

PROYECTO FINAL INDIVIDUAL

Predicción de la edad de los abulones



INTRODUCCIÓN

En el presente documento se describe una solución MLOps para un 'dataset' llamado 'Abalon.data' que es un conjunto de datos clásico utilizado en el aprendizaje automático, que contiene información sobre abulones, un tipo de molusco marino. Cada abulón tiene características asociadas, como la longitud, el diámetro, el peso de la carne, el peso de las vísceras, el peso del caparazón, etc.

El objetivo típico al trabajar con este conjunto de datos es predecir la edad de un abulón; ésta se puede determinar cortando la concha a través del cono, tiñendola y contando el número de anillos a través de un microscopio (como el tronco de un árbol) y sumándole el factor 1.5, una tarea tediosa que consume mucho tiempo. Otra forma de hacerlo es mediante las medidas físicas mencionadas, que son más fáciles de obtener y con las que se puede estimar el número de anillos y de ahí la edad del abulón.

.

A) DESARROLLO DE LA SOLUCIÓN PROPUESTA

A1.-Análisis y comprensión del dataset proporcionado mediante un Análisis

Exploratorio de Datos

La información de las variables contenidas en el dataset "Abalon.data" es la siguiente:

NOMBRE VARIABLE	ROLE	TIPO	DESCRIPCIÓN	UNIDADES
Sex	Atributo	Categórica	M, F, I (Infante)	
Length	Atributo	Continua	Medida más larga del caparazón	mm
Diameter	Atributo	Continua	Diámetro, perpendicular a la longitud	mm
Height	Atributo	Continua	Altura de carne con caparazón	mm
Whole_weight	Atributo	Continua	Peso del abulón completo	grms
Shucked_weight	Atributo	Continua	Peso de la carne	grms
Viscera_weight	Atributo	Continua	Peso de las vísceras	grms
Shell_weight	Atributo	Continua	Peso del caparazón	grms
Rings	Objetivo	Entera	Anillos	

El problema puede ser abordado al menos de dos maneras, la primera a partir de una solución de tipo clasificación tomando a la variable objetivo 'rings' como una variable

categoría ordinal y la segunda a partir de una solución de tipo regresión si se le toma como una variable no categórica continua (aunque es un tipo discreta); ésto último implicará que en algún momento tengan que redondearse las predicciones, lo que pudiera introducir un pequeño error de precisión aunque haya efectos cancelatorios de redondear a veces hacia arriba y en otras ocasiones hacia abajo.

Se propone una solución MLOps de tipo regresión para resolver el problema de encontrar la edad de un abulón a partir de los atributos que se encuentran en el 'dataset'.

Específicamente se desea predecir la cantidad de anillos que tiene si se conocen sus otras características físicas. Obviamente se tendrá que explorar el 'dataset' para encontrar y seleccionar el menor número posible de éstas características sin perder información importante de las características no seleccionadas.

Las variables de entrada más importantes pueden variar según el enfoque del análisis y la aplicación específica del modelo. Sin embargo, la longitud y el diámetro suelen ser consideradas como dos de las variables más importantes para predecir la edad de los abulones. debido a que son medidas físicas directas del abulón muy relacionadas con su crecimiento y desarrollo. En general, se espera que abulones más grandes tengan más anillos y, por lo tanto, sean más viejos. Así que, esas variables son dos candidatas naturales para ser consideradas como importantes para predecir la edad de los abulones.

Dependiendo del contexto el peso total o del caparazón también puede ser indicadores de la salud del abulón y por lo tanto de su edad.

Al realizar un análisis exploratorio de los datos se encontró lo siguiente:

- En la imagen (Fig 1: Histograma por variable atributo y variable objetivo) con los histogramas de las variables de entrada se nota que tanto la la variable de salida como las variables de entrada con una distribución normal más evidente son: 'shell_weigth', 'diameter' y 'lenght'
- En la figura (Fig 2: Mapa Calor Matriz Factores Correlación) de los mapas de calor se nota que los atributos más correlacionados con la variable objetivo 'ring' son nuevamente 'shell_weigth', 'diameter', 'lenght' y 'height' siendo 'shell_weigth' y 'diameter' las de coeficientes más altos. También son dos variables muy correlacionadas con los demás atributos (Fig 4: Correlaciones Pares Variables Tipo Atributo y Variable Objetivo), lo que implica que de éstas dos variables se pueden deducir las restantes y no considerar atributos redundantes sin perder información relevante
- En la imagen (Fig 3: Diagramas Caja Variables Tipo Atributo) con los gráficos de caja para las variables de atributos para 'shell_weigth' y 'diameter' se nota que existen valores atípicos que se pueden eliminar, en la parte superior para 'shell_weigth' de 0.95 a 1, y en la parte inferior para 'diameter' de 0 a 0.075) sin afectar a las demás variables
- En la imagen (Fig 4: Correlaciones Pares Variables Tipo Atributo y Variable Objetivo) se puede notar visualmente que hay correlación entre las variables de atributos 'shell_weigth' y 'diameter' con las demás variables atributo, es decir se puede prescindir de los segundos porque se comportan de manera parecida a los primeros

Debido a que la variable objetivo 'age' no existe en el dataset 'Abalon.data', es posible crearla como una nueva columna a partir de la variable 'rings' y entonces se puede deducir

a partir de ésta columna sumando el factor 1.5 y de algunas otras columnas de características, sin embargo, se desechó esa opción pues se consideró una redundancia

En conclusión, se determinó seleccionar al diámetro ('diameter') y al peso del caparazón del abulón ('shell_weigth') como las dos características más relevantes para predecir el número de anillos de la variable objetivo. De manera implícita se obtendrá la edad del Abulón que es la pregunta que se debe responder en última instancia

A2.-Determinación de la pregunta que se desea contestar con un modelo de aprendizaje automático

En la sección anterior se concluyó que las dos variables de tipo atributo más relevantes en el dataset 'Abalon.data' con las que se puede predecir la variable objetivo son 'shell_weigth' y 'diameter'. Por lo tanto la pregunta que se desea contestar con un modelo de aprendizaje automático para predecir la variable objetivo es: ¿Cuántos anillos tendrá un abulón conocidos el peso de su caparazón y su diámetro?. De manera implícita se obtendrá la edad del abulón pues solo bastará sumar el factor 1.5 al número de anillos obtenido

A3.-Identificar por qué se necesita una estrategia de MLOps para éste dataset

El objetivo principal del 'bootcamp' es aprender cómo aplicar un modelo con una estrategia MLOps a un 'dataset'. Sin embargo, específicamente para las predicciones del modelo de aprendizaje automático del dataset 'Abalon.data' , mi punto de vista es la siguiente :

- El problema a resolver es relativamente simple, sin estructuras de datos altamente complejas y con muy pocos atributos. Considero que no se requiere una implementación sofisticada en producción con un flujo completo de trabajo de MLOps
 - El proyecto no tiene requisitos de producción exigentes, como escalabilidad, mantenibilidad, cumplimiento de regulaciones ni de seguridad. No se espera que el uso del modelo ni el tamaño de los datos tengan el potencial de crecer en el futuro ni que se vayan a desplegar muchos modelos que se deban mantener en producción a largo plazo
 - No requiere de un sistema en tiempo real ya que no maneja grandes volúmenes de datos ni que requiera de actualizaciones periódicas, pues la información parece formar parte de un proyecto de investigación que quedará estática. Al no esperar actualizaciones frecuentes implica que el modelo no necesitará reentrenamientos constantes. De hecho, en la página web donde se encuentra 'Abalon.data' se tiene la leyenda 'Expected update frequency' marcada con 'Never'
-

-
- Sus predicciones no son de naturaleza crítica, pues no tiene impactos comerciales o legales. Tampoco incurre en costos financieros por errores en las predicciones. Sus datos tampoco son sensibles ni privados

NOTA:

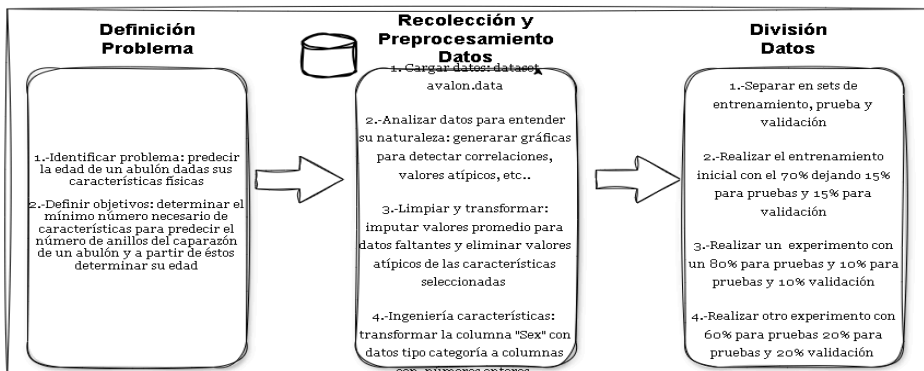
Obviamente, por cuestiones de formación y de aprendizaje de un tema tan importante, sí se adoptará una estrategia MLOPs para las predicciones del dataset 'Abalon.data'

A4.-Arquitectura del pipeline para ésta iniciativa de aprendizaje automático

A4.1) Básicamente se separará en 3 etapas principales: preparación de datos, creación del modelo y el despliegue del mismo, como se ve enseguida:

MLOPS

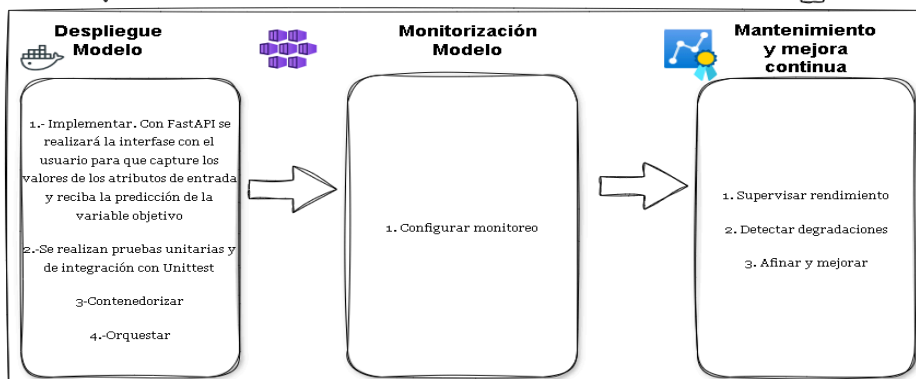
**PREPARACION
DATOS**



**CREACION
MODELO**

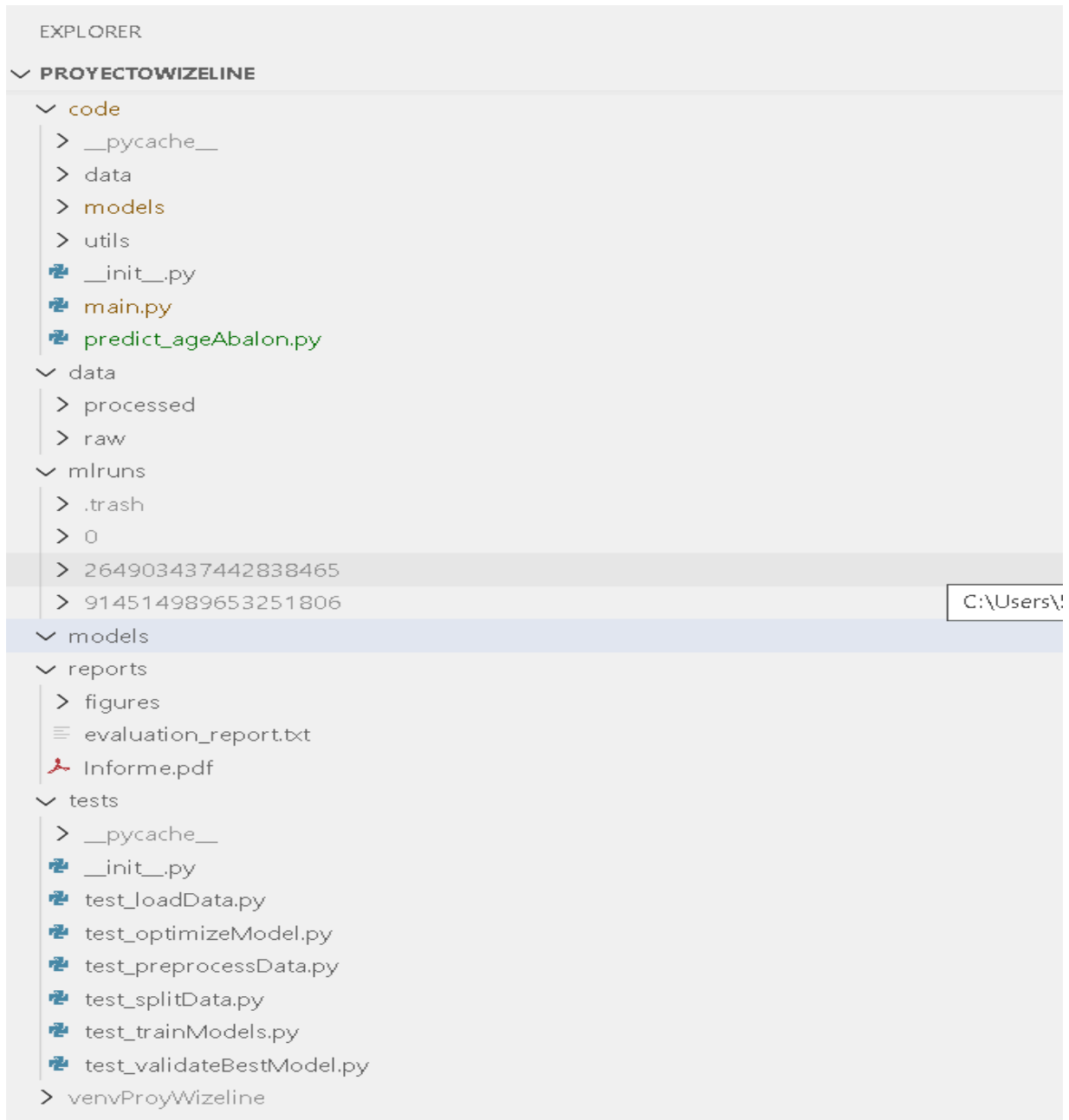


DESPLIEQUE



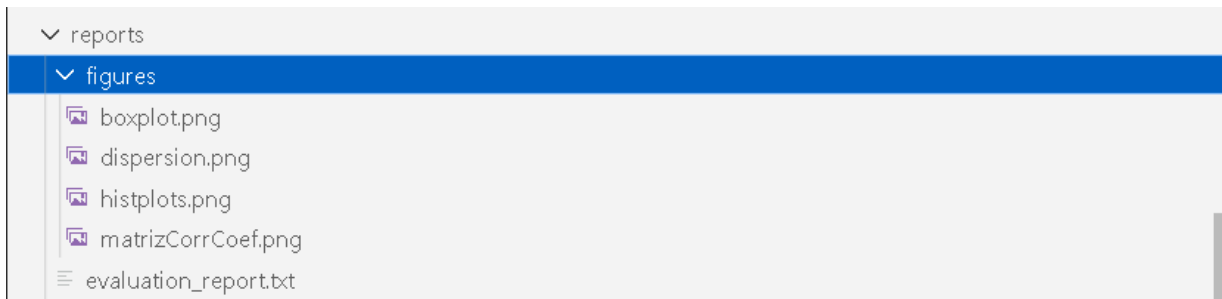
A4.2) Una tarea importante en éste punto es que se realizó la refactorización del script original de Python separándolo en módulos y en carpetas para distinguir la preparación


de datos, el modelo en sí y el despliegue



A4.3) Así por ejemplo entre algunas otras cosas, las gráficas generadas dinámicamente en tiempo de ejecución se guardan en la carpeta 'reports\figures' y en la carpeta

'reports\' se guarda 'evaluation_report' con los resultados más importantes de la corrida, como se ve enseguida



 evaluation_report: Bloc de notas

Archivo Edición Formato Ver Ayuda

Reporte de Validación de Modelos

Test mse

Linear Regression	5.629770992366412
Elastic Net	8.13740458015267
Random Forest	5.179389312977099

El mejor modelo optimizado es: RandomForestRegressor con mse: 5.058295964125561

A5.-Crear un modelo base para abordar tareas de predicción relacionadas con la pregunta. Este modelo no necesita una alta precisión, recall o puntuación F1; el objetivo es crear un modelo rápido para iteración.

A5.1) Se creó un modelo de 'machine learning' en el que se eligieron los siguientes algoritmos para responder a las tareas de predicción de los anillos y edad de una abulón a partir del peso del caparazón y su diámetro:

- 'linear regression',
- 'elastic net' y
- 'random forest'

En los 3 algoritmos se trataron de elegir hiper parámetros con valores default y los 3 se entrenaron con el mismo 'dataset' de entrenamiento ('train') del 70% del total de datos

A5.2) Enseguida con los datos de prueba ('test') se prueban los 3 algoritmos realizando predicciones del número de anillos de los abalones y al compararlos contra los datos reales de ese set de prueba se obtuvo la métrica 'mse' para los 3, determinando que el algoritmo 'random forest' era el que minimizaba los errores

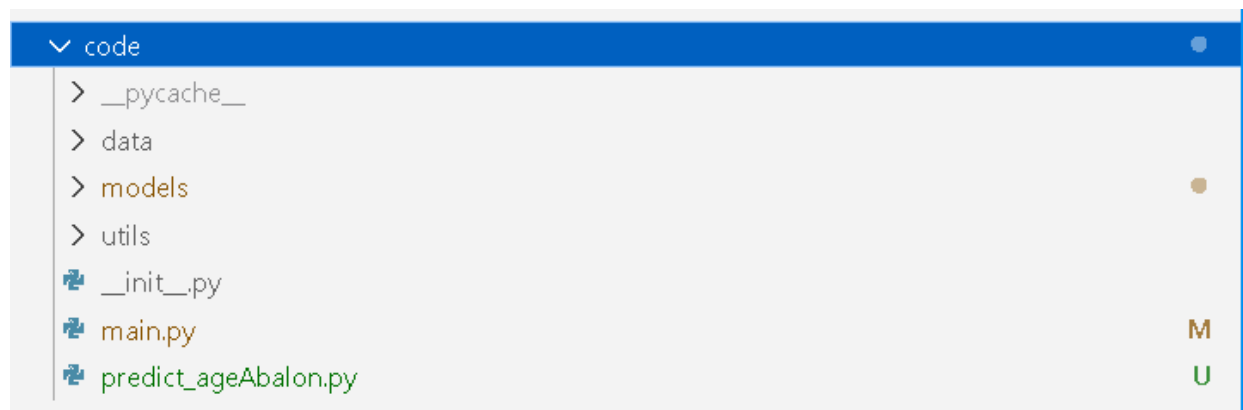
A5.3) Para la etapa de validación del algoritmo 'random forest', se realizaron 2 experimentos con él, variando el tamaño de los datos de prueba y los hiper parámetros a mediante dos 'gridSearch'. Los dos experimentos optimizan la métrica 'mse' con lo que se tuvo la seguridad de que el modelo estaba generalizando adecuadamente

A5.4) Se decidió utilizar a 'MLFlow' para el registro de los experimentos, con sus corridas, sus hiper parámetros y sus métricas con la finalidad de versionalizar y reproducir al experimento elegido

En la carpeta 'mlruns' del proyecto, 'mlflow' guarda todo lo relacionado con la ejecución de los experimentos, incluidas corridas, hiper parámetros, métricas, artefactos y otros archivos de configuración como requerimientos y dependencias. Nótese:



A5.5) Respecto a la forma en cómo el usuario puede utilizar el modelo para predicciones, se desarrolló el script de python 'predict_ageAbalon.py' que hace uso de las librerías 'FastAPI' y 'Pydantic' para ofrecer al usuario una interfaz sencilla para enviar al modelo los valores de las variables de entrada (los atributos 'Shell_weight' y 'Diameter' de un abulón), cargar y ejecutar el modelo ya entrenado y regresar el resultado de la variable objetivo ('Rings' y por deducción 'Age' del abulón)



Basta ejecutar en la terminal el comando:

uvicorn code.predict_ageAbalon:app --reload

Y en el navegador de internet revisar la 'FastAPI' en la dirección:

<http://127.0.0.1:8000/docs>

FastAPI 0.1.0 OAS 3.1
/openapi.json

default

POST	/predict_abalon	Predict Abalon	▼
GET	/	Read Root	▼

La ruta GET solo da la bienvenida a la API. La ruta POST es la que permite enviarle al modelo la características del abulón del cual deseamos predecir su número de anillos y por deducción su edad. Solo debemos abrir la ruta del POST y ejecutarla con 'Try it out' para indicar los valores de entrada (en nuestro ejemplo será 0.114 grs de peso del caparazón y 0.295 mm de diámetro del caparazón) y dar clic sobre el botón 'Execute' para que el modelo se ejecute:

Request body required application/json

```
{  "Shell weight": 0.114,  "Diameter": 0.295}
```

Execute

Lo anterior mostrará una sección 'responses' con el resultado de la predicción, en éste ejemplo el número de anillos es 9 y por lo tanto la edad del abulón es de 10.5 años:

Responses

Curl

```
curl -X 'POST' \  'http://127.0.0.1:8000/predict_abalon' \  -H 'accept: application/json' \  -H 'Content-Type: application/json' \  -d '{  "Shell weight": 0.114,  "Diameter": 0.295  }'
```

Request URL

```
http://127.0.0.1:8000/predict_abalon
```

Server response

Code	Details
200	<div>Response body</div> <pre>{ "predicted_rings": 9, "predicted_age": 10.5}</pre> <div>Download</div>

B) OTROS ASPECTOS DE LA SOLUCIÓN PROPUESTA

B1.-¿Qué elementos matemáticos se consideraron en esta decisión?

- En la parte de la exploración de datos básicamente se consideraron análisis de entre pares de variables como los coeficientes de correlación y las dispersiones de los datos
- En la parte de la selección del algoritmo de aprendizaje se consideró el error cuadrático medio (mse), que es una medida comúnmente utilizada para evaluar la precisión de un modelo del tipo abordado. Es una métrica útil para comparar diferentes modelos o ajustes de un mismo modelo y que suma los cuadrados de las diferencias entre los valores reales y los predichos

B2.-¿Cómo se integrarán nuevos datos?

Dada la naturaleza crítica del versionamiento, trazabilidad y reproducibilidad de los experimentos que se realizan en un modelo MLOps, en modelo que nos atañe se utiliza "git" para versionar el código del modelo y "dvc" para versionar los datos asociados a una versión del código. El flujo básico entre "git" y "dvc" es como sigue:

Datos con DVC:

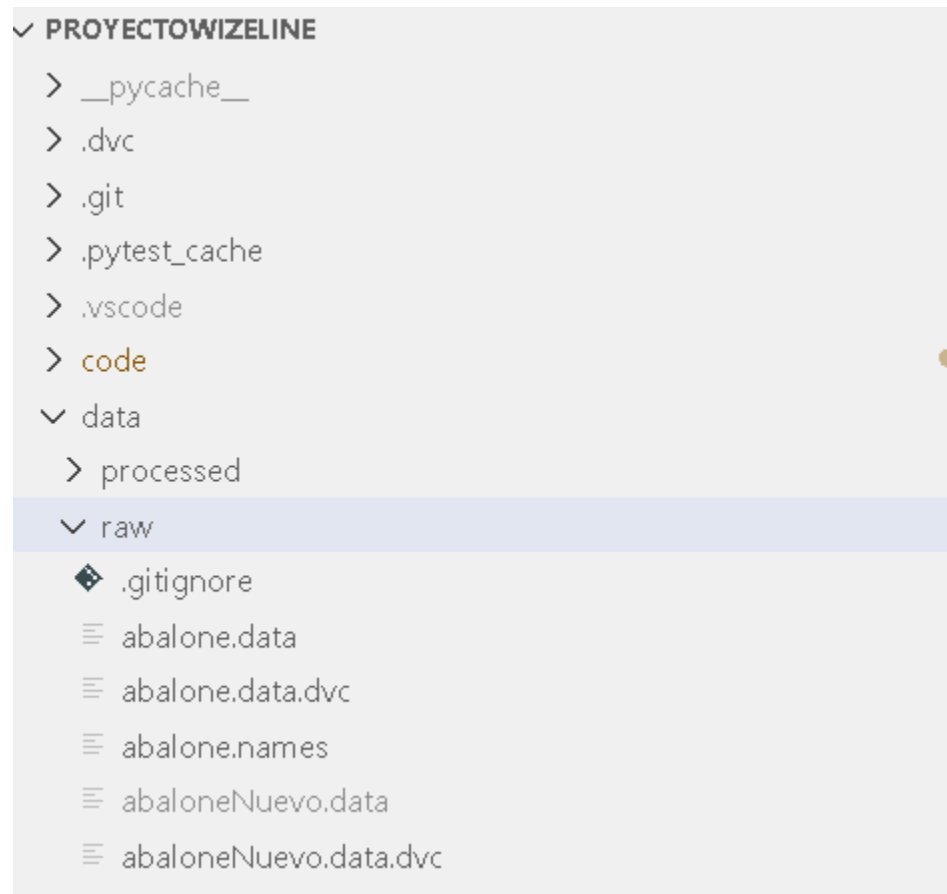
- DVC proporciona un sistema de control de versiones para datos, permitiendo rastrear y versionar conjuntos de datos
- Cada archivo de datos, como ``abalone.data`` o ``abaloneNuevo.data``, es gestionado por DVC. Sin embargo, en el repositorio de Git solo se incluyen archivos ``dvc``: ``abalone.data.dvc``, ``abaloneNuevo.data.dvc`` etc., pero no a los archivos con los datos: ``abalone.data``, ``abaloneNuevo.data``, etc.
- Los archivos ``dvc`` actúan como metadatos que apuntan a la ubicación o versión específica de los datos. Estos archivos son ligeros y contienen información sobre cómo acceder a la versión de los datos correspondiente

Código con Git:

- Git gestiona las versiones de los scripts de código, lo que incluye todo el código relacionado con el preprocesamiento de datos, entrenamiento de modelos, evaluación, etc.
 - En el repositorio Git, solo se incluyen archivos ``dvc`` que hacen referencia a los datos versionados en DVC. Los datos reales no se almacenan en Git
 - Esta separación mantiene el repositorio Git más liviano y enfocado en el código, evitando la carga de datos grandes en el historial de Git
-

Relación entre datos y código:

- Cuando el código necesita utilizar nuevos datos o se modifican los datos existentes, se actualizan los archivos '.dvc'
- Estas actualizaciones en los archivos '.dvc', que apuntan a versiones específicas de los datos, se registran junto con los cambios en el código en un 'commit' de Git
- Esta relación explícita entre los datos y el código permite una trazabilidad clara de qué versión del código está asociada con qué versión de los datos, lo que facilita la reproducción de experimentos y la colaboración en equipo



Cuando se desea ejecutar el modelo se realiza de la manera siguiente:

PROBLEMS OUTPUT TERMINAL JUPYTER GITLENS

```
(venvProyWizeline) PS C:\Users\52477\Documents\ProyectoWizeLine> python code\main.py data\raw\abalone.data.dvc .\
```

lo que lanzará la ejecución del modelo para una versión de datos 'abalone.data.dvc' y el modelo buscará en el repositorio de dvc donde está el archivo de datos correspondiente

PROBLEMS OUTPUT TERMINAL JUPYTER GITLENS

```
0 Sex 4176 non-null object
1 Length 4176 non-null float64
2 Diameter 4176 non-null float64
3 Height 4176 non-null float64
4 Whole_weight 4176 non-null float64
5 Shucked_weight 4176 non-null float64
6 Viscera_weight 4176 non-null float64
7 Shell_weight 4176 non-null float64
8 Rings 4176 non-null int32
dtypes: float64(7), int32(1), object(1)
memory usage: 277.4+ KB
```

```
data_Fr.shape: (3492, 11)
test_size: 0.15
Inicia entrenamiento ...
Inician pruebas ...
```

Test mse

```
-----
Linear Regression 5.629771
Elastic Net 8.137405
Random Forest 5.179389
```

```
Inicia experimento 1 ...
```

PROBLEMS OUTPUT TERMINAL JUPYTER GITLENS

```
random_state=42)
Corrida 3 model RandomForestRegressor
```

	Test mse	Val mse
RandomForestRegressor	5.179389	5.058296

Inicia validación del modelo ...

experiment_id	run_id	model	me_val
264903437442838465	bd61ab3df9994567827c897e7f15b661	RandomForestRegressor	5.058295964125561

	Shell_weight	Diameter	Rings_Predicted	Rings
0	0.0700	0.260	8.0	9.0
1	0.2040	0.385	11.0	14.0
2	0.2740	0.425	12.0	11.0
3	0.1130	0.315	8.0	10.0
4	0.1250	0.350	9.0	13.0
..
441	0.3145	0.470	11.0	12.0
442	0.0600	0.255	8.0	7.0
443	0.2650	0.470	10.0	10.0
444	0.1620	0.375	9.0	8.0
445	0.3250	0.505	11.0	11.0

[446 rows x 4 columns]

F I N A L I Z A C I O N

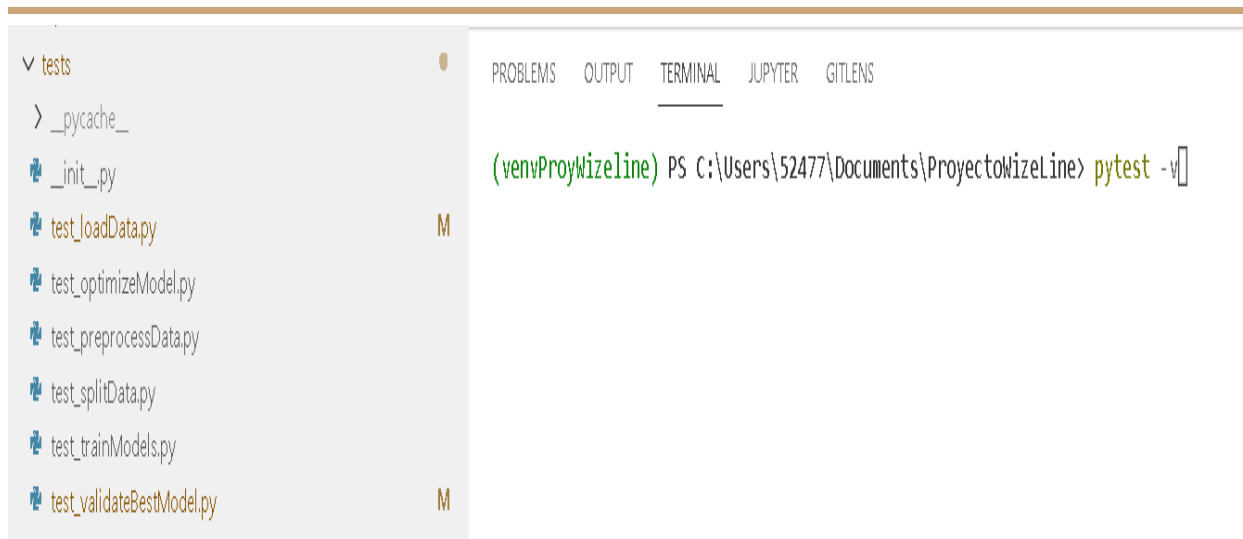
B3.-¿Cómo se medirá el drifting?

B4.-¿Se considera la prueba en el desarrollo del pipeline?

Se desarrollaron 6 scripts con un total de 12 pruebas unitarias e integración con la librería unittest a funciones de diferentes módulos desarrollados como se ve a continuación:



Es posible ejecutar en la terminal todas las pruebas con el comando `pytest -v`



```
(venvProyWizeline) PS C:\Users\52477\Documents\ProyectoWizeline> pytest -v
```

El comando anterior ocasionará que de manera automática se lleven a cabo 12 pruebas que serán verificadas y se mostrará el resultado de éstas, como se ve a continuación:



```
ms\Python\Python310\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\52477\Documents\ProyectoWizeline
configfile: pytest.ini
collected 12 items

tests/test_loadData.py::TestDataLoading::test_column_conversion PASSED [ 8%]
tests/test_loadData.py::TestDataLoading::test_column_names PASSED [ 16%]
tests/test_loadData.py::TestDataLoading::test_data_loading PASSED [ 25%]
tests/test_loadData.py::TestDataLoading::test_data_types PASSED [ 33%]
tests/test_optimizeModel.py::TestYourFunctions::test_gen_gridSearchHiperParam PASSED [ 41%]
tests/test_preprocessData.py::TestPreprocessData::test_preprocess_data_imputation PASSED [ 50%]
tests/test_preprocessData.py::TestPreprocessData::test_preprocess_data_output PASSED [ 58%]
tests/test_splitData.py::TestSplitDataset::test_split_dataset_proportions PASSED [ 66%]
tests/test_splitData.py::TestSplitDataset::test_split_dataset_sizes PASSED [ 75%]
tests/test_trainModels.py::TestInitialTrainModels::test_initialTrainModels PASSED [ 83%]
tests/test_validateBestModel.py::TestValidation::test_output_format PASSED [ 91%]
tests/test_validateBestModel.py::TestValidation::test_output_type PASSED [100%]
```


C) GRÁFICAS

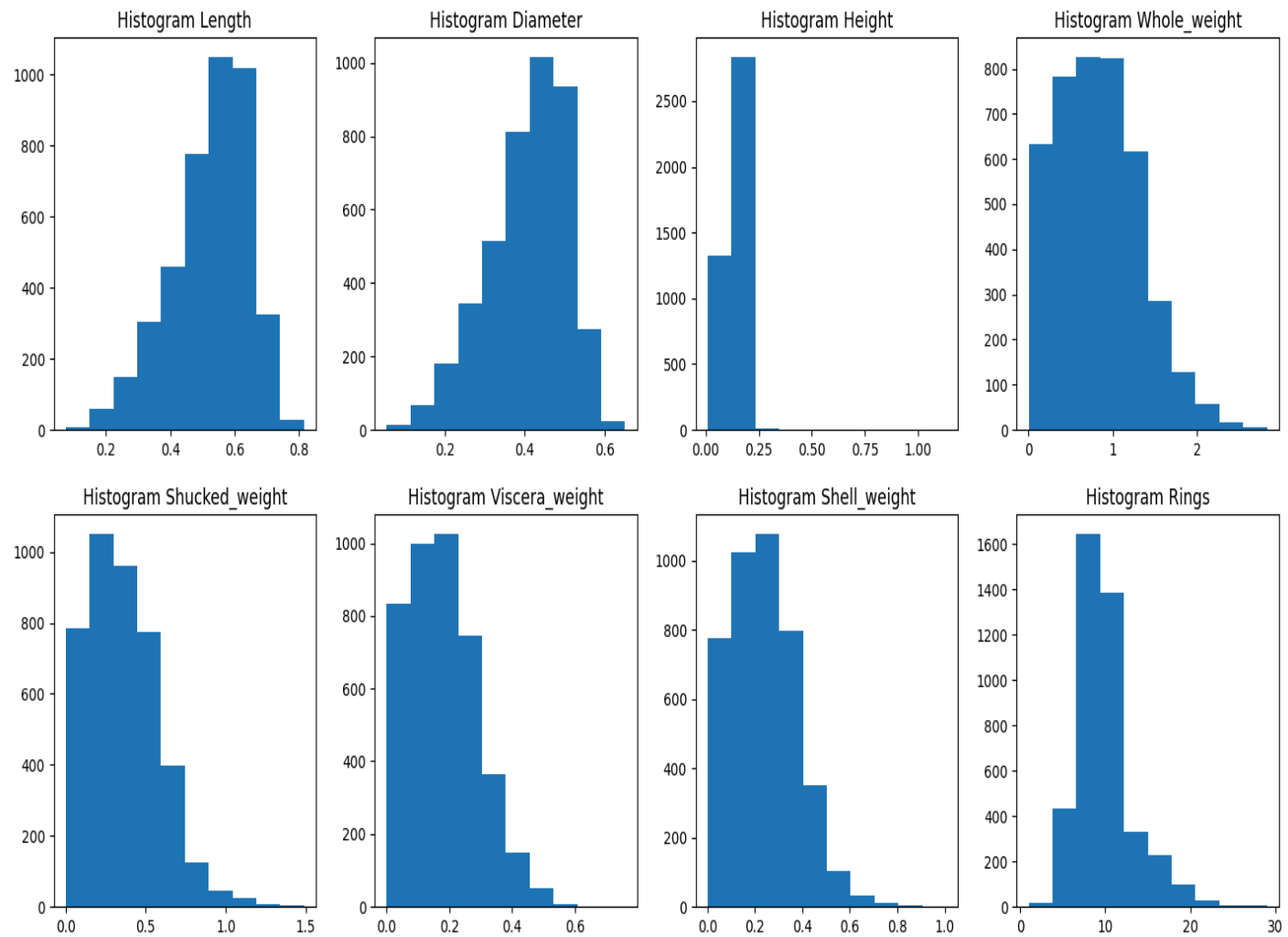


Fig 1: Histograma por variable atributo y variable objetivo

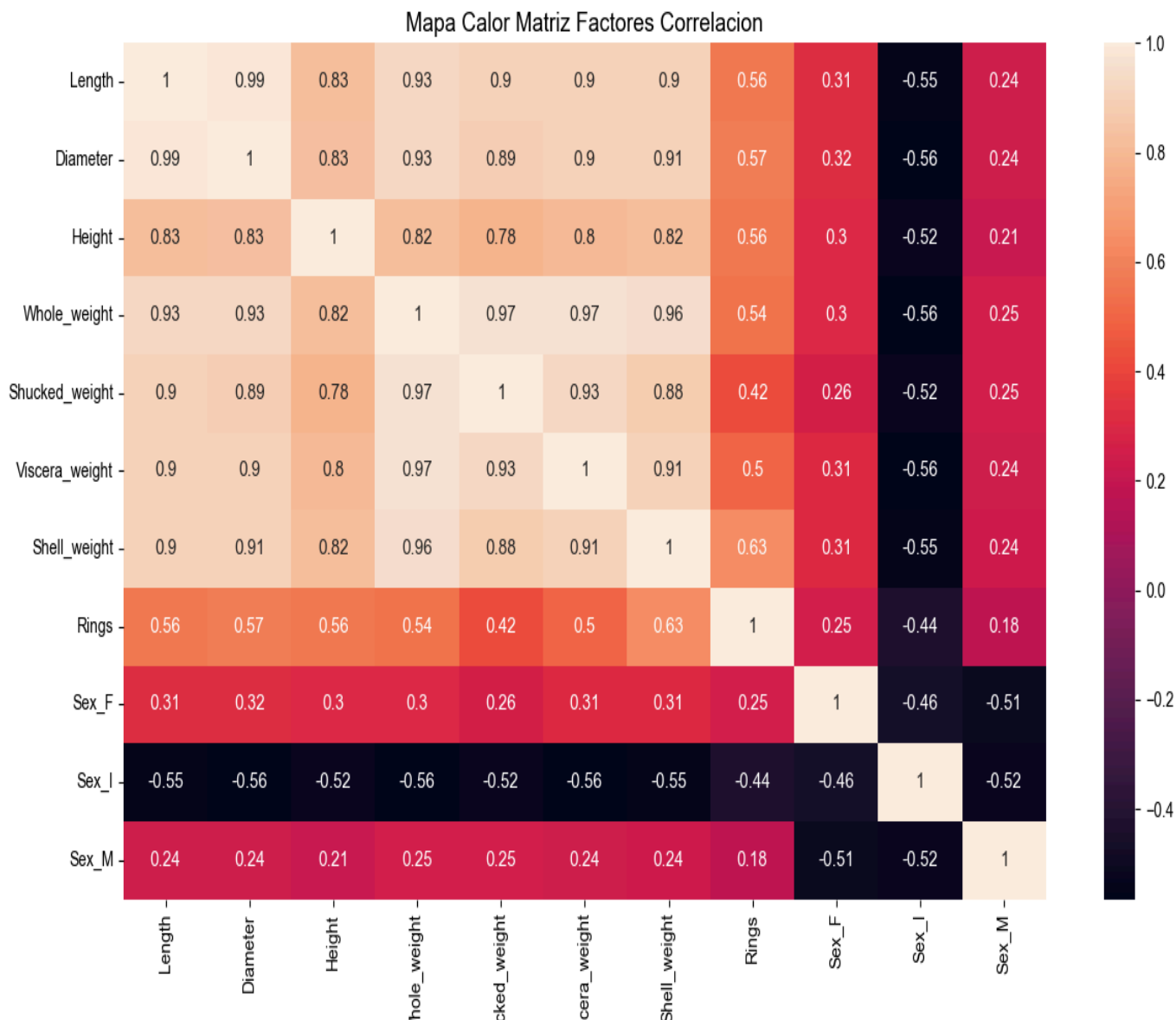


Fig 2: Mapa Calor Matriz Factores Correlación

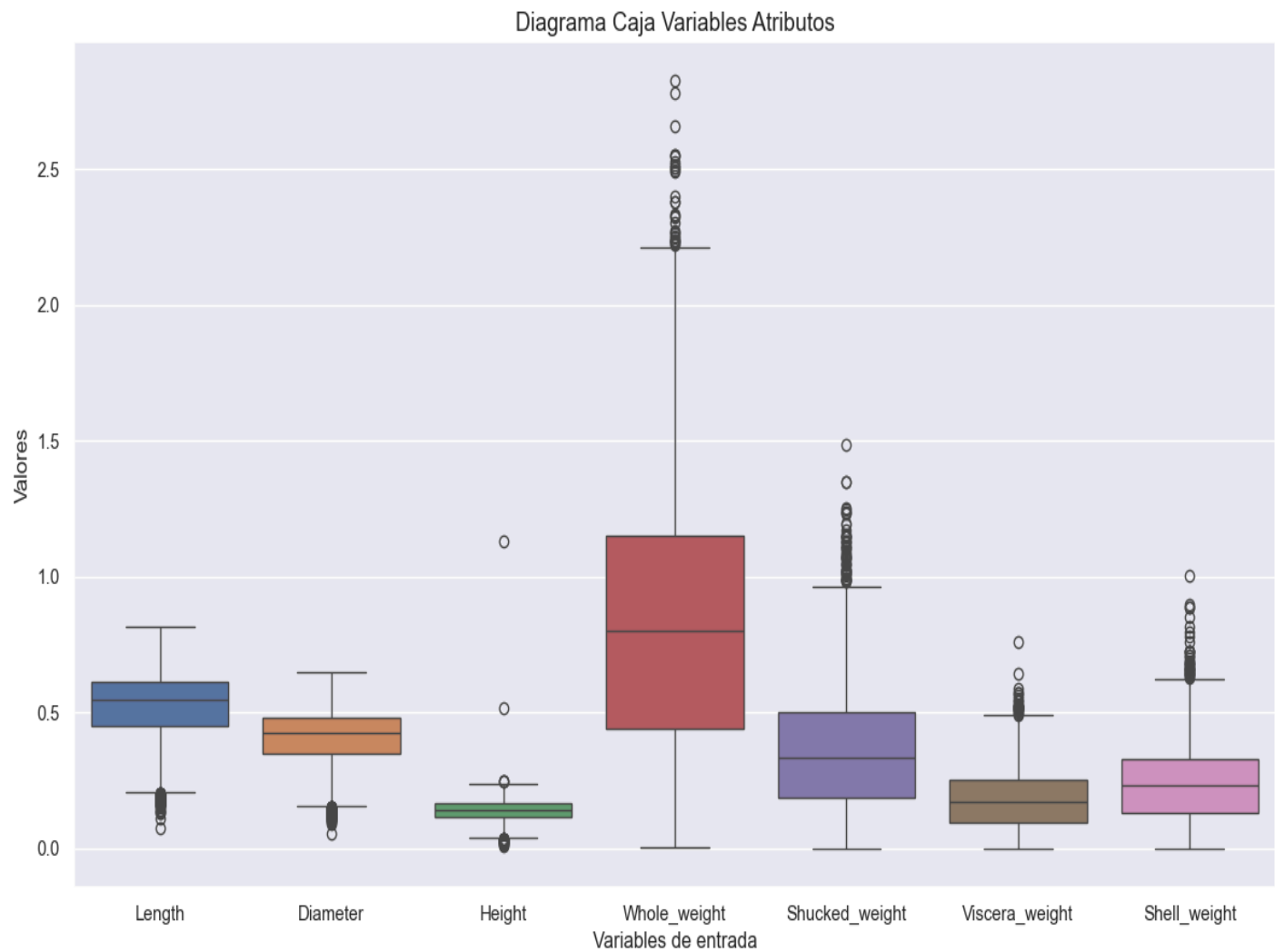


Fig 3: Diagramas Caja Variables Tipo Atributo

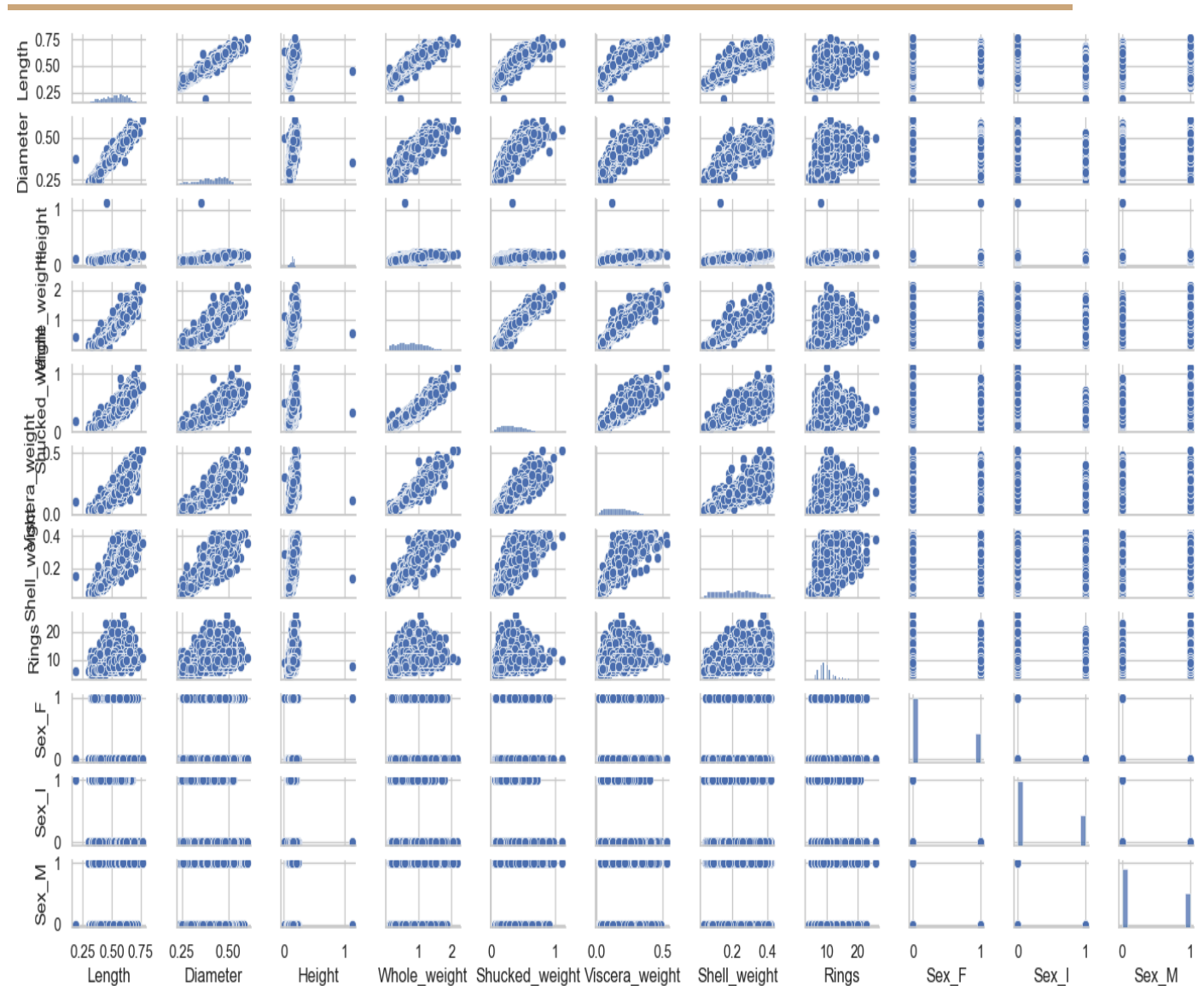


Fig 4: Correlaciones Pares Variables Tipo Atributo y Variable Objetivo

D.-IMPLEMENTACIONES DE CÓDIGO
