VFScript y Tipos de Resultados Exportados

¿Qué hace VFScript?

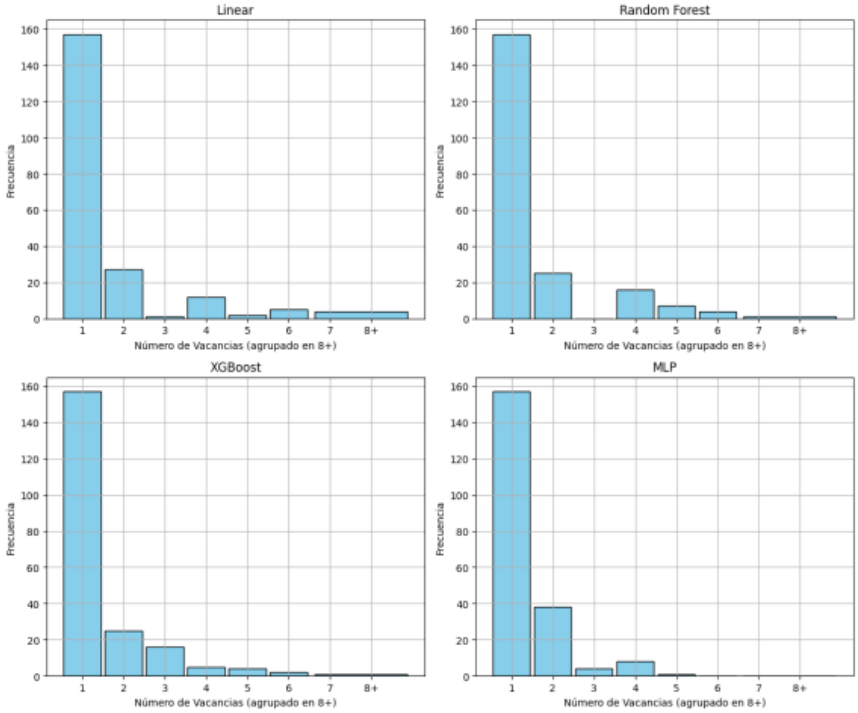
VFScript es un software diseñado para la detección, análisis y predicción de vacancias en muestras cristalinas a partir de simulaciones atómicas. Su funcionamiento se divide en tres etapas principales:

1. **Generación de Datos y Entrenamiento**: Se introducen defectos en una muestra inicial y se extraen características estructurales mediante OVITO. Estos datos se utilizan para entrenar modelos predictivos como Regresión Lineal, Random Forest, XGBoost y MLP.
2. **Identificación de Áreas Clave**: Se procesan muestras defectuosas para eliminar regiones sin defectos y se agrupan los átomos que rodean las vacancias en clusters mediante algoritmos de clustering.
3. **Predicciones**: Se emplean los modelos entrenados para estimar la presencia de vacancias y defectos en nuevas muestras, considerando métricas como área de superficie, volumen vacío y densidad atómica.

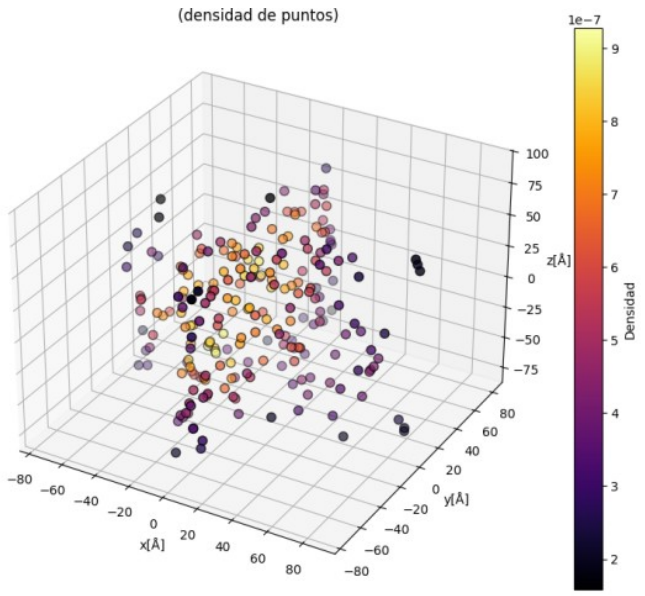
Tipos de Resultados Exportados

VFScript genera diversas representaciones visuales y estadísticas para facilitar la interpretación de los datos analizados. Entre los principales resultados se incluyen:

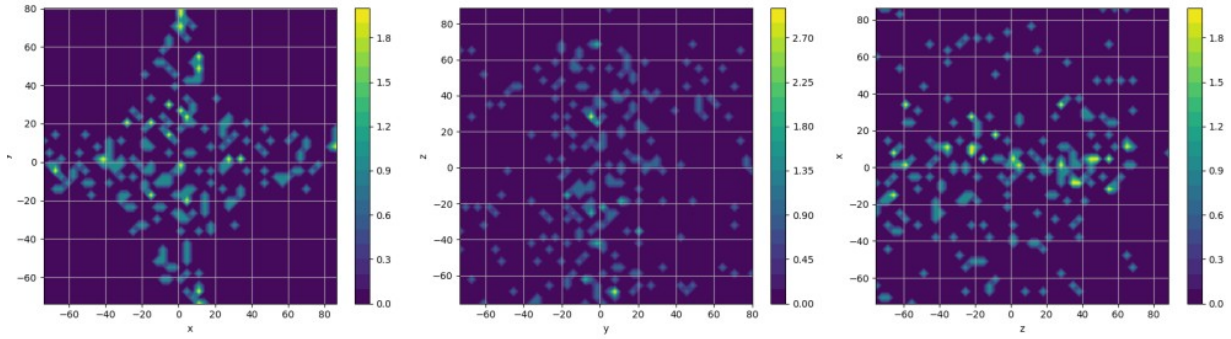
- **Histograma de Frecuencia de Vacancias Detectadas**: Representa la distribución de vacancias identificadas por los modelos de predicción.



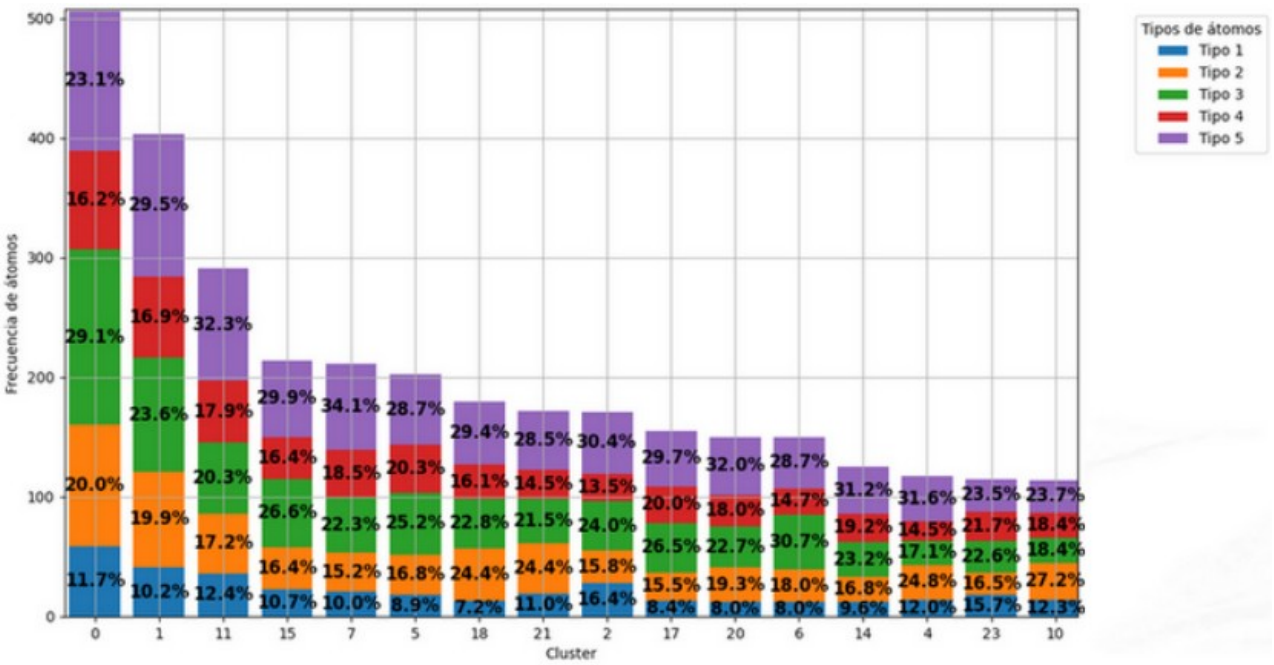
- **Mapa de Calor de Clusters de Centroides**: Visualización de la concentración y densidad de los clusters de defectos en la muestra.



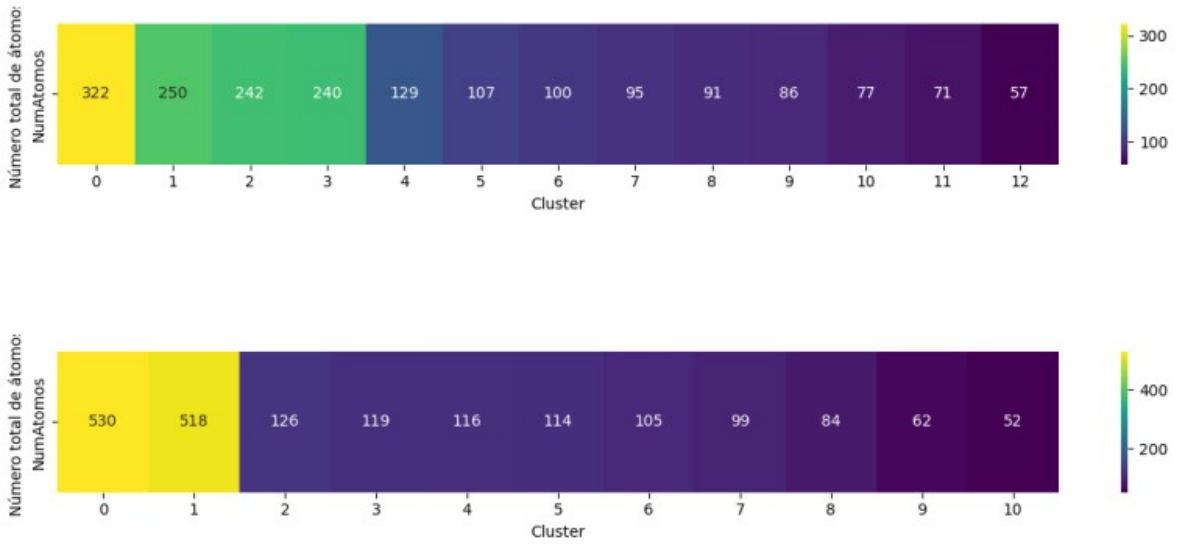
- **Mapa de Contorno de Densidad de Clusters de Defectos**: Muestra la distribución espacial de los defectos identificados.



- **Histograma de Población Total de Átomos por Cluster de Centroides**: Diferencia los clusters según los porcentajes de partículas de distintos tipos que los componen.



- **Mapa de Calor de Densidad de Población de los Clusters de Centroides**: Indica la distribución de los clusters en función de su densidad de población.



Estos resultados permiten analizar la evolución de defectos en materiales cristalinos, identificar patrones en las particulas que son recurrentes en los defectos y evaluar su impacto en las propiedades estructurales y mecánicas del material estudiado.

Manual de Uso de VFScript en Visual Studio Code

Visual Studio Code es un entorno de desarrollo ligero, potente y altamente personalizable que facilita la ejecución de scripts en diversos lenguajes de programación. Su compatibilidad con entornos virtuales, integración con Git y depuración eficiente lo convierten en una excelente opción para desarrollar y ejecutar VFScript. Además, su terminal integrada permite una configuración rápida y sencilla del entorno necesario para correr el algoritmo.

Pasos para Utilizar VFScript en Visual Studio Code

1 .Abrir Visual Studio Code

Inicie Visual Studio Code y abra la carpeta donde se encuentra el código de VFScript.

2 .Abrir la Terminal

En la barra de menú, seleccione "Terminal" y luego haga clic en "Abrir Nueva Terminal". También puede utilizar el atajo de teclado:

Windows y Linux: Ctrl + Ñ

MacOS: Cmd + Ñ

3 . Crear y Activar un Entorno Virtual

Antes de ejecutar VFScript, es recomendable crear un entorno virtual para gestionar las dependencias. Los pasos varían según el sistema operativo:

En Linux:

```
python3 -m venv venv
source venv/bin/activate
```

En Windows:

```
python -m venv venv
venv\Scripts\activate
```

4 .Instalar las Dependencias Necesarias

Una vez activado el entorno virtual, instale los paquetes requeridos ejecutando el siguiente comando en la terminal:

```
pip install ovito pandas scikit-learn matplotlib xgboost numpy
```

5 . Configurar los Parámetros de Entrada

Abra el archivo input_params.py en Visual Studio Code. En este archivo, debe especificar:

La ruta del archivo de entrenamiento.

La lista de archivos que desea analizar para contabilizar vacancias y extraer información sobre defectos.

Ejemplo de configuración en input_params.py:

```
training_file = "ruta/archivo_de_entrenamiento.dump"
data_files = ["ruta/archivo1.dump", "ruta/archivo2.dump"]
```

6 . Ejecutar VFScript

Una vez configurado input_params.py, ejecute el algoritmo desde la terminal con el siguiente comando:

```
python vfscript.py
```

Esto iniciará el proceso de análisis de vacancias y extracción de información sobre defectos en los archivos especificados.

Parámetros de Configuración en input_params.py

El archivo input_params.py contiene una lista de diccionarios (CONFIG) con la configuración del procesamiento y entrenamiento. A continuación, se detallan los parámetros más importantes:

Parámetros Generales

- **training_step** *(bool)*: Indica si se debe realizar una nueva fase de entrenamiento (True) o solo ejecutar la predicción (False).
- **relax** *(str)*: Ruta del archivo de entrada utilizado para el cálculo de relajación de la estructura cristalina.
- **defect** *(list[str])*: Lista de archivos que describen defectos o vacancias a analizar. Puede contener múltiples rutas a archivos de simulación.
- **radius** *(int)*: Define el radio en el que se evalúan interacciones geométricas relevantes.

Parámetros de Suavizado y Filtrado

- **smoothing level** *(int)*: Nivel de suavizado aplicado a los datos generales.
- **smoothing_level_training** *(int)*: Nivel de suavizado específico para los datos de entrenamiento.

Parámetros de Clustering y Análisis

- **cutoff radius** *(int)*: Radio límite para determinar interacciones significativas.
- **radius_training** *(int)*: Radio utilizado en la selección/generación de datos durante el entrenamiento.

- **cluster tolerance** *(float)*: Define la distancia máxima para agrupar puntos en un mismo cluster.
- **divisions_of_cluster** *(int)*: Factor de división del cluster (valores recomendados: `1`, `2` o `3`).

Opciones de Métodos y Guardado

- **other method** *(bool)*: Si True, permite utilizar métodos alternativos en el análisis.
- **save_training** *(bool)*: Si True, guarda el estado del entrenamiento en cada iteración.

Parámetros de Simulación y Gráficos

- **strees** *(list[float])*: Lista que representa la deformación aplicada a la muestra en cada eje. Ejemplo: `[0.8, 1, 1]` indica un 20% de compresión en el eje X.
- **CDScanner** *(bool)*: Activa la generación de gráficos de defectos.
- **Histograma** *(bool)*: Habilita la visualización de histogramas de defectos.

Generación de Datos y Predicción

- **generic_data** *(bool)*: Si True, se generan datos sintéticos para mejorar el entrenamiento.
- **chat** *(bool)*: Activa o desactiva funcionalidades interactivas en el modelo.
- **iteraciones_clusterig** *(int)*: Define el número de iteraciones para el algoritmo de clustering.

Columnas Predictoras

- **PREDICTOR_COLUMNS** *(list[str])*: Lista de características utilizadas en la predicción. Pueden ser **[‘surface_area ’,’ filled_volume ’,’ cluster_size ’,’ mean_distance ’]**