterior, los elementos en una celda base se representan de la siguiente forma; las flechas comunes con solo una punta en su extremo corresponden a un elemento que une la placa inferior con la superior en el sentido que indica la flecha. Luego las flechas con doble punta son la combinación de dos elementos de flecha común, no obstante, estos poseen igual características y propiedades, por ende, comparten color y se denominan como una flecha doble.

Al igual que la estructura anterior, la de kagome posee 4 elementos por celda base y con 9 celdas base en total se cuentan con 36 elementos. Luego en la IBZ se encuentran 10 elementos, 2 de ellos se encuentran completos (rojo y azul) y 8 de ellos se encuentran a la mitad (2 de cada color: verde, negro, magenta y amarillo), cabe mencionar que una mitad de los elementos verdes o negros se cuentan como 2 debido a su definición. Es así como la IBZ logra ser representada con 6 colores.

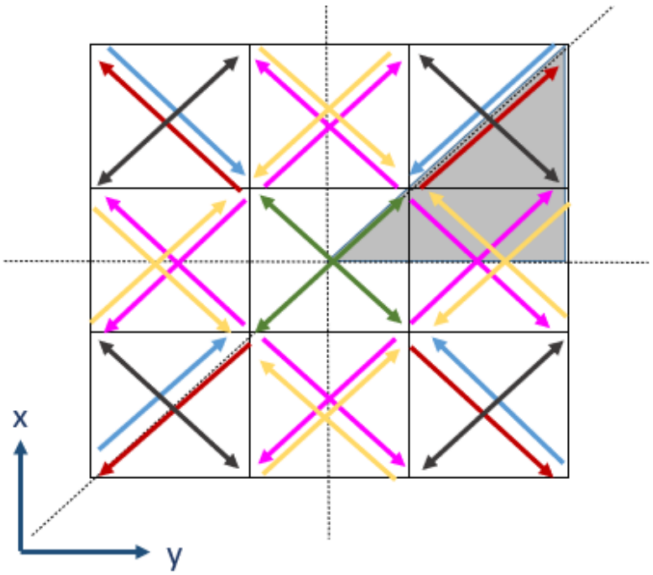


Figura 3.1.5: Vista celdas base en el plano XY de una estructura tipo kagome con líneas de simetría y zona de IBZ marcadas.

Como la estructura es representada por los 10 elementos de la IBZ, que a su vez logran ser representados con 6 colores, por ende, se requiere de 6 factores de radio de la sección transversal. Estos factores se denotan como f_1, f_2, f_3, f_4, f_5 y f_6. Además, se utiliza un factor altura representado por f_L, el cual modifica la distancia o altura entre las placas exteriores del panel, por ende, modifica a su vez la altura de las pirámides. Finalmente, los rangos de valores de dichas variables y factores son:

- **Materialidad:** Los núcleos son de acero, por lo que su *Modulo de Young* es $E = 2,1 \cdot 10^{11}[Pa]$, *Coeficiente de Poisson* es $\nu = 0,3$ y su *Densidad* es $\rho = 7800[\frac{Kg}{m^3}]$.
- **Factores de radio** f_i**:** Cada factor posee un valor $f_i \in [0,01\ ,\ 1]$ modificando así los radios de la IBZ, tal que, el radio del i-esimo elemento de la estructura es $r_i \in [0,004\ ,\ 0,8]\ [mm]$.
- **Largo y ancho de celda** L_c**:** Posee un valor de $L_c \in [0,015\ ,\ 0,030]\ [m]$.
- **Factor de altura** f_L**:** Con un valor de $f_L \in [0,1\ ,\ 1]$ modifica la distancia entre placas o altura L_z.

3.1.3. Pirámide Central

La estructura de pirámide central difiere de las anteriores en su número de celdas unitarias que conforman la base, pues $n_c = n_x = n_y = n_z = 1$. Luego sus largos son $L_c = L_x = L_y$ formando así una celda cuadrada, mientras que los largos son $L_c = L_x = L_y$ y se mantiene $L_z = L_c/3$.

La pirámide central del núcleo (figura 3.1.6 a) posee 8 aristas que se extienden desde la placa inferior hasta la superior y que forman la pirámide misma, es desde estas aristas que se extienden triángulos a lo largo de los ejes de simetría completando el enrejado del núcleo.

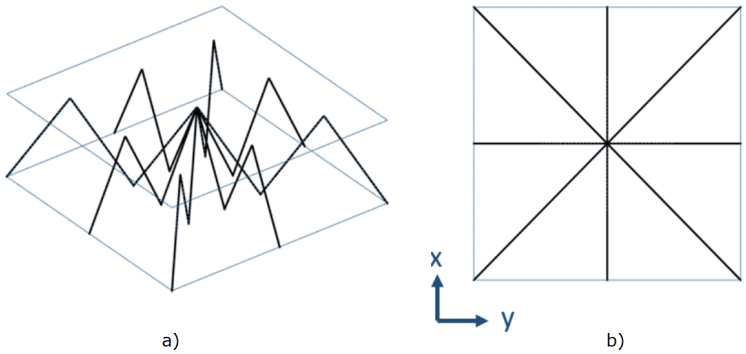


Figura 3.1.6: a) Visualización isométrica de celda base. b) Plano XY de celda base.

En cuanto a la IBZ (resaltada en gris), se produce de igual forma que en las estructuras anteriores con 3 ejes de simetría en la celda base tal como se observa en la figura 3.1.7. La estructura se compone de 48 elementos, mientras que la IBZ se forma a partir de 6 se ellos identificados por un color cada uno. En la figura 3.1.8 se presenta la vista isométrica de la celda base, donde se puede apreciar de mejor forma cada elemento y sus colores.

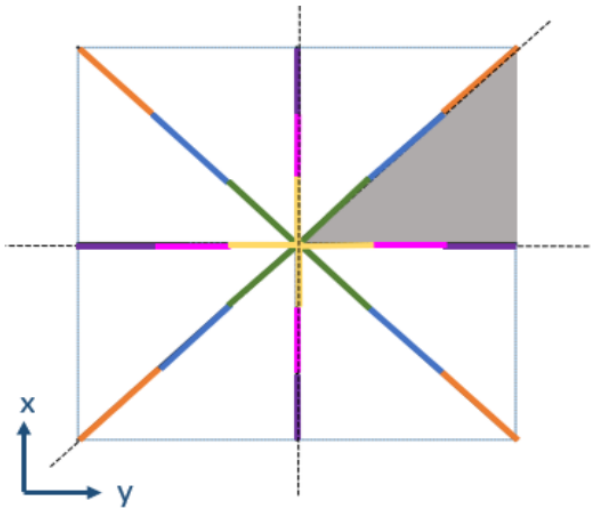


Figura 3.1.7: Vista celda base en el plano XY de una estructura tipo pirámide central con líneas de simetría y zona de IBZ marcadas.

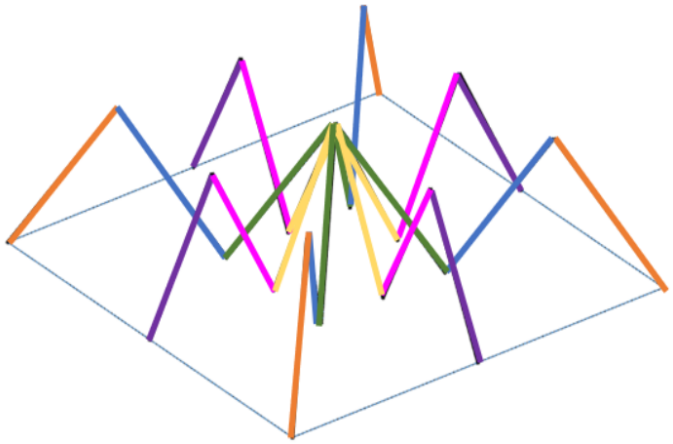


Figura 3.1.8: Vista isométrica de celda base de una estructura tipo pirámide central.

En base a esto la estructura es representada finalmente por los 6 elementos de la IBZ con su factor radio de la sección transversal, estos factores se denotan como f_1, f_2, f_3, f_4, f_5 y f_6. Además, se utiliza un factor altura representado por f_L, el cual modifica la distancia o altura entre las placas exteriores del panel, que modifica a su vez la altura de las pirámides.

- **Materialidad:** Los núcleos son de acero, por lo que su *Modulo de Young* es $E = 2,1 \cdot 10^{11}[Pa]$, *Coeficiente de Poisson* es $\nu = 0,3$ y su *Densidad* es $\rho = 7800[\frac{Kg}{m^3}]$.
- **Factores de radio f_i:** Cada factor posee un valor $f_i \in [0,01\ ,\ 1]$ modificando así los radios de la IBZ, tal que, el radio del i-esimo elemento de la estructura es $r_i \in [0,004\ ,\ 0,8]\ [mm]$.
- **Largo y ancho de celda L_c:** Posee un valor de $L_c \in [0,015\ ,\ 0,030]\ [m]$.
- **Factor de altura f_L:** Con un valor de $f_L \in [0,1\ ,\ 1]$ modifica la distancia entre placas o altura L_z.

3.2. Cálculo de características del Band Gap

Los diagramas de bandas entregados se caracterizan por tener 8 bandas a lo largo de un vector de ondas ($\mathbf{k}$) de 28 datos, es decir, cada banda se define a partir de 28 puntos y las frecuencias naturales son denotadas por $w_i(k)[Hz]$, donde i representa el número de la banda. Luego el band gap se produce cuando 2 bandas tienen una diferencia de valor positivo, esta diferencia se llama ancho de band gap o ancho (Δw) y para efectos de la predicción en la regresión se utilizan también los valores de ancho negativos (sin band gap).

La intuición dicta que el band gap se produce entre bandas adyacentes y en general ese es el caso, sin embargo, una banda superior no es estrictamente mayor a la banda inferior. Por lo tanto, cuando se calcula el ancho se utiliza la diferencia entre el valor mínimo de todas las bandas superiores y el valor máximo de todas las bandas inferiores, tal y como se muestra en la siguiente ecuación:

$$\Delta w = min\{w_{i+1}, \ldots, w_8\} - max\{w_1, \ldots, w_i\} \tag{3.1}$$

Luego de encontrar el ancho es posible calcular la frecuencia media en donde se ubica el band gap, la cual es simplemente un promedio de la frecuencia mínima y máxima utilizada:

$$w_t = \frac{min\{w_{i+1}, \ldots, w_8\} + max\{w_1, \ldots, w_i\}}{2} \tag{3.2}$$

Capítulo 4

Metodología

1. **Revisión de literatura:** Este trabajo de título requiere conocer los paneles sándwich y sus características, pues de esta forma se comprende el band gap estudiado. Además, se necesitan implementar los algoritmos SVR y GPR.

2. **Creación de base de datos:** Se deben crear topologías con distintas características para armar una base de datos lo suficientemente grande para entrenar los algoritmos. A partir de un modelo de elementos finitos (FEM) proporcionado, cuyas configuraciones son 3: pirámide cuadrada, pirámide central y kagome tridimensional.

3. **División de base de datos:** Previo al entrenamiento se debe preparar la base de datos. Para ello esta es dividida en 2 datasets; entrenamiento y pruebas. El primero es usado para entrenar los modelos y sus parámetros, además, es utilizado para evaluar las distintas combinaciones de estos. Finalmente se realiza una evaluación del desempeño de los modelos utilizando el dataset de pruebas.

4. **Selección de hiperparámetros:** En este paso se diseñan los modelos de SVR y GPR. Para ello se utiliza una estrategia *grid search* de los hiperparámetros de los algoritmos en búsqueda del mejor resultado.

5. **Análisis de resultados:** Luego de los procedimientos anteriores, se comparan y analizan las predicciones obtenidas de los modelos diseñados con los resultados de los modelos FEM.

4.1. Base de datos

La base de datos proporcionada consta de 3 estructuras de paneles tipo sándwich, estas son; Pirámide cuadrada, Kagome tridimensional y de Pirámide central. Cada una de estas estructuras posee un dataset de 20.000 casos y cada uno de estos casos están definidos por 7 parámetros.

Luego cada caso posee un diagrama de bandas definido por 8 bandas en un vector posición de 28 datos. Por lo tanto, cada estructura posee 8 datasets de 20.000 casos con 28 columnas cada uno. A continuación, se presenta un diagrama de la base de datos:

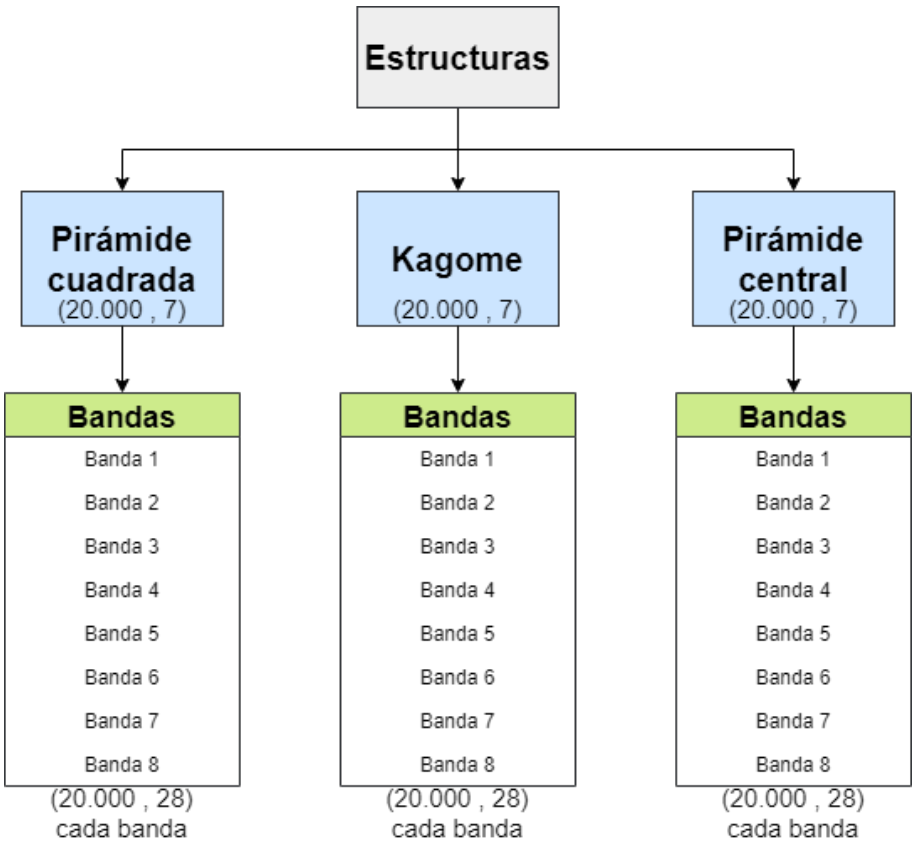


Figura 4.1.1: Diagrama de la base de datos.

Para cada estructura se cuenta con que los casos pueden tener o no presencia band gap. En cada una la cantidad de casos con band gap son minoría, en la siguiente tabla se presenta la distribución de estos:

Tabla 4.1: Distribución de casos con y sin band gap.

Estructura	Casos con Band Gap	Casos sin Band Gap
Pirámide Cuadrada	390	19.610
Kagome	342	19.658
Pirámide Central	401	19.599

4.2. Enfoques

Para este trabajo se consideran tres enfoques para la predicción del band gap. El primer y segundo enfoque consiste en predecir el diagrama de bandas completo, es decir, para cada caso se obtiene una predicción de cada banda. El tercer enfoque predice directamente el ancho del band gap y su frecuencia media.

4.2.1. Enfoque I: Un modelo para cada estructura

Este enfoque busca predecir el diagrama de bandas completo desarrollando un modelo para cada estructura a partir de los parámetros de diseño de esta. Los parámetros de entrada corresponden a los parámetros de diseño para cada caso de una de las estructuras mencionadas, esto es, una matriz de $(20.000, 7)$. La salida corresponde al diagrama de bandas completo, en forma de una matriz de $(20.000, 224)$.

Finalmente, al ser la predicción del diagrama completo, se debe calcular el band gap para cada caso. Esto es, tanto su ancho de banda como la frecuencia media en que se encuentra.

4.2.2. Enfoque II: Un modelo por cada banda

Al igual que el enfoque anterior este busca predecir el diagrama de bandas completo, no obstante, en vez de desarrollar solo un modelo de aprendizaje de máquinas, se crea un modelo por cada banda. Por lo tanto, se poseen 8 modelos para cada estructura. Los parámetros de entrada se mantienen para cada modelo, siendo estos los parámetros de diseño para cada estructura con una matriz de tamaño $(20.000, 7)$. Luego la salida para cada modelo corresponde a la banda que este prediga siendo esta de tamaño $(20.000, 28)$.

El procedimiento final procede de la misma forma que el enfoque anterior, requiriendo este el cálculo de las características del band gap.

4.2.3. Enfoque III: Predicción directa de band gap

En este enfoque se busca predecir directamente las características del band gap, es decir, su ancho de banda y la frecuencia media de este. Por lo que se requiere del cálculo previo de estos valores para generar la nueva base de datos. Los parámetros de entrada son los mismo que en los enfoques anteriores una matriz de tamaño $(20.000, 7)$ para cada estructura. En cambio, la salida de este se enfoca en dos valores, lo cual corresponde a una matriz de $(20.000, 2)$.

Finalmente, como los datos de salida son directamente el ancho y la frecuencia media del band gap no se requiere de un procesamiento posterior.

4.3. Preproceso de datos

Para llevar a cabo los enfoques anteriores se requiere de un procesar previamente las bases de datos, en específico las bandas para los enfoques I y II, además del ancho y frecuencia media del band gap para el enfoque III.

Esto se logra al estandarizar los datos a una distribución normal $N(\mu = 0, \sigma^2 = 1)$, donde μ es el promedio y σ^2 es la desviación estándar. De esta forma los datos en diferente escala afectan de igual forma al entrenamiento del modelo. Esto se aplica a los diagramas de banda para los primeros dos enfoques, mientras que para el tercero se aplica para las características del band gap.

En el enfoque I se deben concatenar todas las bandas en un solo dataset, es decir, se crea un archivo de 20.000 filas que representan los casos y de 224 columnas, las cuales son las bandas juntas. Las primeras 28 columnas corresponden a la Banda 1, luego las siguientes 28 corresponden a la banda 2 y así sucesivamente. Luego para los demás enfoques este proceso no es necesario.

4.4. División de base de datos

Previo al entrenamiento del modelo se deben dividir los datasets, tanto el de entrada como el de salida. Se crean 2 conjuntos; entrenamiento y pruebas. Estos consisten en un dataset **X** correspondiente a la entrada del modelo y un dataset **Y** correspondiente a la salida. Finalmente se cuenta con los archivos X_{train} y Y_{train} para el entrenamiento, los cuales corresponden al 80 % de los datos. Mientras que X_{test} y Y_{test} son para realizar pruebas y estos corresponden al 20 % de los datos.

4.5. Selección de hiperparámetros

Cada modelo se define por sus hiperparámetros y su selección determina el desempeño de estos. Para los modelos de SVR se seleccionan entre los parámetros C, gamma, épsilon y kernel. Luego para los modelos de GPR el único parámetro a seleccionar es el kernel. Finalmente se realiza una búsqueda gridsearch sobre estos parámetros para determinar los valores que logren un mejor desempeño.

Esta búsqueda se realiza una vez por parámetro, mientras los otros se dejan fijos durante esa búsqueda. En otras palabras, se hace variar un parámetro a la vez para determinar su valor óptimo.

4.5.1. SVR

A continuación, se presentan los valores estudiados para cada parámetro de la búsqueda realizada:

- **C:** [10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}]
- **Gamma:** [10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, *default = scale*]
- **Épsilon:** [10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}]
- **Kernel:** [*rbf*, *poly*, *sigmoid*]

4.5.2. GPR

A continuación, se presentan los kernels estudiados para la búsqueda realizada:

- **RBF**

- **RBF + WhiteKernel**
- **ExpSineSquared**
- **DotProduct**

4.6. Positividad del dataset

Dada la distribución de datos (tabla 4.1) en los datasets donde aproximadamente se posee entre un $1,7$ y $2\,\%$ de datos con ancho de band gap positivo (con bandgap), se decide realizar un análisis extra al comportamiento de los modelos del tercer enfoque frente a un cambio de positividad en los dataset.

Para ello se crean otros 5 datasets donde el porcentaje de positividad es mayor al original, no obstante, estos poseen una menor cantidad de datos en total. Estos datasets constan de $10\,\%$, $20\,\%$, $50\,\%$, $80\,\%$ y $100\,\%$ de casos con band gap. Cabe destacar que este análisis es realizado solo sobre el dataset de Kagome, ya que como se muestra en la sección de resultados es la estructura donde se obtuvieron predicciones con mejores resultados.

La creación de ellos se hace de la siguiente forma: primero se identifican todos los casos con band gap y luego se le agregan de forma aleatoria casos sin band gap hasta completar el $\%$ de positividad deseado. Esto implica que para todos los nuevos datasets se tienen la misma cantidad y los mismos casos con band gap, solo con la cantidad de negativos variante y aleatorios. De esta forma se obtienen los siguientes datasets agregando el original de 20.000 casos:

Tabla 4.2: Distribución de casos con (Positivo) y sin (Negativo) band gap para los distintos datasets.

% de Positivos	**Kagome**		
	Positivos	**Negativos**	**Total**
1,71	342	19.658	20.000
10		3.078	3.420
20		1.368	1.710
50		342	684
80		86	428
100		0	342

Luego de obtener los resultados en los datasets anteriormente descritos, estos son R^2 de ancho y frecuencia media, se realiza un análisis solo a los casos positivos en cada uno. Para ello se identifican todos los casos con band gap en el dataset de pruebas y se les calcula su R^2. Esto logra aportar información sobre el desempeño del modelo y su predicción con mayor profundidad en respecto a solo los casos que poseen band gap y, por ende, como afecta la cantidad de datos sin band gap en el dataset.

Capítulo 5

Resultados

En este capítulo se presentan los resultados de los modelos entrenados según los enfoques mencionados, en específico se trabaja con la métrica del R^2 score sobre el dataset de pruebas que se representa mediante un gráfico de dispersión o *scatter plot*. Estos presentan en el eje de ordenadas los valores de predicción, mientras que en el eje de las abscisas se presentan los valores reales que se desean predecir. Este gráfico permite tener una representación de la predicción, además de una línea de tendencia roja donde $x = y$ la cual representa $R^2 = 1$, es decir, una predicción perfecta.

A continuación, se presentan dos tablas resúmenes para cada modelo, cabe notar que SVR ha sido utilizado para los 3 enfoques mientras que GPR solo para el tercero. Esto se debe a múltiples razones que son explicadas en los siguientes capítulos.

Tabla 5.1: Resumen de resultados de SVR sobre el dataset de pruebas para los distintos enfoques, en rojo predicciones con bajo desempeño.

SVR	**Enfoque I**			**Enfoque II**		**Enfoque III**	
R2	Bandas	Ancho	F. Media	Ancho	F. Media	Ancho	F. Media
Pirámide Cuadrada	0.98	-0.07	-0.89	-0.21	-1.6	0.94	0.64
Kagome	-	-	-	-0.78	-0.63	0.95	0.93
Pirámide Central	-	-	-	-0.013	0.43	0.84	0.87

Tabla 5.2: Resumen de resultados de GPR sobre el dataset de pruebas para el tercer enfoque.

GPR	**Enfoque III**	
R2	Ancho	F. Media
Pirámide Cuadrada	0.65	0.94
Kagome	0.96	0.93
Pirámide Central	0.86	0.88

5.1. Enfoque I

A continuación se muestran los resultados obtenidos con el primer enfoque. En la figura 5.1.1 se muestra un gráfico de dispersión de la predicción del diagrama de bandas realizada con SVR.

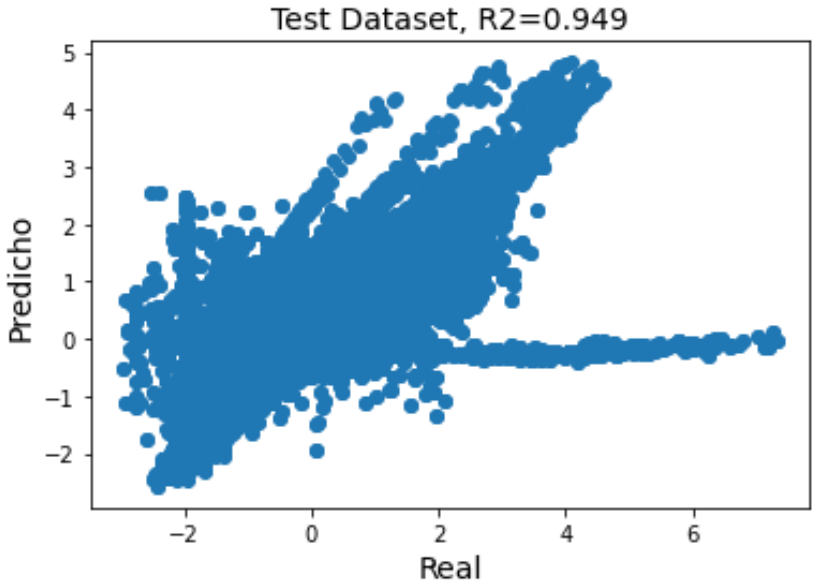


Figura 5.1.1: Predicción de SVR sobre el dataset de pruebas de la estructura de pirámide cuadrada con $R^2 = 0.949$.

En la figura 5.1.2 se muestra la predicción anterior separada por banda en sus respectivos gráficos de dispersión.

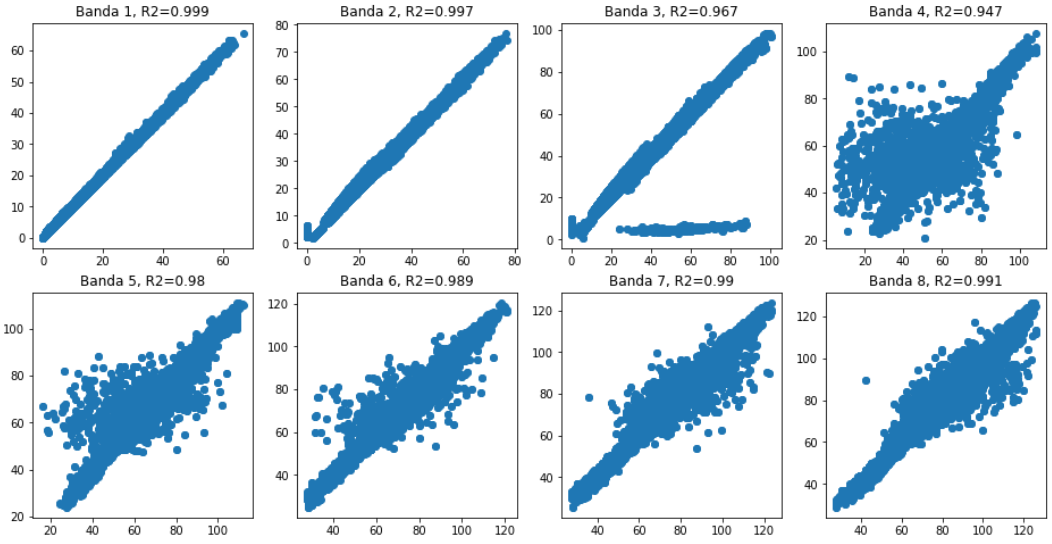
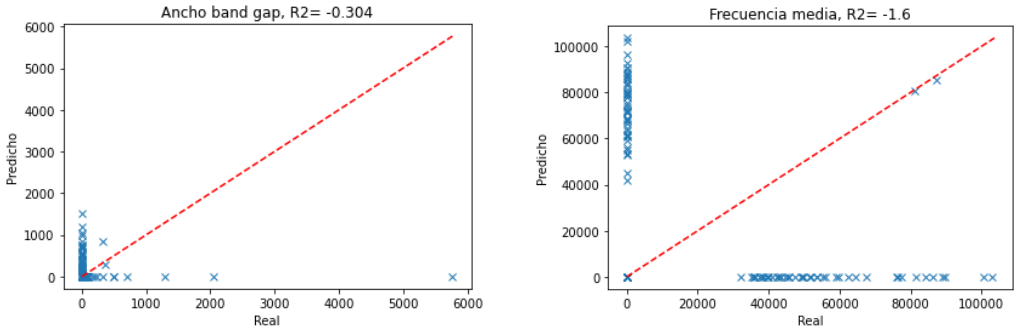


Figura 5.1.2: Predicción de SVR desglose por cada banda de la estructura de pirámide cuadrada.

La figura 5.3 muestra los gráficos obtenidos al calcular el ancho y frecuencia media del band gap a partir de la predicción del diagrama de bandas. En rojo se muestra la linea de tendencia de predicción perfecta.



(a) Ancho (b) Frecuencia media

Figura 5.3: SVR: Score R^2 del ancho y frecuencia media de la estructura de pirámide cuadrada.

5.2. Enfoque II

5.2.1. Estructura pirámide cuadrada

En la siguiente figura 5.2.1 se muestra la predicción obtenida sobre cada una de las bandas de la estructura de pirámide cuadrada. Notar que para cada banda se entrena un modelo de SVR.

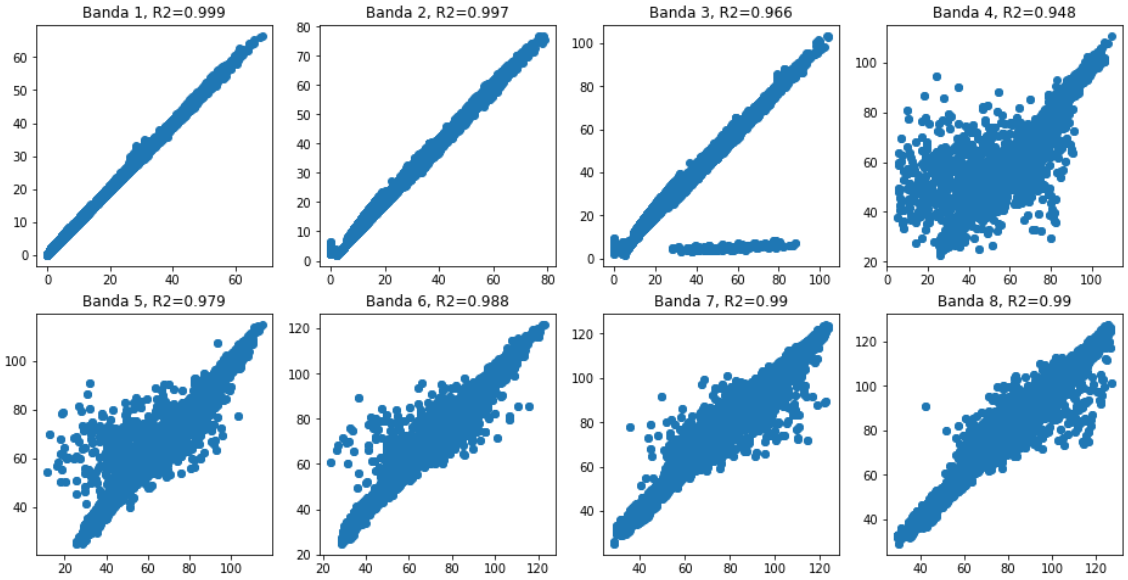
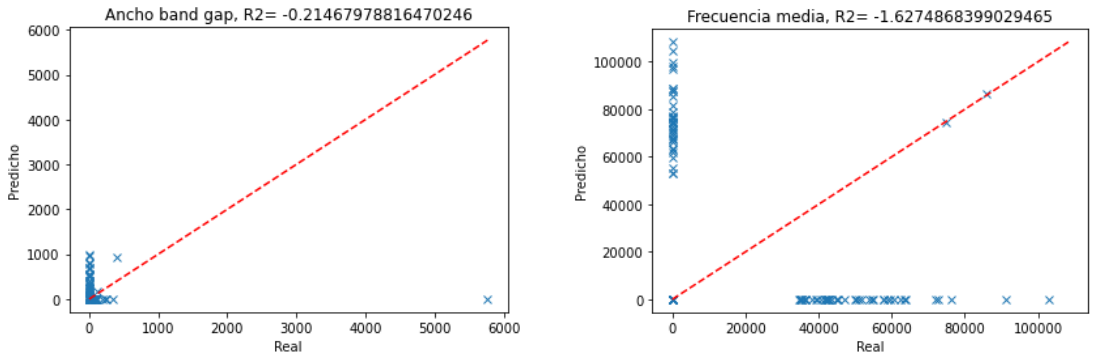


Figura 5.2.1: Predicción de SVR para cada banda de la estructura de pirámide cuadrada.

La figura 5.2.b se muestra el ancho y frecuencia media calculados a partir de las bandas predichas.



(a) Ancho (b) Frecuencia media

Figura 5.2: SVR: Score R^2 del ancho y frecuencia media de la estructura de pirámide cuadrada.

5.2.2. Estructura kagome

La figura 5.2.3 muestra las predicciones obtenidas para cada banda de la estructura de kagome, para las cuales se utiliza un modelo para cada banda.

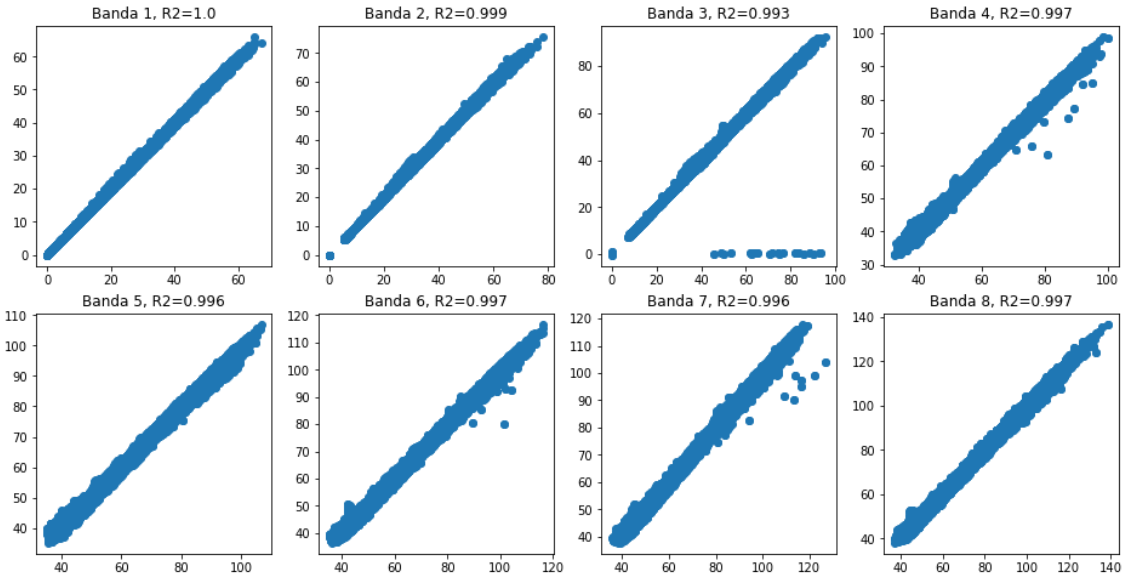
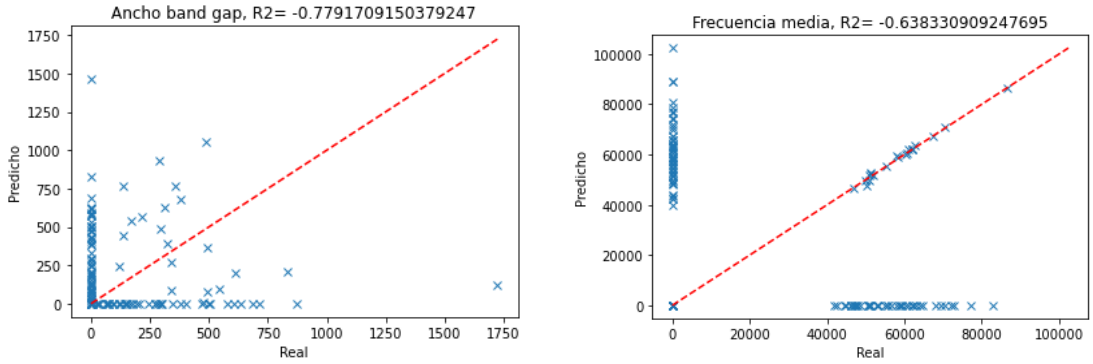


Figura 5.2.3: Predicción de SVR para cada banda de la estructura de kagome.

Luego se calcula el ancho y frecuencia media del band gap obtenido a partir de la predicción anterior de las bandas.



(a) Ancho (b) Frecuencia media

Figura 5.4: SVR: Score R^2 del ancho y frecuencia media de la estructura de kagome.

5.2.3. Estructura pirámide central

A continuación se muestran los resultados obtenidos para la estructura de pirámide central. En la figura 5.2.5 se observan los diagramas de dispersión para las predicciones para cada

banda.

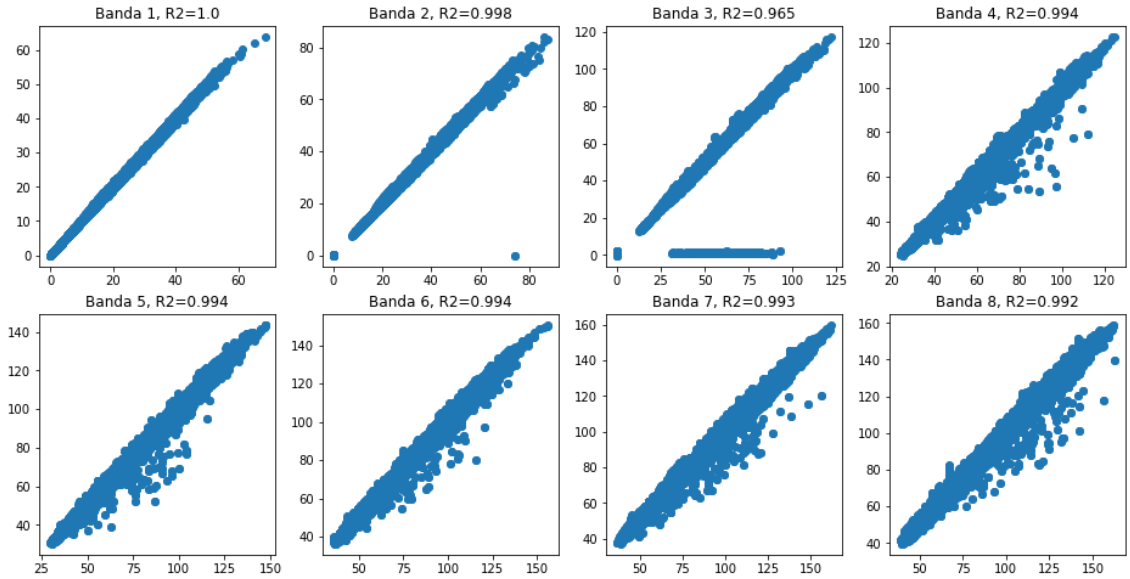
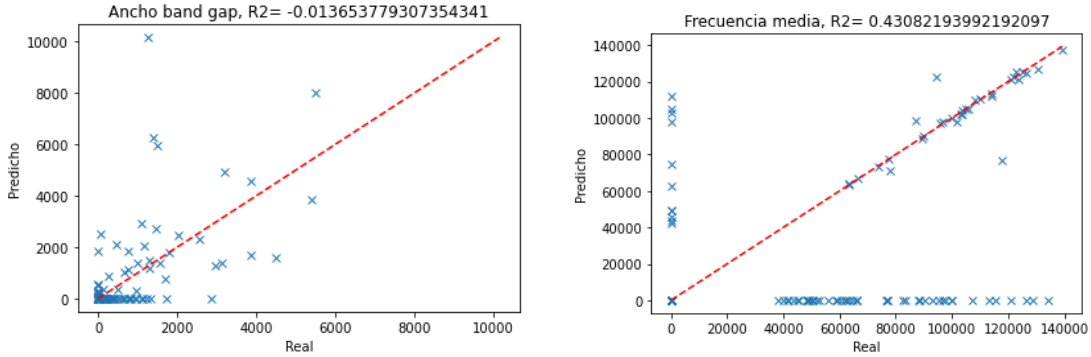


Figura 5.2.5: Predicción de SVR para cada banda de la estructura de pirámide central.

Luego en la figura 5.6 se muestra el ancho y frecuencia media calculados a partir de las predicciones anteriores.



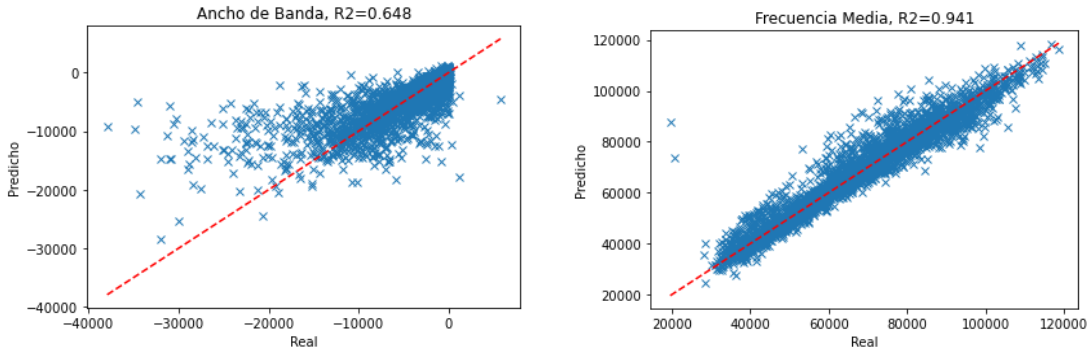
(a) Ancho (b) Frecuencia media

Figura 5.6: SVR: Score R^2 del ancho y frecuencia media de la estructura de pirámide central.

5.3. Enfoque III

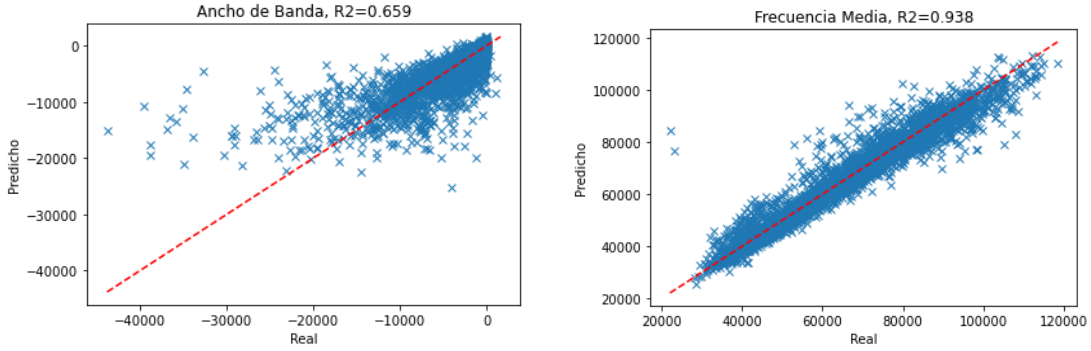
5.3.1. Estructura pirámide cuadrada

A continuación se muestran los resultados de la predicción directa del ancho y frecuencia media para la estructura de pirámide cuadrada. En la figura 5.1 se observan los gráficos de dispersión para el ancho y frecuencia media utilizando el modelo de SVR, mientras que en la figura 5.2 se muestran los resultados obtenidos con el modelo de GPR.



(a) Ancho (b) Frecuencia media

Figura 5.1: SVR: Score R^2 del ancho y frecuencia media de la estructura de pirámide cuadrada.

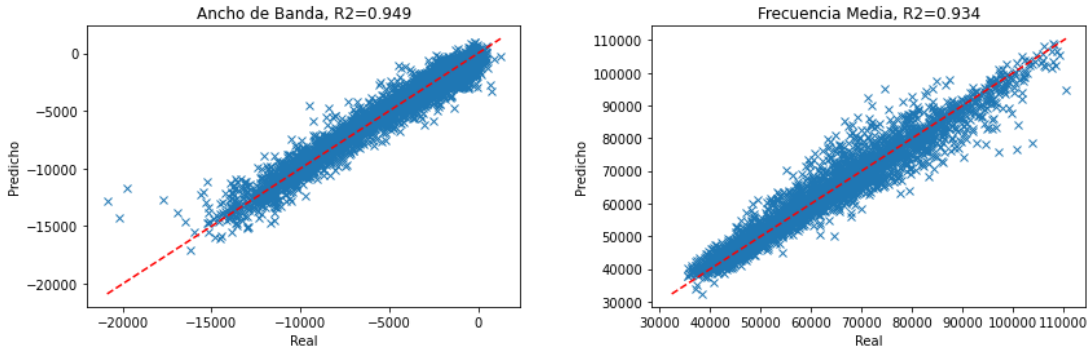


(a) Ancho (b) Frecuencia media

Figura 5.2: GPR: Score R^2 del ancho y frecuencia media de la estructura de pirámide cuadrada.

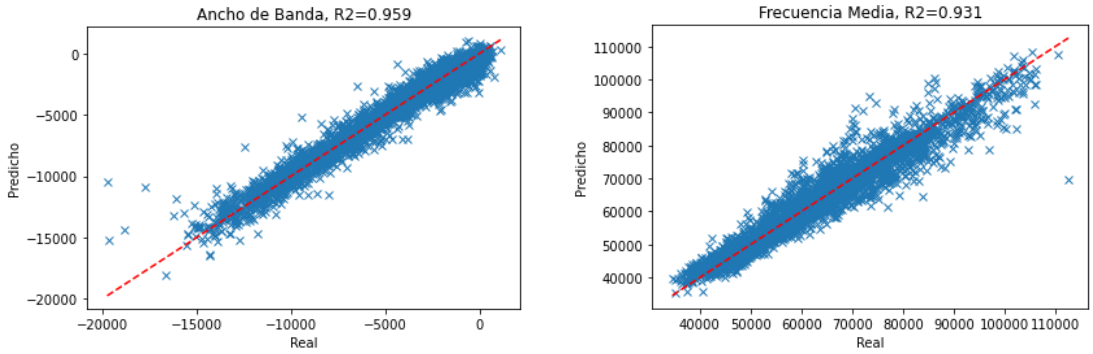
5.3.2. Estructura kagome

A continuación se muestran los resultados de la predicción directa del ancho y frecuencia media para la estructura de Kagome. En la figura 5.3 se observan los gráficos de dispersión para el ancho y frecuencia media utilizando el modelo de SVR, mientras que en la figura 5.4 se muestran los resultados obtenidos con el modelo de GPR.



(a) Ancho (b) Frecuencia media

Figura 5.3: SVR: Score R^2 del ancho y frecuencia media de la estructura de kagome.

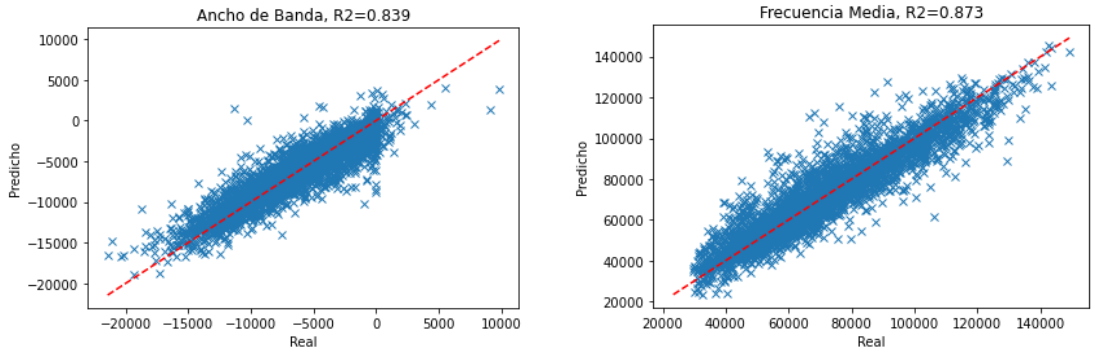


(a) Ancho (b) Frecuencia media

Figura 5.4: GPR: Score R^2 del ancho y frecuencia media de la estructura de kagome.

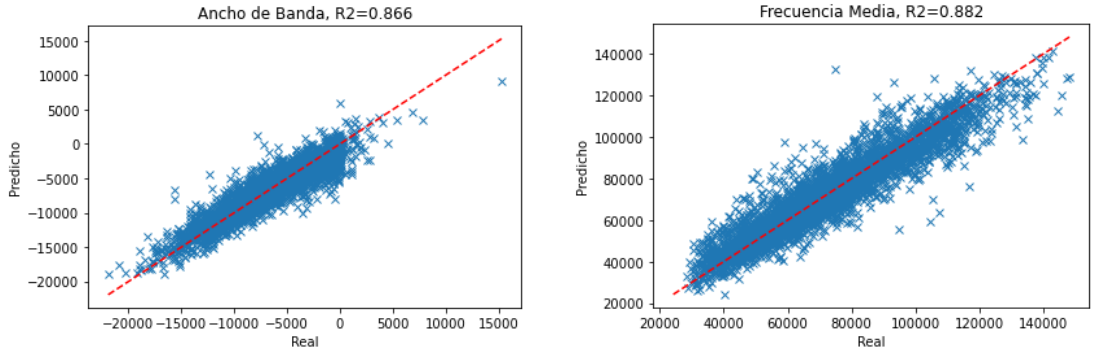
5.3.3. Estructura pirámide central

A continuación se muestran los resultados de la predicción directa del ancho y frecuencia media para la estructura de pirámide central. En la figura 5.5 se observan los gráficos de dispersión para el ancho y frecuencia media utilizando el modelo de SVR, mientras que en la figura 5.6 se muestran los resultados obtenidos con el modelo de GPR.



(a) Ancho (b) Frecuencia media

Figura 5.5: SVR: Score R^2 del ancho y frecuencia media de la estructura de pirámide central.



(a) Ancho (b) Frecuencia media

Figura 5.6: GPR: Score R^2 del ancho y frecuencia media de la estructura de pirámide central.

5.4. Selección de hiperparámetros

Luego de los gridsearch realizados se obtiene la siguiente selección de hiperparámetros para SVR:

- **Kernel:** RBF
- **C:** 100
- **Épsilon:** 0.1
- **Gamma:** 1

Mientras que para el GPR se realizan 2 gridsearchs; una para el modelo que predice el ancho de banda y otra para el modelo que predice la frecuencia media. En ambos se obtiene que el mejor kernel de los estudiados es:

- **Kernel:** RBF + WhiteKernel

5.5. Positividad en datasets

A continuación se muestran los resultados obtenidos del análisis de positividad de los datasets mencionados en la tabla 4.2 del capítulo anterior. En la tabla 5.3 se observa el coeficiente R^2 obtenido por los modelos de SVR y GPR para cada uno de los datasets.

Tabla 5.3: Resultados de análisis de positividad en datasets de estructura de Kagome, R^2 score de SVR y GPR, en rojo predicciones con bajo desempeño.

R2	**Kagome**			
	SVR		**GPR**	
% de Positivos	Ancho	F. Media	Ancho	F. Media
1,71	0.95	0.93	0.96	0.93
10	0.91	0.9	0.92	-1.56
20	0.88	0.86	0.91	-1.53
50	0.83	0.84	0.84	-1.95
80	0.66	0.94	0.74	-1.15
100	-0.25	0.94	0.06	0.12

Luego para profundizar el análisis se cálcula el coeficiente R^2 solo para los casos con band gap de las predicciones obtenidas originalmente. En otras palabras, el rendimiento de los modelos de SVR y GPR sobre los casos positivos.

Tabla 5.4: R^2 score calculado con solo los casos con band gap de los datasets utilizados en el análisis de positividad, en rojo predicciones con bajo desempeño.

R2 calculado datos con band gap	**Kagome**			
	SVR		**GPR**	
% de Positivos	Ancho	F. Media	Ancho	F. Media
1,71	0,96	-20	0,97	-14
10	0,95	-11	0,96	-9
20	0,93	-26	0,97	-5
50	0,93	-4	0,96	-11
80	0,95	-1,9	0,94	-13
100	0,94	-0,25	0,96	0,06

Capítulo 6

Discusión

6.1. Sobre el enfoque I

Los resultados de este enfoque en primera instancia parecen tener buen desempeño al utilizar la métrica de R^2 sobre la predicción de las bandas en la figura 5.1.1, no obstante, al graficar todas las bandas en la figura 5.1.2 se observa que solo logra predecir satisfactoriamente las primeras 3 bandas. Luego al calcular el ancho de band gap y la frecuencia media (figura 5.3) es evidente que el modelo no logra predecir satisfactoriamente las bandas con tal de obtener buenos resultados al calcular estas características.

El principal problema de este enfoque es que el dataset de salida u output posee demasiados datos (representado en la figura 5.1.1), ya que es un conjunto de 224 columnas que representan las 8 bandas, por lo tanto, una regresión no es capaz de ajustarse y realizar una predicción que satisfaga todas las bandas al mismo tiempo.

Otro problema que surge respecto al tamaño del dataset es que el entrenamiento es bastante costoso computacionalmente, aún más la implementación de la estrategia gridsearch para optimizar los hiperparámetros. Finalmente se decide utilizar solo SVR para este enfoque, ya que no es un problema del modelo, sino que del dataset.

6.2. Sobre el enfoque II

Es a partir de esta línea de pensamiento que nace el segundo enfoque, ya que, al entrenar un modelo para cada banda, no solo son menos datos de output, sino que también baje el tiempo de computación. El desempeño de los modelos a la hora de predecir las bandas es satisfactorio a través de las tres estructuras, en especial para la de núcleo de Kagome (figura 5.2.3), mientras que las de pirámide cuadrada y central (figuras 5.2.1 y 5.2.5) a pesar de poseer buenos R^2 en ciertas bandas, los casos no logran agruparse en la línea de tendencia de predicción perfecta.

Luego al calcular las características del band gap y observar las figuras 5.2, 5.4 y 5.6, es notorio que estos modelos no logran predecir satisfactoriamente las bandas dado su bajo R^2 tanto en el ancho como en la frecuencia media transversalmente para las tres estructuras. Debido a que los resultados de la regresión de las bandas son, en cuanto a R^2 bastante altos, resulta complejo optimizar los hiperparámetros con tal de obtener un $R^2 = 1$ en todas las bandas y en consecuencia aumentar el desempeño de la predicción de las características del band gap. No obstante, el costo computacional es bastante menor comparado con el enfoque I, por lo que, efectivamente una reducción del dataset de output es efectiva para su disminución.

6.3. Sobre el enfoque III

A partir de los enfoques anteriores se determina otra línea de trabajo con tal de mejorar la obtención de las características del band gap, para ello no solo se predice directamente el ancho y frecuencia media, sino que también se utilizan tanto los casos con band gap (ancho positivo) como los casos sin band gap (ancho negativo). Nuevamente el tamaño del dataset de output se ve reducido, por lo que el tiempo de entrenamiento y por ende costo computacional disminuye. Esto permite realizar un gridsearch extenso sobre cada hiperparámetro y optimizarlos.

Se observa para las tres estructuras una mejora en la predicción de tanto el ancho como la frecuencia media con respecto a los enfoques anteriores. En la figura 5.1 y 5.2 de la estructura de pirámide cuadrada tanto SVR como GPR obtienen un R^2 por sobre de 0,9, pero menor a 1 para la frecuencia media, mientras que la regresión del ancho no logra sobrepasar un R^2 de 0,7.

Luego para la estructura de kagome se observa en las figuras 5.3 y 5.4 que tanto SVR como GPR reproducen de forma satisfactoria las características del band gap.

En cuanto a los resultados para la estructura de pirámide central, se observa en las figuras 5.5 y 5.6 que tanto para las regresiones de ancho como de frecuencia se logra predecir satisfactoriamente sus valores con un R^2 superior a 0,8, pero menor a 0,9.

6.4. Sobre la selección de hiperparámetros

Tal como se mencionó anteriormente la selección de hiperparámetros a través del método gridsearch es eficiente cuando se poseen datasets con menor cantidad de datos de salida para los modelos y, por ende, para el tercer enfoque es donde se utilizó exhaustivamente logrando así una optimización satisfactoria de los hiperparámetros.

No obstante, so ha sido posible aumentar el desempeño de los modelos para las estructuras de pirámide cuadrada y central con tal de superar la barrera del $R^2 = 0.9$.

6.5. Comparación de enfoques

Al observar las tablas 5.1 y 5.2 se puede descartar el enfoque I y II, pues no logran predecir ni el ancho de banda ni su frecuencia media. Además, debido al coste computacional de estos, aumenta la dificultad de optimización y entrenamiento por lo que no se recomiendan. Luego para el enfoque III ambos modelos logran altos R^2 para las 3 estructuras, especialmente para la estructura de kagome.

6.6. Sobre la positividad en datasets

Debido a que se logran mejores resultados en la estructura de kagome tanto con SVR como con GPR, se decide realizar un análisis del dataset utilizado en búsqueda de mejorar el desempeño de los modelos. En la tabla 5.3 se observa que para datasets de tamaño inferior al original (20.000 casos, 1,71 % positividad) el modelo de GPR no logra predecir satisfactoriamente la frecuencia media, en cambio para el ancho este si logra predecirlo con una menor cantidad de datos y alta positividad (hasta el 80 %).

En la misma tabla mencionada anteriormente se observa que el modelo SVR logra predecir satisfactoriamente tanto el ancho como la frecuencia media hasta un 80 % de positividad. No obstante, con el dataset de 100 % el R^2 del ancho es negativo, esto se debe a que el tamaño del dataset es bastante pequeño con solo 342 datos.

Luego al observar la tabla 5.4 que muestra el R^2 calculado solo con los casos con band gap de los datasets anteriores, es decir, el desempeño de los modelos sobre los 342 datos con band gap que se encuentran presentes constantemente en los 6 sets de datos utilizados. Se observa que tanto SVR como GPR logran predecir el ancho de band gap para estos casos en los 6 datasets, mientras que no logra predecir la frecuencia media en ninguno de ellos. Sin embargo, se observa la tendencia que a mayor positividad mejor es la predicción de la frecuencia media para los casos con band gap.

Capítulo 7

Conclusión

En el presente trabajo se ha logrado desarrollar modelos de aprendizaje de máquinas capaces de predecir el band gap y sus características en paneles tipo sándwich, en específico los algoritmos de support vector regression y gaussian process regression entrenados para predecir directamente el ancho del band gap y su frecuencia media en estructuras con núcleo de tipo pirámide cuadrada, kagome tridimensional y pirámide central.

A partir de los resultados de los enfoques I y II se hace evidente que no es posible calcular satisfactoriamente las características del band gap a partir de una predicción directa de las bandas. Esto se debe a que el band gap es muy sensible a los valores de las bandas, por lo tanto, un error en estas no permite identificarlo y, por ende, por la forma en cómo se calcula, tampoco es posible obtener la frecuencia media si no se obtiene el ancho en estos enfoques.

Finalmente, el tercer enfoque propuesto logra predecir directamente de forma satisfactoria las características del band gap para las tres estructuras estudiadas, en especial para la de kagome. Tanto GPR como SVR otorgan resultados similares, sin embargo, se recomienda el uso de SVR incluso con la mayor cantidad de hiperparámetros que se deben optimizar en SVR. Pues, este algoritmo se logra entrenar en un menor tiempo sin tener efectos negativos en los resultados.

Los resultados no permiten determinar tajantemente el comportamiento de los modelos al ser entrenados con datasets de mayor tamaño y alto porcentaje de positividad, sin embargo, se observa la tendencia de que al aumentar ambos se pueda mejorar la predicción del ancho y frecuencia para casos con y sin band gap.

Para trabajos futuros se recomienda investigar el desempeño de los modelos frente a un dataset con un número de casos entre 5.000 y 20.000 que a su vez poseen una mayor cantidad de casos con band gap entre un 80 % a 100 %.

Bibliografía

[1] T. A. Schaedler and W. B. Carter, "Architected cellular materials," *Annual Review of Materials Research*, vol. 46, no. 1, pp. 187–210, 2016.

[2] J. R. Vinson, "Sandwich structures: Past, present, and future," in *Sandwich Structures 7: Advancing with Sandwich Structures and Materials* (O. Thomsen, E. Bozhevolnaya, and A. Lyckegaard, eds.), (Dordrecht), pp. 3–12, Springer Netherlands, 2005.

[3] V. D'Alessandro, G. Petrone, F. Franco, and S. De Rosa, "A review of the vibroacoustics of sandwich panels: Models and experiments," *Journal of Sandwich Structures and Materials*, vol. 15, pp. 541–582, 09 2013.

[4] A. Marshall, *Sandwich Construction*, pp. 557–601. Boston, MA: Springer US, 1982.

[5] P. Deymier, *Acoustic Metamaterials and Phononic Crystals*. 01 2013.

[6] G. Yi and B. D. Youn, "A comprehensive survey on topology optimization of phononic crystals," *Structural and Multidisciplinary Optimization*, vol. 54, 11 2016.

[7] I. Psarobas, N. Stefanou, and A. Modinos, "Scattering of elastic waves by periodic arrays of spherical bodies," *Physical Review B*, vol. 62, p. 278, 07 2000.

[8] M. Sigalas, M. Kushwaha, E. Economou, M. Kafesaki, I. Psarobas, and W. Steurer, "Classical vibrational modes in phononic lattices: Theory and experiment," *Zeitschrift Fur Kristallographie*, vol. 220, pp. 765–809, 05 2005.

[9] H.-W. Dong, X.-X. Su, Y.-S. Wang, and C. Zhang, "Topological optimization of two-dimensional phononic crystals based on the finite element method and genetic algorithm," *Structural and Multidisciplinary Optimization*, vol. 50, 10 2014.

[10] M. T. Ribeiro, S. Singh, and C. Guestrin, "Model-agnostic interpretability of machine learning," 2016.

[11] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," 2004.

[12] Scikit Learn, "Support vector machines." https://scikit-learn.org/stable/modules/svm.html#svm-regression, 2022. [Online; accessed 17-January-2022].

[13] M. Kuss and C. E. Rasmussen, "Assessing approximate inference for binary gaussian process classification," *Journal of Machine Learning Research*, vol. 6, no. 57, pp. 1679–1704, 2005.

[14] Scikit Learn, "Gaussian processes." https://scikit-learn.org/stable/modules/gaussian_process.html#gaussian-process, 2022. [Online; accessed 17-January-2022].

[15] Scikit Learn, "R2 score, the coefficient of determination." https://scikit-learn.org/stable/modules/model_evaluation.html#r2-score, 2022. [Online; accessed 17-January-2022].

[16] Scikit Learn, "Exhaustive grid search." https://scikit-learn.org/stable/modules/grid_search.html#exhaustive-grid-search, 2022. [Online; accessed 17-January-2022].

Anexos

Anexo A

Resultados gridsearch

A.1. SVR

Código A.1: Resultados gridsearch cv realizado para SVR en su parámetro C.

```
best_score: 0.939547768059322
best_params: {'estimator___C': 100.0, 'estimator___kernel': 'rbf'}
test_score: 0.9418401324420789
cv_results: {'mean_fit_time': array([   39.07549129,    28.40473619,    33.45557714,
    29.20987401,
       148.67102137,  1131.47076697, 10903.01463504]), 'std_fit_time': array([ 11.03966593,
     4.89655257,   4.13189165,   8.15821237,
      25.08032861, 179.68749082, 230.01098969]), 'mean_score_time': array([20.86162715,
    16.14955187, 17.7874877 ,  9.48515701,  9.02546082,
      6.96334014,  6.17813649]), 'std_score_time': array([5.38051363, 4.50619518,
    1.99933275, 4.44331625, 3.08762045,
     0.73045588, 0.38046924]), 'param_estimator___C': masked_array(data=[0.001, 0.01,
    0.1, 1.0, 10.0, 100.0, 1000.0],
          mask=[False, False, False, False, False, False, False],
     fill_value='?',
          dtype=object), 'param_estimator___kernel': masked_array(data=['rbf', 'rbf', 'rbf', '
    rbf', 'rbf', 'rbf', 'rbf'],
          mask=[False, False, False, False, False, False, False],
     fill_value='?',
          dtype=object), 'params': [{'estimator___C': 0.001, 'estimator___kernel': 'rbf'}, {'
    estimator___C': 0.01, 'estimator___kernel': 'rbf'}, {'estimator___C': 0.1, '
    estimator___kernel': 'rbf'}, {'estimator___C': 1.0, 'estimator___kernel': 'rbf'}, {'
    estimator___C': 10.0, 'estimator___kernel': 'rbf'}, {'estimator___C': 100.0, '
    estimator___kernel': 'rbf'}, {'estimator___C': 1000.0, 'estimator___kernel': 'rbf'}], '
    split0_test_score': array([0.41276075, 0.81575373, 0.89492923, 0.92001296,
    0.93347175,
     0.93727021, 0.9278255 ]), 'split1_test_score': array([0.41374747, 0.81561292,
    0.89751602, 0.92313734, 0.93609001,
     0.94000178, 0.93148203]), 'split2_test_score': array([0.42615478, 0.82953608, 0.90523   ,
     0.92713802, 0.93718153,
     0.93957299, 0.93079855]), 'split3_test_score': array([0.42927989, 0.82772978,
    0.90409654, 0.92749618, 0.93809786,
```

```
18      0.9409035 , 0.93215817]), 'split4_test_score': array([0.43007137, 0.82323246,
    ↪ 0.90289829, 0.92661924, 0.93753446,
19      0.93999036, 0.93105405]), 'mean_test_score': array([0.42240285, 0.82237299,
    ↪ 0.90093402, 0.92488075, 0.93647512,
20      0.93954777, 0.93066366]), 'std_test_score': array([0.00759029, 0.00583536, 0.00400139,
    ↪ 0.00288766, 0.00163848,
21      0.00121885, 0.00149191]), 'rank_test_score': array([7, 6, 5, 4, 2, 1, 3])}
```

Código A.2: Resultados gridsearch cv realizado para SVR en su parámetro Gamma.

```
1  best_score:  0.9337622189583931
2  best_params:  {'estimator___C': 100, 'estimator___gamma': 1.0, 'estimator___kernel': 'rbf'}
3  test_score:  0.9360458662197886
4  cv_results:  {'mean_fit_time': array([ 50.34671454,  36.3134841 ,  34.88758135,
       ↪ 27.54687791,
5         30.25886712,  37.73972778,  62.92991343, 474.21710186,
6        202.68504143]), 'std_fit_time': array([ 8.67454937,  7.92081864,  8.07851754,
       ↪ 5.55399244,  9.39071688,
7        10.13691619, 18.87146568, 48.09083592, 25.26465203]), 'mean_score_time': array
       ↪ ([21.84809966, 15.23019943, 15.87196398, 16.90324025, 14.77694912,
8        19.87960486, 10.58312802,  6.18827481,  7.61694918]), 'std_score_time': array
       ↪ ([5.6154237 , 5.75222429, 6.06449974, 4.53980788, 4.58354552,
9        1.85129898, 3.29120615, 0.2168788 , 0.85527703]), 'param_estimator___C':
       ↪ masked_array(data=[100, 100, 100, 100, 100, 100, 100, 100, 100, 100],
10            mask=[False, False, False, False, False, False, False, False,
11                  False],
12       fill_value='?',
13            dtype=object), 'param_estimator___gamma': masked_array(data=[1e-07, 1e-06, 1e
       ↪ -05, 0.0001, 0.001, 0.01, 0.1, 1.0,
14                  10.0],
15            mask=[False, False, False, False, False, False, False, False,
16                  False],
17       fill_value='?',
18            dtype=object), 'param_estimator___kernel': masked_array(data=['rbf', 'rbf', 'rbf', '
       ↪ rbf', 'rbf', 'rbf', 'rbf', 'rbf',
19                  'rbf'],
20            mask=[False, False, False, False, False, False, False, False,
21                  False],
22       fill_value='?',
23            dtype=object), 'params': [{'estimator___C': 100, 'estimator___gamma': 1e-07, '
       ↪ estimator___kernel': 'rbf'}, {'estimator___C': 100, 'estimator___gamma': 1e-06, '
       ↪ estimator___kernel': 'rbf'}, {'estimator___C': 100, 'estimator___gamma': 1e-05, '
       ↪ estimator___kernel': 'rbf'}, {'estimator___C': 100, 'estimator___gamma': 0.0001, '
       ↪ estimator___kernel': 'rbf'}, {'estimator___C': 100, 'estimator___gamma': 0.001, '
       ↪ estimator___kernel': 'rbf'}, {'estimator___C': 100, 'estimator___gamma': 0.01, '
       ↪ estimator___kernel': 'rbf'}, {'estimator___C': 100, 'estimator___gamma': 0.1, '
       ↪ estimator___kernel': 'rbf'}, {'estimator___C': 100, 'estimator___gamma': 1.0, '
       ↪ estimator___kernel': 'rbf'}, {'estimator___C': 100, 'estimator___gamma': 10.0, '
       ↪ estimator___kernel': 'rbf'}], 'split0_test_score': array([-0.04503901,  0.16634266,
       ↪ 0.63975876,  0.69751818,  0.7115857 ,
24        0.84099321,  0.87069418,  0.93724649,  0.92926264]), 'split1_test_score': array
```

```
    ↪ ([-0.0557798 ,  0.15267989,  0.62767566,  0.69059875,  0.70525472,
25    0.83933612,  0.87477826,  0.93408556,  0.92483117]), 'split2_test_score': array
    ↪ ([-0.05180529,  0.15826085,  0.63311799,  0.69303234,  0.70759533,
26    0.83711373,  0.87005049,  0.9336696 ,  0.92703934]), 'split3_test_score': array
    ↪ ([-0.04839724,  0.16269049,  0.63423155,  0.68897951,  0.70316149,
27    0.83809718,  0.87094352,  0.93296378,  0.92590773]), 'split4_test_score': array
    ↪ ([-0.0471173 ,  0.16287092,  0.63956968,  0.69786439,  0.71198003,
28    0.83908474,  0.86663661,  0.93084568,  0.92145257]), 'mean_test_score': array
    ↪ ([-0.04962773,  0.16056896,  0.63487073,  0.69359864,  0.70791545,
29    0.838925  ,  0.87062061,  0.93376222,  0.92569869]), 'std_test_score': array
    ↪ ([0.00378054, 0.00470532, 0.00449939, 0.00358374, 0.00345756,
30   0.00129908, 0.00259203, 0.00206867, 0.00258166]), 'rank_test_score': array([9, 8, 7, 6,
    ↪ 5, 4, 3, 1, 2])}
```

Código A.3: Resultados gridsearch cv realizado para SVR en su parámetro Épsilon.

```
1  best_score:  0.9328218667514939
2  best_params:  {'estimator___C': 100, 'estimator___epsilon': 0.1, 'estimator___gamma': 1, '
       ↪ estimator___kernel': 'rbf'}
3  test_score:  0.9379967145367817
4  cv_results:  {'mean_fit_time': array([1.07672782e+03, 9.61051397e+02, 9.76176565e+02,
       ↪ 2.46255631e+00,
5       3.92591476e-02]), 'std_fit_time': array([1.70499353e+02, 1.66384070e+02, 9.50849679e
       ↪ +01, 2.03808920e-01,
6       3.17465194e-02]), 'mean_score_time': array([1.02648684e+01, 1.03079029e+01,
       ↪ 7.23246603e+00, 4.12827301e-01,
7       2.00867653e-03]), 'std_score_time': array([7.81489347e-01, 2.91874617e-01, 1.20375018e
       ↪ +00, 3.80857359e-02,
8       1.54181444e-05]), 'param_estimator___C': masked_array(data=[100, 100, 100, 100,
       ↪ 100],
9                    mask=[False, False, False, False, False],
10      fill_value='?',
11           dtype=object), 'param_estimator___epsilon': masked_array(data=[0.001, 0.01, 0.1,
       ↪ 1.0, 10.0],
12                   mask=[False, False, False, False, False],
13      fill_value='?',
14           dtype=object), 'param_estimator___gamma': masked_array(data=[1, 1, 1, 1, 1],
15                   mask=[False, False, False, False, False],
16      fill_value='?',
17           dtype=object), 'param_estimator___kernel': masked_array(data=['rbf', 'rbf', 'rbf', '
       ↪ rbf', 'rbf'],
18                   mask=[False, False, False, False, False],
19      fill_value='?',
20           dtype=object), 'params': [{'estimator___C': 100, 'estimator___epsilon': 0.001, '
       ↪ estimator___gamma': 1, 'estimator___kernel': 'rbf'}, {'estimator___C': 100, '
       ↪ estimator___epsilon': 0.01, 'estimator___gamma': 1, 'estimator___kernel': 'rbf'}, {'
       ↪ estimator___C': 100, 'estimator___epsilon': 0.1, 'estimator___gamma': 1, '
       ↪ estimator___kernel': 'rbf'}, {'estimator___C': 100, 'estimator___epsilon': 1.0, '
       ↪ estimator___gamma': 1, 'estimator___kernel': 'rbf'}, {'estimator___C': 100, '
       ↪ estimator___epsilon': 10.0, 'estimator___gamma': 1, 'estimator___kernel': 'rbf'}], '
       ↪ split0_test_score': array([ 0.92731855,  0.92748266,  0.92894799,  0.81067629,
```

```
    ↪ -0.82681802]), 'split1_test_score': array([ 0.93311899,  0.93336642,  0.93477043,
    ↪ 0.81492084, -1.26674362]), 'split2_test_score': array([ 0.93284387,  0.93323266,
    ↪ 0.93372354,  0.82011532, -0.93426445]), 'split3_test_score': array([ 0.93271757,
    ↪ 0.93291754,  0.933723  ,  0.81776667, -0.88740123]), 'split4_test_score': array([
    ↪ 0.93124425,  0.93136942,  0.93294437,  0.81370032, -0.90473774]), 'mean_test_score':
    ↪ array([ 0.93144865,  0.93167374,  0.93282187,  0.81543589, -0.96399301]), '
    ↪ std_test_score': array([0.00216547, 0.0022136 , 0.00202207, 0.00326377, 0.15539381]),
    ↪  'rank_test_score': array([3, 2, 1, 4, 5])}
```

A.2. GPR

Código A.4: Resultados gridsearch cv sobre el kernel realizado para GPR en la predicción de la frecuencia media.

```
1  best_score:  0.9303453246253071
2  best_params:  {'kernel': RBF(length_scale=1) + WhiteKernel(noise_level=1), '
       ↪ n_restarts_optimizer': 4}
3  test_score:  0.9307458262471024
4  cv_results:  {'mean_fit_time': array([1674.07856973, 9310.17048049,  404.14404798,
       ↪ 3371.67861605]),
5            'std_fit_time': array([ 684.36388124, 1428.93970416,  120.17008898,
       ↪ 1763.68164353]),
6            'mean_score_time': array([20.65531532, 14.81729873, 12.43265653, 10.85949087]),
7            'std_score_time': array([1.28812291, 7.21108368, 8.81775639, 2.90778191]),
8            'param_kernel': masked_array(data=[RBF(length_scale=1),
9                 RBF(length_scale=1) + WhiteKernel(noise_level=1),
10                ExpSineSquared(length_scale=1, periodicity=1),
11                DotProduct(sigma_0=1)],
12           mask=[False, False, False, False],
13      fill_value='?',
14           dtype=object), 'param_n_restarts_optimizer': masked_array(data=[4, 4, 4, 4],
15            mask=[False, False, False, False],
16      fill_value='?',
17           dtype=object), 'params': [{'kernel': RBF(length_scale=1), 'n_restarts_optimizer':
       ↪ 4},
18                              {'kernel': RBF(length_scale=1) + WhiteKernel(noise_level=1),
       ↪ 'n_restarts_optimizer': 4},
19                              {'kernel': ExpSineSquared(length_scale=1, periodicity=1), '
       ↪ n_restarts_optimizer': 4},
20                              {'kernel': DotProduct(sigma_0=1), 'n_restarts_optimizer': 4}],
21                              'split0_test_score': array([0.87963292, 0.93116955,        nan,
       ↪ 0.75549936]),
22                              'split1_test_score': array([-6.09659462e-05,  9.28472712e-01,
       ↪ 8.30016540e-01,  4.60763991e-01]),
23                              'split2_test_score': array([-1.98958690e-05,  9.31393712e-01,
       ↪ 7.36394983e-01,  7.87726578e-01]),
24                              'mean_test_score': array([0.29318402, 0.93034532,        nan,
       ↪ 0.66799664]),
25                              'std_test_score': array([0.41468199, 0.0013273 ,        nan,
       ↪ 0.14712507]),
26                              'rank_test_score': array([3, 1, 4, 2])}
```

Código A.5: Resultados gridsearch cv sobre el kernel realizado para GPR en la predicción del ancho de banda.

```
best_score:  0.9540771075029499
best_params:  {'kernel': RBF(length_scale=1) + WhiteKernel(noise_level=1), '
    ↪ n_restarts_optimizer': 4}
test_score:  0.9574470706167482
cv_results:  {'mean_fit_time': array([2542.3214047 , 8593.48940539, 4392.33910044]),
          'std_fit_time': array([1198.64291362, 2141.12262278, 1249.49114418]),
          'mean_score_time': array([22.57548086, 13.99111891, 10.50500886]),
          'std_score_time': array([6.89652174, 4.89431459, 4.34014534]),
          'param_kernel': masked_array(data=[RBF(length_scale=1),
                RBF(length_scale=1) + WhiteKernel(noise_level=1),
                DotProduct(sigma_0=1)],
          mask=[False, False, False],
      fill_value='?',
         dtype=object), 'param_n_restarts_optimizer': masked_array(data=[4, 4, 4],
          mask=[False, False, False],
      fill_value='?',
         dtype=object), 'params': [{'kernel': RBF(length_scale=1), 'n_restarts_optimizer':
    ↪ 4},
                            {'kernel': RBF(length_scale=1) + WhiteKernel(noise_level=1),
    ↪ 'n_restarts_optimizer': 4},
                            {'kernel': DotProduct(sigma_0=1), 'n_restarts_optimizer': 4}],
                            'split0_test_score': array([0.93102341, 0.95423132, 0.62953963]),
                            'split1_test_score': array([-2.49205037e-06,  9.52002838e-01,
    ↪ 6.17712643e-01]),
                            'split2_test_score': array([-2.85272603e-06,  9.55997162e-01,
    ↪ 6.29769771e-01]),
                            'mean_test_score': array([0.31033936, 0.95407711, 0.62567401]),
                            'std_test_score': array([0.43888991, 0.00163432, 0.00563032]),
                            'rank_test_score': array([3, 1, 2])}
```

Anexo B

Códigos

B.1. Códigos análisis positividad

Código B.1: Preprocesamiento de datos.

```
# -*- coding: utf-8 -*-
"""
Created on Fri Oct  1 18:07:14 2021

@author: Vicente
"""

#importar librerías
import scipy.io as sio
from scipy.io import savemat
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error as mse
from sklearn import preprocessing
from random import sample
import pickle
import os

##########################################################
#       Pre processing
##########################################################

dataDir1 = "D:/Otoño 2021/Titulo/Python/1 Modelo x Banda/Archivos Kagome/"
dataDir2 = "D:/Otoño 2021/Titulo/Python/1 Modelo x Banda/DataBase_FM y AB/
    ↪ KGM_GS/"
mats_e1 = []
for file in os.listdir( dataDir1 ) :
   mats_e1.append( sio.loadmat( dataDir1+file ) )

B={"banda1": None, "banda2": None, "banda3": None, "banda4": None, "banda5": None, "
    ↪ banda6": None, "banda7": None, "banda8": None, "Base": None}
i=0
for dato in B:
```

```
    xe1=mats_e1[i][dato]
    B[dato]=xe1
    i+=1

savemat(dataDir2+'Bandas_Base.mat', B)

##########################################################
#       Frec Media y Ancho de Banda
##########################################################
nb = 8
filas = B['banda1'].shape[0]
columnas = B['banda1'].shape[1]

ancho = [0 for col in range(filas)]
freq = [0 for col in range(filas)]

for jj in range(filas):
    bands = [[0 for col in range(columnas)] for row in range(nb)]
    anchomax = -1e10
    for j in range(nb):
        b = 'banda'+str((j+1))
        datos = B[b]
        bands[j] = datos[jj,:]
    for i in range(7):
        if len(bands[0:i]):
            maxi = np.max(bands[0:i+1])
            mini = np.min(bands[i+1:nb])
            delta = mini - maxi
        else:
            maxi = np.max(bands[i])
            mini = np.min(bands[i+1:nb])
            delta = mini - maxi
        if delta > anchomax:
            ancho[jj] = delta
            freq[jj] = maxi + delta/2
            anchomax = ancho[jj]

np.save(dataDir2+'freq', freq)
np.save(dataDir2+'ancho', ancho)

##################################
#   Graficos Band Gaps
##################################
ancho_banda = ancho[:]
frec_media = freq[:]
x_base = B['Base']
Pos_A = []
Pos_FM = []
Pos_X = []
ix_Pos_A = []
ix_Pos_FM = []
ix_Pos_X = []
```

```
84
85 for i in range(len(ancho)):
86     a = ancho[i]
87     fm = freq[i]
88     x = B['Base'][i]
89     if a > 0:
90         Pos_A.append(a)
91         ix_Pos_A.append(i)
92
93         Pos_FM.append(fm)
94         ix_Pos_FM.append(i)
95
96         Pos_X.append(x)
97         ix_Pos_X.append(i)
98
99 for index in sorted(ix_Pos_A, reverse=True):
100     del ancho_banda[index]
101
102 for index in sorted(ix_Pos_FM, reverse=True):
103     del frec_media[index]
104
105 for index in sorted(ix_Pos_X, reverse=True):
106     x_base = np.delete(x_base,index,0)
107
108 Datasets = {'100': [], '80': [], '50': [], '20': [], '10': [], '1': []}
109 n_pos = len(Pos_A)
110 Negativos = ((np.array([frec_media[:],ancho_banda[:]])).T).tolist()
111 Negativos = (np.hstack((Negativos,x_base))).tolist()
112 Y = (np.array([Pos_FM[:], Pos_A[:]])).T
113 for key,dato in Datasets.items():
114     if key == '100':
115         Datasets[key] = [Pos_X,Y]
116     elif key == '1':
117         Neg_X = B['Base']
118         Pos_y_Neg = (np.vstack((freq, ancho))).T
119         Datasets[key] = [Neg_X,Pos_y_Neg]
120     else:
121         n_samples = int((100*n_pos/int(key))-n_pos)
122         rndm_samples = np.array(sample(Negativos,n_samples))
123         Neg_FM_A = rndm_samples[:,0:2]
124         Neg_X = np.vstack((Pos_X,rndm_samples[:,2:10]))
125         Pos_y_Neg = np.vstack((Y, Neg_FM_A))
126         Datasets[key] = [Neg_X,Pos_y_Neg]
127
128
129
130 ##########################################################
131 #       Scaling
132 ##########################################################
133 for key,datos in Datasets.items():
134     Xbase = datos[0]
135     Y = datos[1]
```

```
    # estandarizar los datos FREQ
    scaler = preprocessing.StandardScaler().fit(Y)
    pickle.dump(scaler, open(dataDir2+'scaler'+key+'.sav', 'wb'))
    Yscaled = scaler.transform(Y)
    #Guardar archivos
    np.save(dataDir2+'Y'+key+'.npy', Yscaled)
    np.save(dataDir2+'X'+key+'.npy', Xbase)
```

Código B.2: Entramiento modelos.

```
import numpy as np
import pickle
from sklearn.model_selection import train_test_split
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import WhiteKernel, RBF
from sklearn.multioutput import MultiOutputRegressor
from sklearn.svm import SVR

#################################
#      GAUSSIAN PROCESS
#################################

Estructuras = ['KGM_GS']#, 'TETRA_GS', 'PC_GS']
Datasets = ['100','80','50','20','10', '1']
for estruc in Estructuras:
    for key in Datasets:
        # Directorios
        dataDir1 = "D:/Otoño 2021/Titulo/Python/1 Modelo x Banda/DataBase_FM y AB/"
    ↪ +estruc+"/"
        dataDir2 = "D:/Otoño 2021/Titulo/Python/1 Modelo x Banda/DataBase_FM y AB/"
    ↪ +estruc+"/"+key+"/GPR2/"
        # Carga de datasets
        Xt = np.load(dataDir1+'X'+key+'.npy')
        Yt = np.load(dataDir1+'Y'+key+'.npy')
        # Train Test Split
        X_train, X_test, y_train, y_test = train_test_split(Xt, Yt, test_size=0.20)
        yf_train = y_train[:,0]
        ya_train = y_train[:,1]
        yf_test = y_test[:,0]
        ya_test = y_test[:,1]

        # Kernel
        modelo_kernel = RBF() + WhiteKernel()

        #Modelo Frecuencia
        # Entramiento de modelo
        GPR_FREQ = GaussianProcessRegressor(kernel=modelo_kernel,
    ↪ n_restarts_optimizer=4)
        GPR_FREQ.fit(X_train, yf_train)
        # Guardar modelo
        pickle.dump(GPR_FREQ, open(dataDir2+'GPR_FREQ'+key+'.sav', 'wb'))
        print('_____')
```

```
        print('test_score freq'+key+': ', GPR_FREQ.score(X_test, ya_test))

        # #Modelo Ancho banda
        # Entramiento de modelo
        GPR_Ancho = GaussianProcessRegressor(kernel=modelo_kernel,
    ↪ n_restarts_optimizer=4)
        GPR_Ancho.fit(X_train, ya_train)
        # Guardar modelo
        pickle.dump(GPR_Ancho, open(dataDir2+'GPR_FREQ'+key+'.sav', 'wb'))
        print('_____')
        print('test_score ancho'+key+': ', GPR_Ancho.score(X_test, ya_test))

        #guardar resultados
        Ypf, yf_sigma = GPR_FREQ.predict(X_test, return_std=True)
        Ypa, ya_sigma = GPR_Ancho.predict(X_test, return_std=True)
        Yp = np.hstack((Ypf,Ypa))
        Yp_len = int(len(Yp))
        Yp_len05 = int(len(Yp)/2)
        Yp = ((np.array([Yp[0:Yp_len05],Yp[Yp_len05:Yp_len]])).T)
        np.save(dataDir2+'Yp'+key+'.npy', Yp)
        np.save(dataDir2+'y_test'+key+'.npy', y_test)
        np.save(dataDir2+'yf_sigma'+key+'.npy', yf_sigma)
        np.save(dataDir2+'ya_sigma'+key+'.npy', ya_sigma)
        del GPR_FREQ,GPR_Ancho

##################################
#  SUPPORT VECTOR REGRESSION
##################################

Estructuras = ['KGM_GS']#, 'TETRA_GS', 'PC_GS']
Datasets = ['100','80','50','20','10', '1']
for estruc in Estructuras:
    for key in Datasets:
        # Directorios
        dataDir1 = "D:/Otoño 2021/Titulo/Python/1 Modelo x Banda/DataBase_FM y AB/"
    ↪ +estruc+"/"
        dataDir2 = "D:/Otoño 2021/Titulo/Python/1 Modelo x Banda/DataBase_FM y AB/"
    ↪ +estruc+"/"+key+"/SVR/"
        # Carga de datasets
        Xt = np.load(dataDir1+'X'+key+'.npy')
        Yt = np.load(dataDir1+'Y'+key+'.npy')
        # Train Test Split
        X_train, X_test, y_train, y_test = train_test_split(Xt, Yt, test_size=0.20)

        # Entramiento de modelo
        MOR_SVR = MultiOutputRegressor(SVR(kernel='rbf', C=100, epsilon = 0.1, gamma
    ↪ ='scale'))
        MOR_SVR.fit(X_train, y_train)
        # Guardar modelo
        pickle.dump(MOR_SVR, open(dataDir2+'MOR_SVR'+key+'.sav', 'wb'))
        print('_____')
        print('test_score'+key+': ', MOR_SVR.score(X_test, y_test))
```

```
        #guardar resultados
        Yp = MOR_SVR.predict(X_test)
        np.save(dataDir2+'Yp'+key+'.npy', Yp)
        np.save(dataDir2+'y_test'+key+'.npy', y_test)
        del MOR_SVR
```

Código B.3: Gráficos de resultados.

```
############
#   SVR
############
import pickle
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import mean_squared_error as mse
from scipy.io import savemat

Estructuras = ['KGM_GS']#, 'TETRA_GS', 'PC_GS']
Datasets = ['100','80','50','20','10']
for estruc in Estructuras:
    for key in Datasets:
        # Directorios
        dataDir1 = "D:/Otoño 2021/Titulo/Python/1 Modelo x Banda/DataBase_FM y AB/"
    ↪ +estruc+"/"
        dataDir2 = "D:/Otoño 2021/Titulo/Python/1 Modelo x Banda/DataBase_FM y AB/"
    ↪ +estruc+"/"+key+"/SVR/"
        # Carga de datasets
        Xt = np.load(dataDir1+'X'+key+'.npy')

        scaler = pickle.load(open(dataDir1+'scaler'+key+'.sav', 'rb'))
        Yp = np.load(dataDir2+'Yp'+key+'.npy')
        y_test = np.load(dataDir2+'y_test'+key+'.npy')

        #reconstruir
        X=scaler.inverse_transform(y_test)
        Xp=scaler.inverse_transform(Yp)
        #Data dict
        Graficos = {"freq": [X[:,0], Xp[:,0]],"ancho": [X[:,1], Xp[:,1]]}
        #Guardar resultados
        Bandas_test = {"freq": X[:,0],"ancho": X[:,1]}
        Bandas_testP = {"freq": Xp[:,0],"ancho": Xp[:,1]}
        savemat(dataDir2+"Bandas_test"+key+".mat", Bandas_test)
        savemat(dataDir2+"Bandas_testP"+key+".mat", Bandas_testP)

        ##########################################################
        #       Graficos y evaluacion
        ##########################################################
        freq_scaled = Graficos['freq'][1]
        freq = Graficos['freq'][0]
        ancho_scaled = Graficos['ancho'][1]
        ancho = Graficos['ancho'][0]

```

```
    #RMSE y RMSPE
    freq_rmse=np.sqrt(mse(freq,freq_scaled))
    ancho_rmse=np.sqrt(mse(ancho,ancho_scaled))

    plt.figure()
    plt.bar('Frecuencia',freq_rmse)
    plt.bar('Ancho',ancho_rmse)
    plt.title(key+' % Positivos'+': RMSE de Frecuencia media y Ancho')
    plt.xlabel('RMSE (Hz)')
    plt.show()

    ##########################################
    #      Band Gap
    ##########################################
    plt.figure()
    plt.plot(np.array(freq)*1e-3,np.array(ancho)/np.array(freq),'x')
    plt.title(key+' % Positivos'+': Band gap normalizado vs Frecuencia media')
    plt.xlabel('Frecuencia media (kHz)')
    plt.ylabel('Band Gap Normalizado')
    plt.show()

    plt.figure()
    plt.plot(freq,ancho,'x')
    plt.title(key+' % Positivos'+': Band gap vs Frecuencia media')
    plt.xlabel('Frecuencia media (Hz)')
    plt.ylabel('Band Gap (Hz)')
    plt.show()

    SSerror = np.sum((np.array(ancho)-np.array(ancho_scaled))**2)
    SStotal = np.sum((np.array(ancho)-np.mean(np.array(ancho)))**2)
    R2 = 1-SSerror/SStotal
    Titulo = key+' % Positivos'+': Ancho de Banda, R2=' + str(round(R2, 3))

    plt.figure()
    plt.plot(ancho,ancho_scaled,'x')
    plt.plot([np.min([ancho, ancho_scaled]),np.max([ancho, ancho_scaled])],[np.min([ancho
↪ , ancho_scaled]),np.max([ancho, ancho_scaled])],'--r')
    plt.title(Titulo)
    plt.xlabel('Real')
    plt.ylabel('Predicho')
    plt.show()

    SSerror = np.sum((np.array(freq)-np.array(freq_scaled))**2)
    SStotal = np.sum((np.array(freq)-np.mean(np.array(freq)))**2)
    R2 = 1-SSerror/SStotal
    Titulo = key+' % Positivos'+': Frecuencia Media, R2=' + str(round(R2, 3))

    plt.figure()
    plt.plot(freq,freq_scaled,'x')
    plt.plot([np.min([freq, freq_scaled]),np.max([freq, freq_scaled])],[np.min([freq,
↪ freq_scaled]),np.max([freq, freq_scaled])],'--r')
    plt.title(Titulo)
```

```
        plt.xlabel('Real')
        plt.ylabel('Predicho')
        plt.show()



#########################################################
#   Evaluación predicción positivos
#########################################################
Estructuras = ['KGM_GS']#, 'TETRA_GS', 'PC_GS']
Datasets = ['100','80','50','20','10']
for estruc in Estructuras:
    for key in Datasets:
        # Directorios
        dataDir1 = "D:/Otoño 2021/Titulo/Python/1 Modelo x Banda/DataBase_FM y AB/"
    ↪ +estruc+"/"
        dataDir2 = "D:/Otoño 2021/Titulo/Python/1 Modelo x Banda/DataBase_FM y AB/"
    ↪ +estruc+"/"+key+"/SVR/"
        # Carga de datasets
        Xt = np.load(dataDir1+'X'+key+'.npy')

        scaler = pickle.load(open(dataDir1+'scaler'+key+'.sav', 'rb'))
        Yp = np.load(dataDir2+'Yp'+key+'.npy')
        y_test = np.load(dataDir2+'y_test'+key+'.npy')

        #reconstruir
        X=scaler.inverse_transform(y_test)
        Xp=scaler.inverse_transform(Yp)
        #Data dict
        Graficos = {"freq": [X[:,0], Xp[:,0]],"ancho": [X[:,1], Xp[:,1]]}
        #Guardar resultados
        Bandas_test = {"freq": X[:,0],"ancho": X[:,1]}
        Bandas_testP = {"freq": Xp[:,0],"ancho": Xp[:,1]}
        savemat(dataDir2+"Bandas_test"+key+".mat", Bandas_test)
        savemat(dataDir2+"Bandas_testP"+key+".mat", Bandas_testP)

        ###########################################################
        #       Graficos y evaluacion
        ###########################################################
        freq_scaled = Graficos['freq'][1]
        freq = Graficos['freq'][0]
        ancho_scaled = Graficos['ancho'][1]
        ancho = Graficos['ancho'][0]

        samples = len(ancho)
        Pos_BandGap = []
        ix_BandGap = []
        Pos_BandGapP = []

        Pos_FM = []
        Pos_FMP = []

```

```
        for i in range(samples):
            a = ancho[i]
            ap = ancho_scaled[i]
            f = freq[i]
            fp = freq_scaled[i]
            if a > 0:
                Pos_BandGap.append(a)
                Pos_BandGapP.append(ap)
                ix_BandGap.append(i)

                Pos_FM.append(f)
                Pos_FMP.append(fp)

        SSerror = np.sum((np.array(Pos_BandGap)-np.array(Pos_BandGapP))**2)
        SStotal = np.sum((np.array(Pos_BandGap)-np.mean(np.array(Pos_BandGap)))**2)
        R2 = 1-SSerror/SStotal
        Titulo = key+' % Positivos'+': Ancho de Banda, R2=' + str(round(R2, 3))

        plt.figure()
        plt.plot(Pos_BandGap,Pos_BandGapP,'x')
        plt.plot([np.min([Pos_BandGap, Pos_BandGapP]),np.max([Pos_BandGap,
    ↪ Pos_BandGapP])],[np.min([Pos_BandGap, Pos_BandGapP]),np.max([Pos_BandGap,
    ↪  Pos_BandGapP])],'--r')
        plt.title(Titulo)
        plt.xlabel('Real')
        plt.ylabel('Predicho')
        plt.show()

        SSerror = np.sum((np.array(Pos_FM)-np.array(Pos_FMP))**2)
        SStotal = np.sum((np.array(Pos_FM)-np.mean(np.array(Pos_FM)))**2)
        R2 = 1-SSerror/SStotal
        Titulo = key+' % Positivos'+': Frecuencia Media, R2=' + str(round(R2, 3))

        plt.figure()
        plt.plot(Pos_FM,Pos_FMP,'x')
        plt.plot([np.min([Pos_FM, Pos_FMP]),np.max([Pos_FM, Pos_FMP])],[np.min([
    ↪ Pos_FM, Pos_FMP]),np.max([Pos_FM, Pos_FMP])],'--r')
        plt.title(Titulo)
        plt.xlabel('Real')
        plt.ylabel('Predicho')
        plt.show()



##########
#  GPR
##########
import pickle
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import mean_squared_error as mse
from scipy.io import savemat
```

```
192
193 Estructuras = ['KGM_GS']#, 'TETRA_GS', 'PC_GS']
194 Datasets = ['100','80','50','20','10']
195 for estruc in Estructuras:
196     for key in Datasets:
197         # Directorios
198         dataDir1 = "D:/Otoño 2021/Titulo/Python/1 Modelo x Banda/DataBase_FM y AB/"
        ↪ +estruc+"/"
199         dataDir2 = "D:/Otoño 2021/Titulo/Python/1 Modelo x Banda/DataBase_FM y AB/"
        ↪ +estruc+"/"+key+"/GPR/"
200         # Carga de datasets
201         Xt = np.load(dataDir1+'X'+key+'.npy')
202
203         scaler = pickle.load(open(dataDir1+'scaler'+key+'.sav', 'rb'))
204         Yp = np.load(dataDir2+'Yp'+key+'.npy')
205         Yp_len = int(len(Yp))
206         Yp_len05 = int(len(Yp)/2)
207         Yp = ((np.array([Yp[0:Yp_len05],Yp[Yp_len05:Yp_len]])).T)
208         y_test = np.load(dataDir2+'y_test'+key+'.npy')
209
210         #reconstruir
211         X=scaler.inverse_transform(y_test)
212         Xp=scaler.inverse_transform(Yp)
213         #Data dict
214         Graficos = {"freq": [X[:,0], Xp[:,0]],"ancho": [X[:,1], Xp[:,1]]}
215         #Guardar resultados
216         Bandas_test = {"freq": X[:,0],"ancho": X[:,1]}
217         Bandas_testP = {"freq": Xp[:,0],"ancho": Xp[:,1]}
218         savemat(dataDir2+"Bandas_test"+key+".mat", Bandas_test)
219         savemat(dataDir2+"Bandas_testP"+key+".mat", Bandas_testP)
220
221         ##########################################################
222         #       Graficos y evaluacion
223         ##########################################################
224         freq_scaled = Graficos['freq'][1]
225         freq = Graficos['freq'][0]
226         ancho_scaled = Graficos['ancho'][1]
227         ancho = Graficos['ancho'][0]
228
229         #RMSE y RMSPE
230         freq_rmse=np.sqrt(mse(freq,freq_scaled))
231         ancho_rmse=np.sqrt(mse(ancho,ancho_scaled))
232
233         plt.figure()
234         plt.bar('Frecuencia',freq_rmse)
235         plt.bar('Ancho',ancho_rmse)
236         plt.title(key+' % Positivos'+': RMSE de Frecuencia media y Ancho')
237         plt.xlabel('RMSE (Hz)')
238         plt.show()
239
240         ##########################################################
241         #       Band Gap
```

```
        ##########################################################
        plt.figure()
        plt.plot(np.array(freq)*1e-3,np.array(ancho)/np.array(freq),'x')
        plt.title(key+' % Positivos'+': Band gap normalizado vs Frecuencia media')
        plt.xlabel('Frecuencia media (kHz)')
        plt.ylabel('Band Gap Normalizado')
        plt.show()

        plt.figure()
        plt.plot(freq,ancho,'x')
        plt.title(key+' % Positivos'+': Band gap vs Frecuencia media')
        plt.xlabel('Frecuencia media (Hz)')
        plt.ylabel('Band Gap (Hz)')
        plt.show()

        SSerror = np.sum((np.array(ancho)-np.array(ancho_scaled))**2)
        SStotal = np.sum((np.array(ancho)-np.mean(np.array(ancho)))**2)
        R2 = 1-SSerror/SStotal
        Titulo = key+' % Positivos'+': Ancho de Banda, R2=' + str(round(R2, 3))

        plt.figure()
        plt.plot(ancho,ancho_scaled,'x')
        plt.plot([np.min([ancho, ancho_scaled]),np.max([ancho, ancho_scaled])],[np.min([ancho
    ↪ , ancho_scaled]),np.max([ancho, ancho_scaled])],'--r')
        plt.title(Titulo)
        plt.xlabel('Real')
        plt.ylabel('Predicho')
        plt.show()

        SSerror = np.sum((np.array(freq)-np.array(freq_scaled))**2)
        SStotal = np.sum((np.array(freq)-np.mean(np.array(freq)))**2)
        R2 = 1-SSerror/SStotal
        Titulo = key+' % Positivos'+': Frecuencia Media, R2=' + str(round(R2, 3))

        plt.figure()
        plt.plot(freq,freq_scaled,'x')
        plt.plot([np.min([freq, freq_scaled]),np.max([freq, freq_scaled])],[np.min([freq,
    ↪ freq_scaled]),np.max([freq, freq_scaled])],'--r')
        plt.title(Titulo)
        plt.xlabel('Real')
        plt.ylabel('Predicho')
        plt.show()



##########################################################
#   Evaluación predicción positivos
##########################################################
Estructuras = ['KGM_GS']#, 'TETRA_GS', 'PC_GS']
Datasets = ['100','80','50','20','10']
for estruc in Estructuras:
    for key in Datasets:
```

```
    # Directorios
    dataDir1 = "D:/Otoño 2021/Titulo/Python/1 Modelo x Banda/DataBase_FM y AB/"
    +estruc+"/"
    dataDir2 = "D:/Otoño 2021/Titulo/Python/1 Modelo x Banda/DataBase_FM y AB/"
    +estruc+"/"+key+"/SVR/"
    # Carga de datasets
    Xt = np.load(dataDir1+'X'+key+'.npy')

    scaler = pickle.load(open(dataDir1+'scaler'+key+'.sav', 'rb'))
    Yp = np.load(dataDir2+'Yp'+key+'.npy')
    y_test = np.load(dataDir2+'y_test'+key+'.npy')

    #reconstruir
    X=scaler.inverse_transform(y_test)
    Xp=scaler.inverse_transform(Yp)
    #Data dict
    Graficos = {"freq": [X[:,0], Xp[:,0]],"ancho": [X[:,1], Xp[:,1]]}
    #Guardar resultados
    Bandas_test = {"freq": X[:,0],"ancho": X[:,1]}
    Bandas_testP = {"freq": Xp[:,0],"ancho": Xp[:,1]}
    savemat(dataDir2+"Bandas_test"+key+".mat", Bandas_test)
    savemat(dataDir2+"Bandas_testP"+key+".mat", Bandas_testP)

    ##########################################################
    #       Graficos y evaluacion
    ##########################################################
    freq_scaled = Graficos['freq'][1]
    freq = Graficos['freq'][0]
    ancho_scaled = Graficos['ancho'][1]
    ancho = Graficos['ancho'][0]

    samples = len(ancho)
    Pos_BandGap = []
    ix_BandGap = []
    Pos_BandGapP = []

    Pos_FM = []
    Pos_FMP = []

    for i in range(samples):
        a = ancho[i]
        ap = ancho_scaled[i]
        f = freq[i]
        fp = freq_scaled[i]
        if a > 0:
            Pos_BandGap.append(a)
            Pos_BandGapP.append(ap)
            ix_BandGap.append(i)

            Pos_FM.append(f)
            Pos_FMP.append(fp)

```

```
        SSerror = np.sum((np.array(Pos_BandGap)-np.array(Pos_BandGapP))**2)
        SStotal = np.sum((np.array(Pos_BandGap)-np.mean(np.array(Pos_BandGap)))**2)
        R2 = 1-SSerror/SStotal
        Titulo = key+' % Positivos'+': Ancho de Banda, R2=' + str(round(R2, 3))

        plt.figure()
        plt.plot(Pos_BandGap,Pos_BandGapP,'x')
        plt.plot([np.min([Pos_BandGap, Pos_BandGapP]),np.max([Pos_BandGap,
    ↪ Pos_BandGapP])],[np.min([Pos_BandGap, Pos_BandGapP]),np.max([Pos_BandGap,
    ↪  Pos_BandGapP])],'--r')
        plt.title(Titulo)
        plt.xlabel('Real')
        plt.ylabel('Predicho')
        plt.show()

        SSerror = np.sum((np.array(Pos_FM)-np.array(Pos_FMP))**2)
        SStotal = np.sum((np.array(Pos_FM)-np.mean(np.array(Pos_FM)))**2)
        R2 = 1-SSerror/SStotal
        Titulo = key+' % Positivos'+': Frecuencia Media, R2=' + str(round(R2, 3))

        plt.figure()
        plt.plot(Pos_FM,Pos_FMP,'x')
        plt.plot([np.min([Pos_FM, Pos_FMP]),np.max([Pos_FM, Pos_FMP])],[np.min([
    ↪ Pos_FM, Pos_FMP]),np.max([Pos_FM, Pos_FMP])],'--r')
        plt.title(Titulo)
        plt.xlabel('Real')
        plt.ylabel('Predicho')
        plt.show()
```

B.2. Códigos gridsearch

Código B.4: Gridsearch SVR.

```
import numpy as np
import pickle
from sklearn.model_selection import train_test_split
from sklearn.multioutput import MultiOutputRegressor
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV

#####################################
#      DATASETS
#####################################
# Carga de datasets
Xt = np.load('X.npy')
Yt = np.load('Y.npy')

# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(Xt, Yt, test_size=0.20)

#####################################
```

```
#   Gridsearch
################################
#Parametros grid search
kernel_range = ['poly', 'rbf', 'sigmoid']
gamma_range = [1e-7,1e-6,1e-5,1e-4,1e-3,1e-2,1e-1,1e0,1e1]
c_range = [1e-3,1e-2,1e-1,1e0,1e1,1e2,1e3]
epsilon_range = [1e-3,1e-2,1e-1,1e0,1e1]
grid_param_svr = {'estimator___kernel':['rbf'],
            'estimator___epsilon': [0.1],
            'estimator___gamma': [1],
            'estimator___C': [10,15,25,50,75,100]
            }

#Iniciacion de modelo
Model = MultiOutputRegressor(SVR())
grid_search = GridSearchCV(Model, param_grid=grid_param_svr, cv=5, verbose=1,
    ↪ n_jobs=-1)
print('_____')
print('Entrenando ')
grid_search.fit(X_train, y_train)
print('Entrega de resultados ')
print('SVR:')
print('Best_score:',grid_search.best_score_)


################################
#       Resultados
################################

print('_____')
print('Guardando ')
print('best_score: ', grid_search.best_score_)
print('best_params: ', grid_search.best_params_)
print('test_score: ', grid_search.score(X_test, y_test))
print('cv_results: ', grid_search.cv_results_)
#guardar resultados
Yp = grid_search.predict(X_test)
np.save('Yp.npy', Yp)
np.save('y_test.npy', y_test)


################################
#       Best Models
################################
# Guardar modelo para cada banda
best_modelo = grid_search.best_params_
modelo_C = best_modelo['estimator___C']
modelo_kernel = best_modelo['estimator___kernel']
# Entramiento de modelo
MOR_SVR = MultiOutputRegressor(SVR(kernel=modelo_kernel, C=modelo_C))
MOR_SVR.fit(X_train, y_train)
```

```
# Guardar modelo
pickle.dump(MOR_SVR, open('MOR_SVR.sav', 'wb'))
```

Código B.5: Gridsearch GPR.

```
import numpy as np
import pickle
from sklearn.model_selection import train_test_split
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import WhiteKernel, DotProduct, RBF,
    ↪ ExpSineSquared
from sklearn.model_selection import GridSearchCV

#####################################
#       DATASETS
#####################################
# Carga de datasets
Xt = np.load('X.npy')
Yt = np.load('Y.npy')

# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(Xt, Yt, test_size=0.20)
yf_train = y_train[:,0]
ya_train = y_train[:,1]
yf_test = y_test[:,0]
ya_test = y_test[:,1]

#####################################
#   Gridsearch
#####################################
#Grid Search para frecuencia
#Parametros grid search
kernel1 = RBF()
kernel2 = RBF() + WhiteKernel()
#kernel3 = ExpSineSquared()
kernel4 = DotProduct()
kernel_range = [kernel1, kernel2, kernel4]
grid_param_gpr = {'kernel':kernel_range,
                  'n_restarts_optimizer':[4]}

#Iniciacion de modelo
Model = GaussianProcessRegressor()
grid_search = GridSearchCV(Model, param_grid=grid_param_gpr, cv=3, verbose=1,
    ↪ n_jobs=4)


print('_____')
print('Entrenando 1')
grid_search.fit(X_train, yf_train)
print('Entrega de resultados ')
print('GPR:')
print('Best_score:',grid_search.best_score_)
```

```
46
47
48 #Grid Search para ancho
49 #Modelo Ancho banda
50 Model2 = GaussianProcessRegressor()
51 grid_search2 = GridSearchCV(Model2, param_grid=grid_param_gpr, cv=3, verbose=1,
       ↪ n_jobs=4)
52 print('_____')
53 print('Entrenando 2')
54 grid_search2.fit(X_train, ya_train)
55 print('Entrega de resultados ')
56 print('GPR:')
57 print('Best_score:',grid_search2.best_score_)
58 print('_____')
59 print('Guardando 2')
60 print('best_score: ', grid_search2.best_score_)
61 print('best_params: ', grid_search2.best_params_)
62 print('test_score: ', grid_search2.score(X_test, ya_test))
63 print('cv_results: ', grid_search2.cv_results_)
64
65 ###################################
66 #       Resultados
67 ###################################
68
69 print('_____')
70 print('Guardando 1')
71 print('best_score: ', grid_search.best_score_)
72 print('best_params: ', grid_search.best_params_)
73 print('test_score: ', grid_search.score(X_test, yf_test))
74 print('cv_results: ', grid_search.cv_results_)
75
76 ###################################
77 #       Best Models
78 ###################################
79
80 # Guardar modelo para cada banda
81 best_modelo = grid_search.best_params_
82 modelo_kernel = best_modelo['kernel']
83 #Modelo Frecuencia
84 # Entramiento de modelo
85 GPR_FREQ = GaussianProcessRegressor(kernel=modelo_kernel, n_restarts_optimizer=4)
86 GPR_FREQ.fit(X_train, yf_train)
87 # Guardar modelo
88 pickle.dump(GPR_FREQ, open('GPR_FREQ.sav', 'wb'))
89
90
91 #Guardar modelo para cada banda
92 best_modelo = grid_search2.best_params_
93 modelo_kernel = best_modelo['kernel']
94 # Entramiento de modelo
95 GPR_Ancho = GaussianProcessRegressor(kernel=modelo_kernel, n_restarts_optimizer=4)
96 GPR_Ancho.fit(X_train, ya_train)
```

```
# Guardar modelo
pickle.dump(GPR_Ancho, open('GPR_FREQ.sav', 'wb'))
print('_____')
print('test_score: ', GPR_Ancho.score(X_test, ya_test))

#guardar resultados
Ypf, yf_sigma = GPR_FREQ.predict(X_test, return_std=True)
Ypa, ya_sigma = GPR_Ancho.predict(X_test, return_std=True)
Yp = []
for i in range(len(Ypf)):
    Yp.append([Ypf[i], Ypa[i]])
np.save('Yp.npy', Yp)
np.save('y_test.npy', y_test)
np.save('yf_sigma.npy', yf_sigma)
np.save('ya_sigma.npy', ya_sigma)
```