“SHIP MOVEMENTS MACHINE LEARNING APLICATION”

1. Introducción

El siguiente proyecto resuelve un caso práctico de Aprendizaje Automático Supervisado, empleando el lenguaje de programación *Python*. El conjunto de datos empleado, disponible en la bibliografía del documento, muestra los movimientos de 45 embarcaciones marítimas registrados en el Puerto Exterior de A Coruña (Punta Langosteira), Galicia, España, desde el año 2015 al año 2020. Cada archivo contiene variables de tipo predictoras (de entrada) y variables a predecir (de salida). Es importante destacar que las características geométricas de un barco inciden de manera directa en el comportamiento dinámico del mismo, y que en el archivo original se muestra el día y la hora exactos correspondientes a cada movimiento grabado.

1.1 Objetivo

Dado un conjunto de datos obtenidos de forma experimental, teniendo variables de entrada (meteorológicas y de atraque de cada buque), el objetivo es determinar qué variable afecta en mayor magnitud a cada movimiento, tanto traslacional como rotacional, de un buque atracado en puerto. Para ello, se han empleado 3 Algoritmos de Aprendizaje Supervisado: Regresión Lineal, K-Vecinos más cercanos o KNN y el algoritmo ensamblado ‘Random Forest’, basado en algoritmos simples de árboles de decisión. En la tabla e ilustración de a continuación se observan los grados de libertad desde un punto de vista más ilustrativo.

Tabla 1. Grados de libertad de un buque

Traslaciones	**Rotaciones**
Avance, Retroceso o Marejada (x)	Escora o Balance (x)
Ronza o Abatimiento (y)	Cabeceo (y)
Ascenso, Descenso o Arfada (z)	Virada o Guiñada (z)

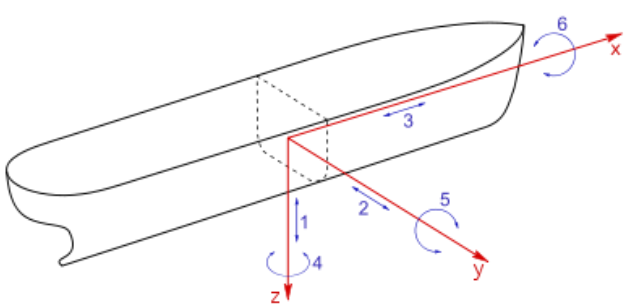


Ilustración 1. Grados de Libertad de un Buque

1.2 Abstract

The following project solves a practical case of Supervised Machine Learning, using the Python programming language. The dataset used, available in the bibliography of the document, shows the movements of 45 maritime vessels recorded in the Outer Port of A Coruña (Punta Langosteira), Galicia, Spain, from the year 2015 to the year 2020. Each file contains predictor variables (inputs) and variables to be predicted (outputs). It is important to highlight that the geometric characteristics of a ship have a direct impact on its dynamic behavior, and that the original file displays the exact day and time corresponding to each recorded movement.

The objective of this project is stated as follows:

Given a set of experimentally obtained data, with input variables (meteorological and docking variables of each vessel), the aim is to predict the values of the 6 degrees of freedom of a vessel's motion (surge, sway, heave, roll, pitch, and yaw).

1. Variables de Entrada (Input)

La infraestructura tecnológica del Puerto dispone de dispositivos de medida: en concreto, una estación meteorológica adaptada, una boya de nivel de oleaje y un mareógrafo. Todo este conjunto de datos está disponible en la URL Web del Sistema de Datos de Puertos Españoles.

- La boya de nivel de oleaje se localiza a 1,8 km del rompeolas principal del puerto. Debido a la sensibilidad del sensor de la boya al ruido, los datos se registran en periodos de 1 hora. Esta cuantificación permite calcular algunos parámetros estadísticos que reflejan el estado del mar mientras mitigan los efectos del ruido. En la *Ilustración 1* se observa la ubicación exacta del sistema.

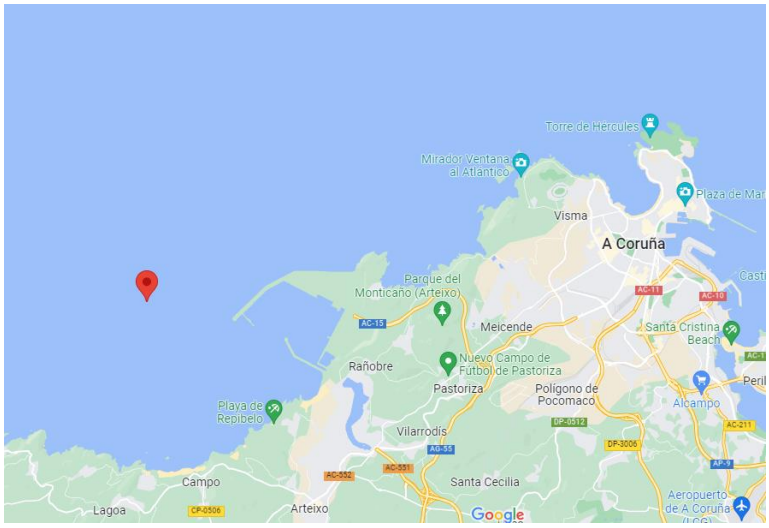


Ilustración 2. (43º 21' 00" N 8º 33' 36" W) Fuente: Google Maps

- La estación meteorológica adaptada se encuentra en el propio rompeolas principal del puerto. Los registros se obtienen en intervalos de 10 minutos. El mareógrafo se encuentra al final del propio rompeolas y proporciona registros de datos en intervalos de 1 minuto. En la tabla adjunta se muestra cada variable, su nomenclatura y su significado físico.

Tabla 2. Nomenclatura y significado de las variables de entrada

Hs (m)	Altura de la ola. Se calcula como la altura media de un intervalo de 3 olas, representando un cierto estado del mar (medido por una boya).
Tp (s)	Periodo de pico de las olas con mayor energía, extraído del análisis espectral de la energía de las olas.
Θm (deg)	Dirección media de las olas, calculada como la media de todas las direcciones individuales en una serie determinada, representada en un estado del mar determinado.
Ws (km/h)	Velocidad media del viento.
Wd (deg)	Dirección media del viento.
H0 (m)	Nivel del mar con respecto al "cero del puerto", esto es, el nivel de referencia igual a la mínima bajamar.
Hsm (m)	Altura de ola significante medida por un mareógrafo, es decir, la media del tercio más alto de las olas de una serie temporal de olas que representa un determinado estado del mar.

Ilustración 3. Rompeolas del Puerto Exterior de A Coruña. (Fuente: Autoridad Portuaria)

2. Variables de Salida (Output)

Mediante 3 sistemas de medida sincronizados entre sí ha sido posible registrar los movimientos de las embarcaciones marítimas, los cuales son tomados como variables de salida. Los sistemas de medida empleados se describen a continuación:

- Inertial Measuring Unit (IMU): Dispositivo electrónico capaz de medir y registrar el ratio angular, la gravedad específica y, en algunas ocasiones, la orientación del objeto que se está analizando. Para ello, se emplean acelerómetros, giroscópicos y magnetómetros. En el presente proyecto, tal y como se verá en los siguientes apartados, se obtienen datos registrados del balanceo, el cabeceo y la rotación de la embarcación. Por ello es esencial el empleo de este tipo de sistemas, tal y como se ilustra el modelo de medida en la ilustración de a continuación. El IMU registra el balanceo y el cabeceo de un barco.

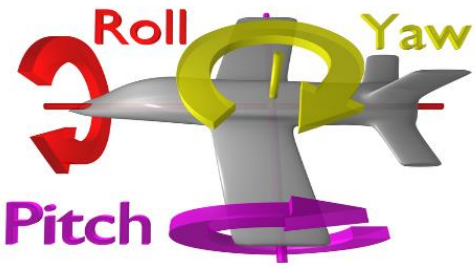


Ilustración 4. Estructura de medida de un IMU. (Fuente: Wikipedia)

- Medidores de distancia láser entre proa y popa del barco. Se emplean 2 unidades y ambas registran el balanceo lineal y la guiñada de un barco.
- Cámaras capaces de rastrear el movimiento del barco mediante técnicas de visión por ordenador. Ambas unidades registran la marejada y la oscilación de un barco.

Una vez descritos los equipos de medida, se describen las variables de salida, las cuales son el resultado de todas las fuerzas actuantes sobre la embarcación atracada en puerto.

Tabla 3. Variables de salida (Movimiento y rotación de un barco atracado)

Marejada (m) (Surge)	Movimiento longitudinal lineal (proa-popa).
Ronza o Abatimiento (m) (Sway)	Movimiento lateral lineal (babor-estribor).
Arfada (m) (Heave)	Movimiento vertical lineal.
Escora o Balance (deg) (Roll)	Rotación basculante del buque alrededor de su eje longitudinal (proa-popa).
Cabeceo (deg) (Pitch)	Rotación ascendente/descendente del buque alrededor de su eje lateral (babor-estribor).
Guiñada (deg) (Yaw)	Rotación del buque en torno a su eje vertical.

Por último, se presenta el significado de las variables empleadas en el proyecto.

Tabla 4. Variables del Dataset

Variable	**Nombre en el conjunto de datos**	**Unidades o formato**
Ship name	ship	ship#
Ship type	type	"General Cargo" or "Bulk carrier"
Date of stay	time	MM/DD/YYYY HH:mm
Length	length	m
Breath	breath	m
DWT	dwt	tonnes
Berthing zone	zone	[1, 12]
Hs	h_s	m

Tp	t_p	s
θm	dir	deg
Ws	wind_speed	km/h
Wd	wind_dir	deg
H0	h_0	m
Hsm	h_sm	m
Average movement	mov_avg	Surge, sway, and heave: m
Roll, pitch, and yaw: deg	mov_sig	Surge, sway, and heave: m Roll, pitch, and yaw: deg

3. Resultados

La regresión, a diferencia de la clasificación, es un método de aprendizaje supervisado en el que se entrena a un algoritmo para predecir una salida a partir de un rango continuo de valores posibles. Es decir, un algoritmo necesita identificar una relación funcional entre los parámetros de entrada y de salida. En este apartado, se muestra el código desarrollado en Python para cada grado de libertad del buque, los coeficientes y valores propios de cada algoritmo obtenidos (Fase de Entrenamiento / Train), y, el empleo de estos valores para la resolución del modelo (Fase de Testeo / Test), empleando, tal y como se comentó en la introducción del proyecto, los algoritmos de Regresión Lineal, K-Vecinos Cercanos y de Bosque Aleatorio.

3.1 Algoritmo de Regresión Lineal (Linear Regression)

Se trata de uno de los algoritmos más sencillos de aprendizaje supervisado. Este procedimiento se documentó por primera vez al publicarse el método de mínimos cuadrados por *Legendre* en el año 1805. La regresión lineal parte de unas variables predictoras numéricas (x1, x2…xN) y supone que existe una relación lineal aproximada entre dichas variables y la variable a predecir, Y. A continuación, se muestra la expresión de dicho algoritmo.

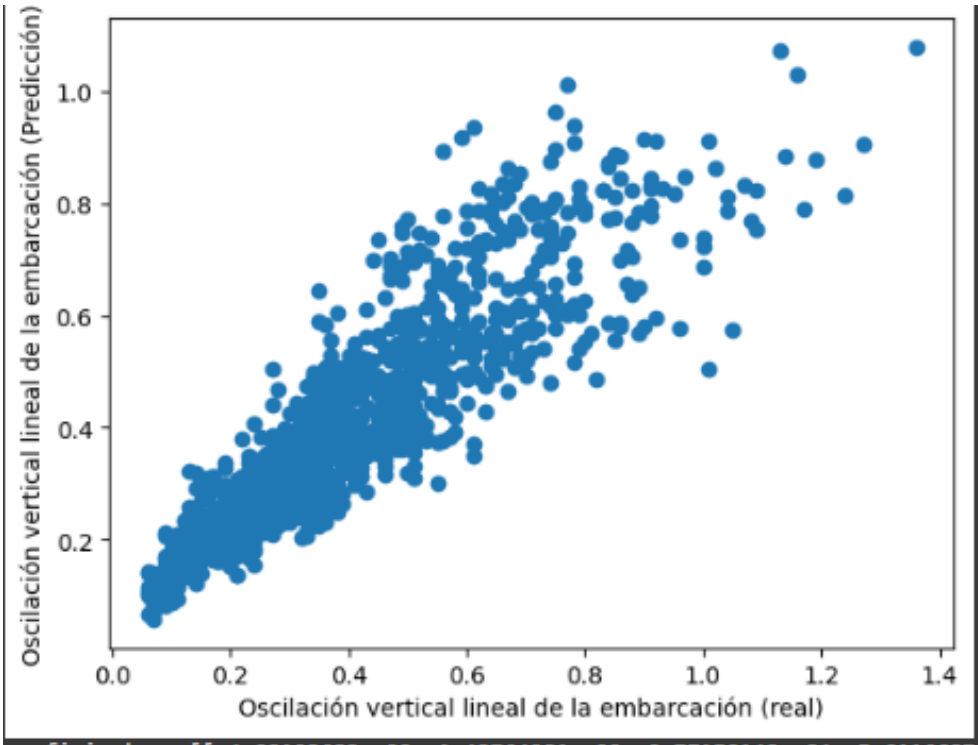
Ecuación 1. Expresión del Algoritmo de Regresión Lineal

$$Y = \beta 0 + X1 * \beta 1 + X2 * \beta 2 + \cdots + \beta N * XN$$

El código desarrollado en Python para cada algoritmo se observa a continuación, mediante celdas de Jupyter Notebook, por lo que, cada celda corresponde al grado de libertad señalado dentro de la misma, observando los resultados del mismo a continuación.

3.1.1 Arfada (Traslación eje Z)

```
# MODELO DE APRENDIZAJE AUTOMÁTICO SUPERVISADO / ALGORITMO DE
REGRESIÓN LINEAL
# 1) Arfada eje Z (heave)
# Se accede a las carpetas de Google Drive y se habilta la entrada
de código
from google.colab import drive
drive.mount('/content/drive')
from ast import Mult
# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

# Se importan las librerías para el acceso al dataset
import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Se accede al archivo CSV
heave_train = '/content/drive/MyDrive/port_coruna/heave_train.csv'
df = pd.read_csv(heave_train)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]

# Se crea la instancia (objeto específico del modelo) y se ajusta
el modelo a los datos
model = LinearRegression()
model.fit(x, y)
prediction_linear = model.predict(x)
r2 = r2_score(y, prediction_linear)

# Se realizan predicciones del modelo y se muestra

plt.scatter(y, prediction_linear)
plt.xlabel('Oscilación vertical lineal de la embarcación (real)')
plt.ylabel('Oscilación vertical lineal de la embarcación
(Predicción)')
plt.show()
print("coeficientes:", model.coef_)
print("Termino independiente:", model.intercept_)
print(r2)
```




```
coeficientes: [[ 4.09123682e-02  1.43764281e-02  2.77152149e-04 -5.61166874e-04
  -8.03855923e-05  8.98990842e-03  1.46471143e-01 -3.12141536e-03
   1.67026729e-02]
 [ 9.01793506e-02  3.30114581e-02  6.50106919e-04 -1.81963777e-03
  -8.16721269e-05  2.19805917e-02  3.19508469e-01 -6.30344473e-03
   3.30593005e-02]
 [ 6.07041464e-02  2.16879355e-02  4.11190343e-04 -9.12496613e-04
  -1.11275559e-04  1.28851406e-02  2.27176813e-01 -4.47947054e-03
   2.40346230e-02]]
Termino independiente: [-0.00721434 -0.04049989 -0.02200456]
0.6798818834801844
```

Ilustración 5. Arfada de los buques (Regresión Lineal)

3.1.2 Guiñada (Rotación Eje Z)

```
# MODELO DE APRENDIZAJE AUTOMÁTICO SUPERVISADO / ALGORITMO DE
REGRESIÓN LINEAL
# 2) Guiñada, rotación del buque (eje Z)
# Se accede a las carpetas de Google Drive y se habilta la entrada
de código
from google.colab import drive
drive.mount('/content/drive')
from ast import Mult
# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

# Se importan las librerías para el acceso al dataset
import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

```
from sklearn.metrics import mean_squared_error, r2_score

# Se accede al archivo CSV
yaw_train = '/content/drive/MyDrive/port_coruna/yaw_train.csv'
df = pd.read_csv(yaw_train)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]

# Se crea la instancia (objeto específico del modelo) y se ajusta
el modelo a los datos
model = LinearRegression()
model.fit(x, y)
prediction_linear = model.predict(x)
r2 = r2_score(y, prediction_linear)

# Se realizan predicciones del modelo y se muestra

plt.scatter(y, prediction_linear)
plt.xlabel('Guiñada de la embarcacion (real)')
plt.ylabel('Guiñada de la embarcación (Predicción)')
plt.show()
print("coeficientes:", model.coef_)
print("Termino independiente:", model.intercept_)
print(r2)
```

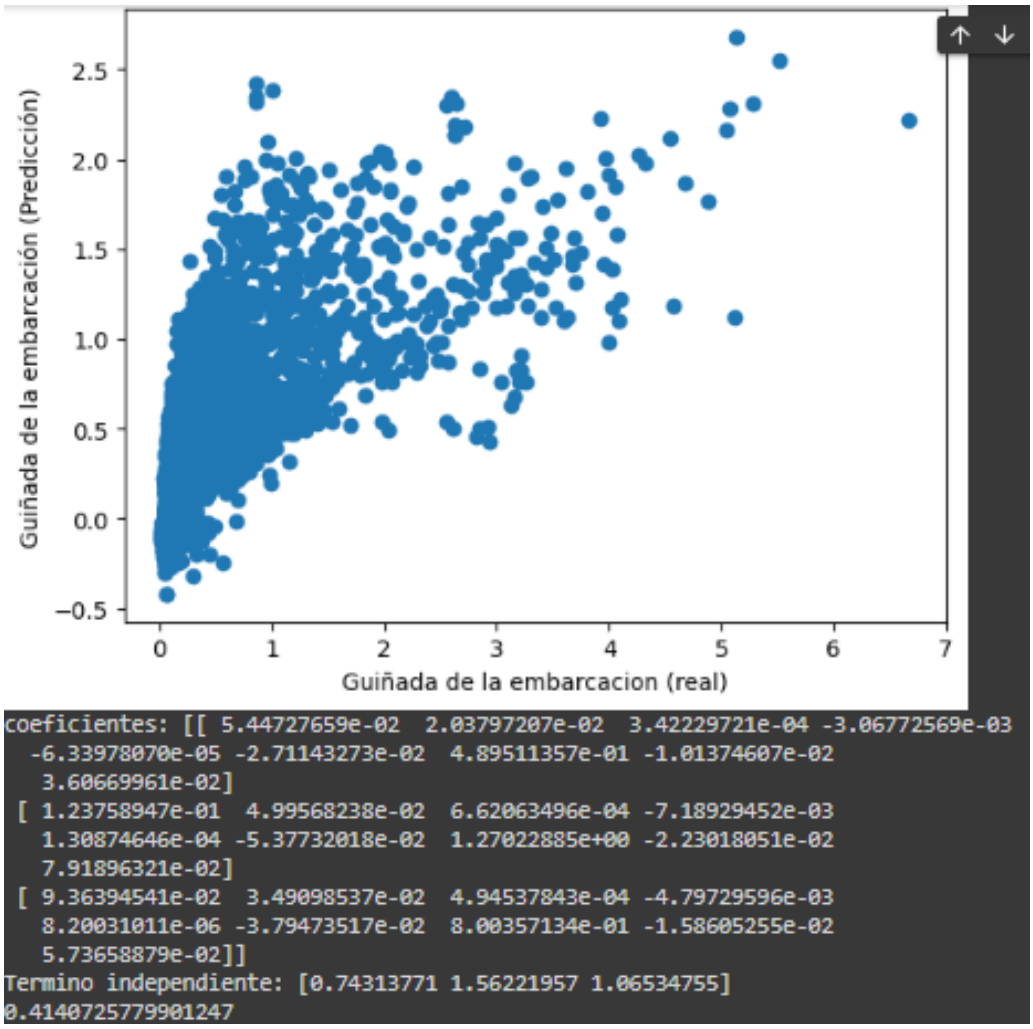
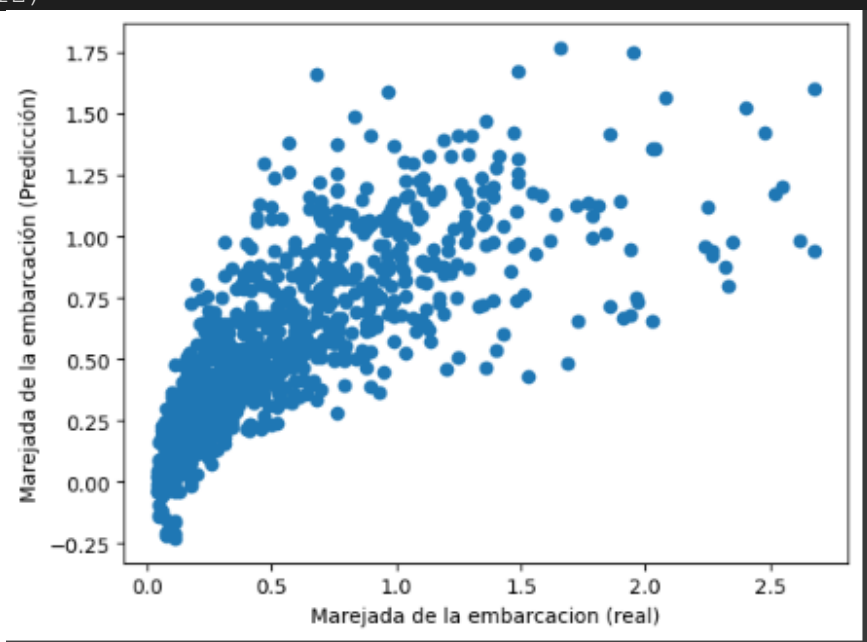


Ilustración 6. Guiñada de los buques (Regresión Lineal)

3.1.3 Marejada (Traslación Eje X)

```
# MODELO DE APRENDIZAJE AUTOMÁTICO SUPERVISADO / ALGORITMO DE
REGRESIÓN LINEAL
# 3) Marejada, movimiento longitudinal lineal (eje x)
# Se accede a las carpetas de Google Drive y se habilta la entrada
de código
from google.colab import drive
drive.mount('/content/drive')
from ast import Mult
# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

# Se importan las librerías para el acceso al dataset
import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
# Se accede al archivo CSV
surge_train = '/content/drive/MyDrive/port_coruna/surge_train.csv'
df = pd.read_csv(surge_train)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]

# Se crea la instancia (objeto específico del modelo) y se ajusta
el modelo a los datos
model = LinearRegression()
model.fit(x, y)
prediction_linear = model.predict(x)
r2 = r2_score(y, prediction_linear)

# Se realizan predicciones del modelo y se muestra

plt.scatter(y, prediction_linear)
plt.xlabel('Marejada de la embarcacion (real)')
plt.ylabel('Marejada de la embarcación (Predicción)')
plt.show()
print("coeficientes:", model.coef_)
print("Termino independiente:", model.intercept_)
print(r2)
```

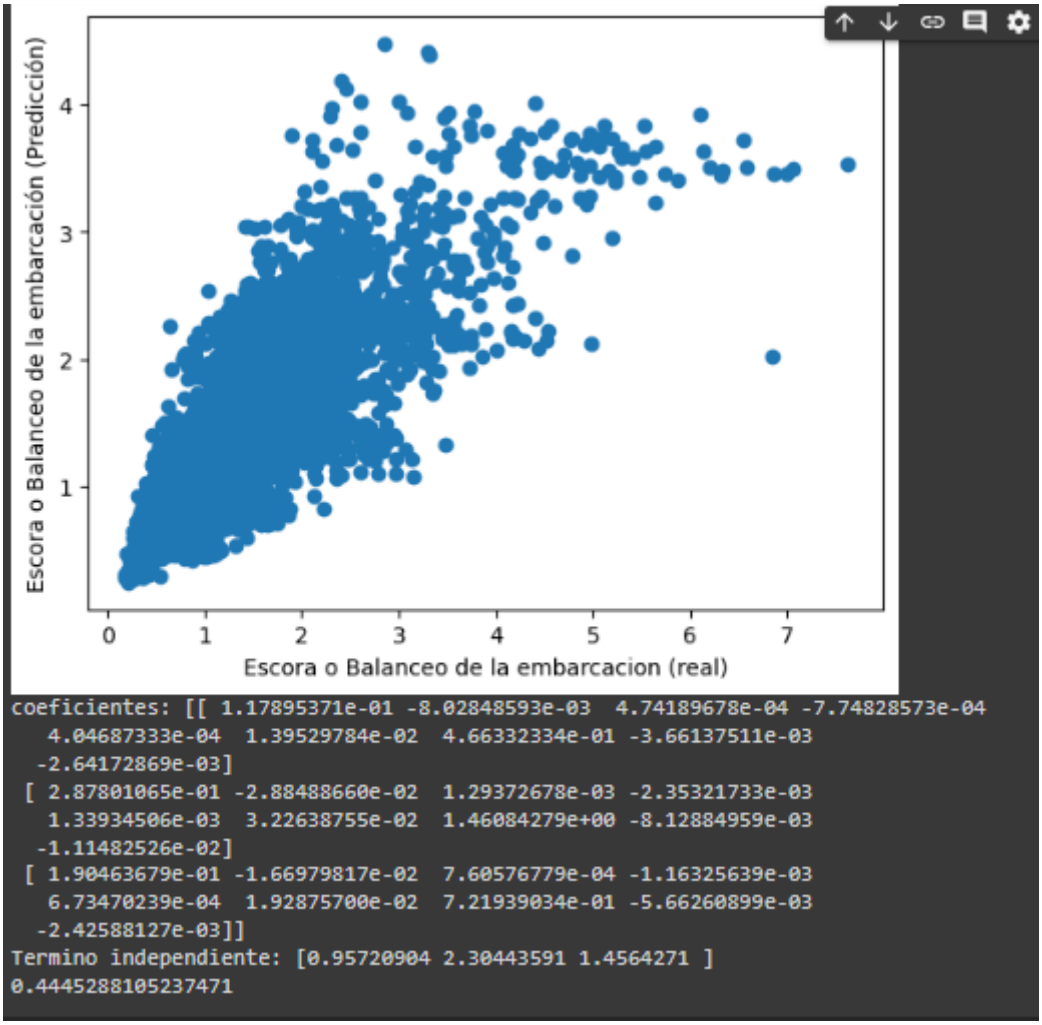



```
coeficientes: [[ 6.88806861e-02  6.91653986e-02  5.40879785e-04  3.16239816e-03
   1.23979824e-04 -1.18116736e-02 -3.60202910e-01  1.25699513e-03
  -2.20426684e-02]
 [ 1.60523335e-01  1.28928580e-01  1.33362046e-03  3.41204477e-03
   3.13122518e-04  1.14256282e-02 -7.24411666e-01 -2.78179269e-04
  -2.85876890e-02]
 [ 1.21016589e-01  1.05418859e-01  8.77922324e-04  3.71138642e-03
   1.92046930e-04 -1.07190150e-02 -6.31186046e-01  9.50088352e-04
  -2.77290809e-02]]
Termino independiente: [-0.43606623 -0.77485813 -0.66532339]
0.5239604124635399
```

Ilustración 7. Marejada de los buques (Regresión Lineal)

3.1.4 Escora o Balanceo (Rotación Eje X)

```
# MODELO DE APRENDIZAJE AUTOMÁTICO SUPERVISADO / ALGORITMO DE
REGRESIÓN LINEAL
# 4) Balanceo basculante del buque (eje x)
# Se accede a las carpetas de Google Drive y se habilta la entrada
de código
from google.colab import drive
drive.mount('/content/drive')
from ast import Mult
# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

# Se importan las librerías para el acceso al dataset
import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Se accede al archivo CSV
roll_train = '/content/drive/MyDrive/port_coruna/roll_train.csv'
df = pd.read_csv(roll_train)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]

# Se crea la instancia (objeto específico del modelo) y se ajusta
el modelo a los datos
model = LinearRegression()
model.fit(x, y)
prediction_linear = model.predict(x)
r2 = r2_score(y, prediction_linear)

# Se realizan predicciones del modelo y se muestra

plt.scatter(y, prediction_linear)
plt.xlabel(Escora o Balanceo de la embarcacion (real)')
plt.ylabel(Escora o Balanceo de la embarcación (Predicción)')
plt.show()
print("coeficientes:", model.coef_)
print("Termino independiente:", model.intercept_)
print(r2)
```



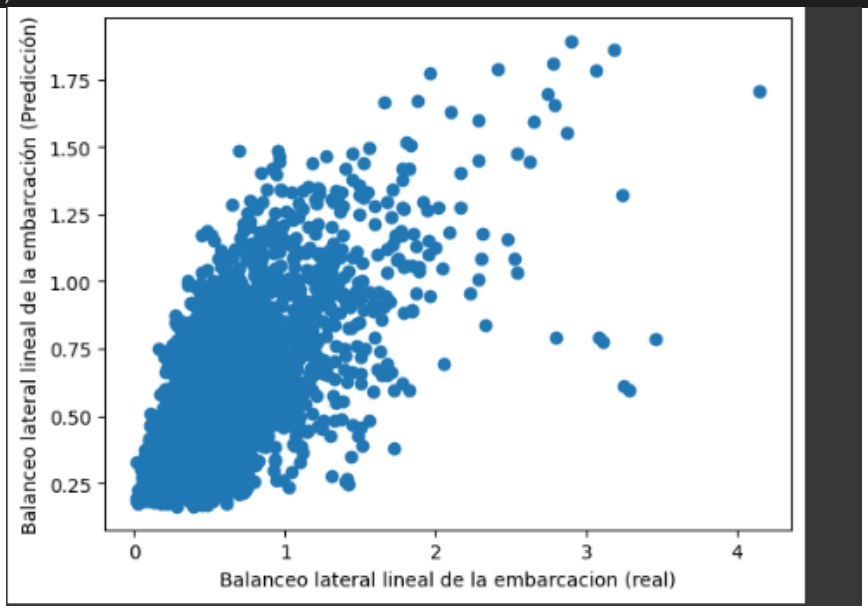

```
coeficientes: [[ 1.17895371e-01 -8.02848593e-03  4.74189678e-04 -7.74828573e-04
   4.04687333e-04  1.39529784e-02  4.66332334e-01 -3.66137511e-03
  -2.64172869e-03]
 [ 2.87801065e-01 -2.88488660e-02  1.29372678e-03 -2.35321733e-03
   1.33934506e-03  3.22638755e-02  1.46084279e+00 -8.12884959e-03
  -1.11482526e-02]
 [ 1.90463679e-01 -1.66979817e-02  7.60576779e-04 -1.16325639e-03
   6.73470239e-04  1.92875700e-02  7.21939034e-01 -5.66260899e-03
  -2.42588127e-03]]
Termino independiente: [0.95720904 2.30443591 1.4564271 ]
0.4445288105237471
```

Ilustración 8. Balanceo de los buques (Regresión Lineal)

3.1.5 Abatimiento / Ronza (Traslación Eje Y)

```
# MODELO DE APRENDIZAJE AUTOMÁTICO SUPERVISADO / ALGORITMO DE
REGRESIÓN LINEAL
# 5) Abatimiento, movimiento lateral lineal (eje y)
# Se accede a las carpetas de Google Drive y se habilta la entrada
de código
from google.colab import drive
drive.mount('/content/drive')
from ast import Mult
# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

# Se importan las librerías para el acceso al dataset
import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
# Se accede al archivo CSV
sway_train = '/content/drive/MyDrive/port_coruna/sway_train.csv'
df = pd.read_csv(sway_train)
df.dropna(inplace= True)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]

# Se crea la instancia (objeto específico del modelo) y se ajusta
el modelo a los datos
model = LinearRegression()
model.fit(x, y)
prediction_linear = model.predict(x)
r2 = r2_score(y, prediction_linear)

# Se realizan predicciones del modelo y se muestra

plt.scatter(y, prediction_linear)
plt.xlabel('Balanceo lateral lineal de la embarcacion (real)')
plt.ylabel('Balanceo lateral lineal de la embarcación
(Predicción)')
plt.show()
print("coeficientes:", model.coef_)
print("Termino independiente:", model.intercept_)
print(r2)
```




```
coeficientes: [[ 6.58398966e-02 -1.03463028e-03 -8.11455380e-04  3.41364147e-05
  -1.13686949e-04 -3.27235559e-02  3.36536069e-02  1.21489842e-03
  -6.83427771e-03]
 [ 1.72165953e-01  1.80494999e-02 -7.05696087e-04 -1.61626709e-03
   8.43664312e-05 -5.10027293e-02  3.35883832e-01 -5.40906128e-03
   1.92363652e-02]
 [ 1.09187723e-01  4.82957469e-03 -8.14054118e-04 -5.85081176e-04
  -5.92598265e-05 -3.96458310e-02  1.47551451e-01 -9.08607775e-04
   1.23001028e-03]]
Termino independiente: [0.4561162  0.77036754 0.56236229]
0.2856688038177042
```

Ilustración 9. Abatimiento de los buques (Regresión Lineal)

3.1.6 Cabeceo (Rotación Eje Y)

```
# MODELO DE APRENDIZAJE AUTOMÁTICO SUPERVISADO / ALGORITMO DE
REGRESIÓN LINEAL
# 6) Cebeceo, rotación ascendente / descendente del buque (eje y)
# Se accede a las carpetas de Google Drive y se habilta la entrada
de código
from google.colab import drive
drive.mount('/content/drive')
from ast import Mult
# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

# Se importan las librerías para el acceso al dataset
import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Se accede al archivo CSV
pitch_train = '/content/drive/MyDrive/port_coruna/pitch_train.csv'
df = pd.read_csv(pitch_train)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]

# Se crea la instancia (objeto específico del modelo) y se ajusta
el modelo a los datos
model = LinearRegression()
model.fit(x, y)
prediction_linear = model.predict(x)
r2 = r2_score(y, prediction_linear)

# Se realizan predicciones del modelo y se muestra

plt.scatter(y, prediction_linear)
plt.xlabel('Cabeceo ascendente/descendente eje lateral buque
(real)')
plt.ylabel('Cabeceo ascendente/descendente eje lateral buque
(Predicción)')
plt.show()
print("coeficientes:", model.coef_)
print("Termino independiente:", model.intercept_)
print(r2)
```

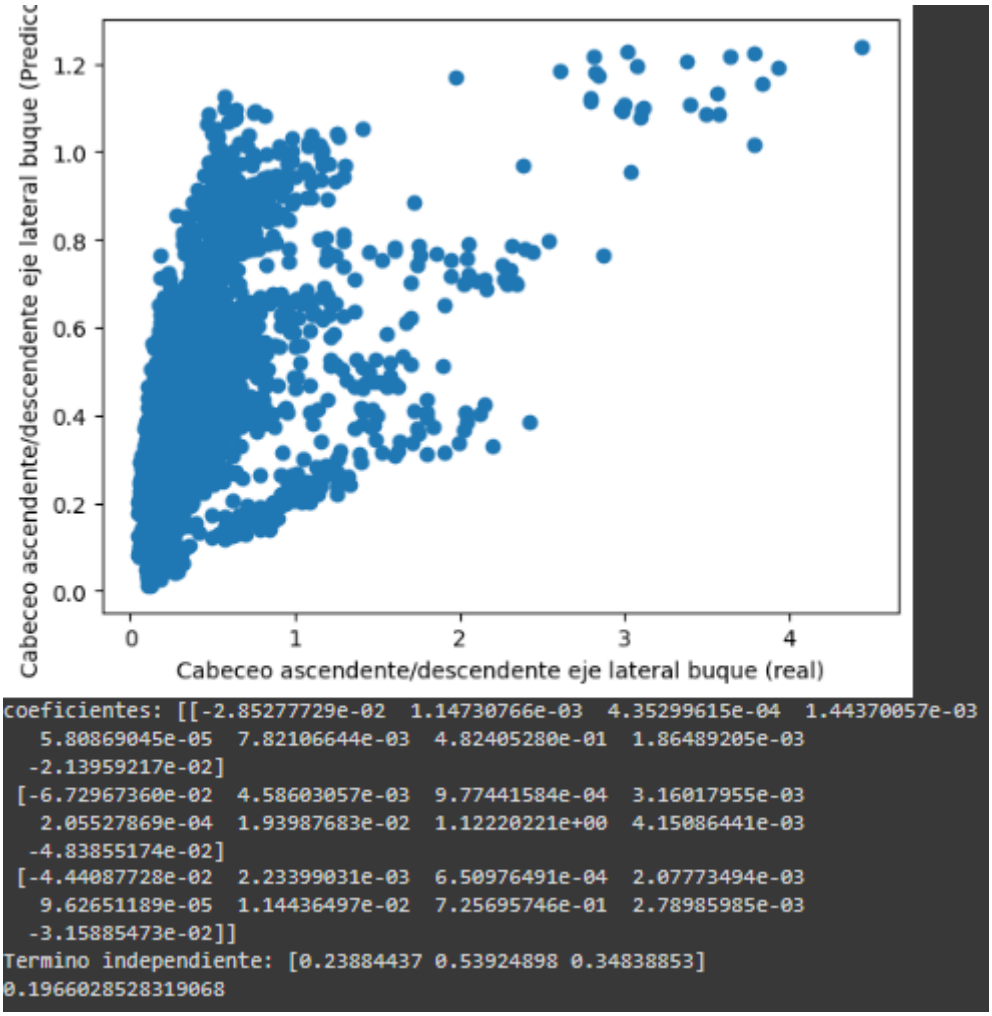


Ilustración 10. Cabeceo de los buques (Regresión Lineal)

Como se observa, los resultados obtenidos mediante el Algoritmo de Regresión Lineal no son satisfactorios, ya que, el coeficiente R^2 más elevado se corresponde a la Arfada de los buques, con un valor de 0,67. Los valores de R cuadrado deben aproximarse lo máximo posible a 1, y como este valor se aleja con un porcentaje de error elevado, el empleo de este algoritmo se descarta.

3.2 Algoritmo K-Vecinos más cercanos (KNN)

El algoritmo de k-vecinos más cercanos (K-nearesr neighbors), conocido como KNN, es un modelo empleado para problemas de regresión y de clasificación. Para este proyecto, se emplea regresión, por lo que los predictores X que están cerca deberían dar predicciones que también son cercanas. Este método se basa en encontrar K muestras de entrenamiento más cercanas en distancia a la muestra de la que se pretende predecir la variable dependiente. La expresión empleada se muestra a continuación.

Ecuación 2. Algoritmo de KNN para casos de Regresión

$$\hat{y} = Y_{kNN}(x) = \frac{1}{k} \sum_{x^p \in N_k(x)} y^p$$

Por tanto, la predicción sería la media de los valores de las variables dependientes de sus K-Vecinos, es un modelo no lineal, lo que indica que se pueden captar relaciones no lineales entre variables X e Y.

3.2.1 Arfada (Traslación Eje Z)

```
# ALGORITMO K-VECINOS MÁS CERCANOS (KNN)
# Se accede a las carpetas de Google drive
from google.colab import drive
drive.mount('/content/drive')

# Se habilita entrada de código
from ast import Mult

# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Se accede al archivo CSV
heave_train = '/content/drive/MyDrive/port_coruna/heave_train.csv'
df = pd.read_csv(heave_train)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]

model = KNeighborsRegressor()
model.fit(x, y)
prediction_knn = model.predict(x)
r2 = r2_score(y, prediction_knn)
plt.scatter(y, prediction_knn)
plt.xlabel('Oscilación vertical lineal de la embarcación (real)')
plt.ylabel('Oscilación vertical lineal de la embarcación
(Predicción)')
plt.show()
print("Número de vecinos obtenido:", model.n_neighbors)
print(r2)
```

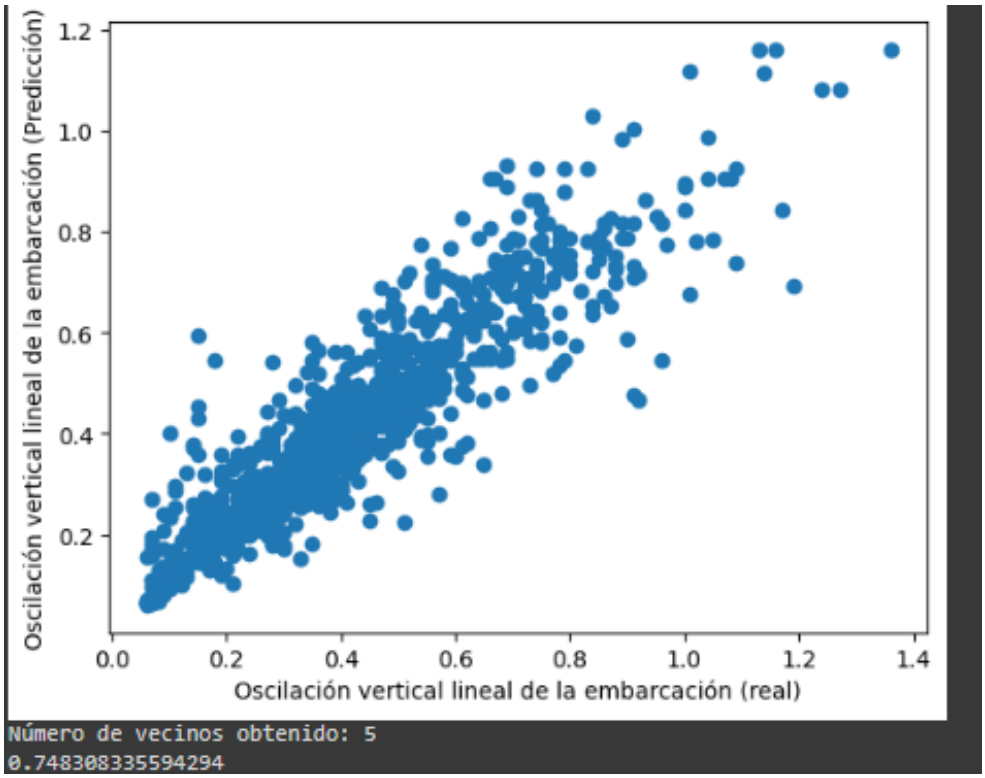
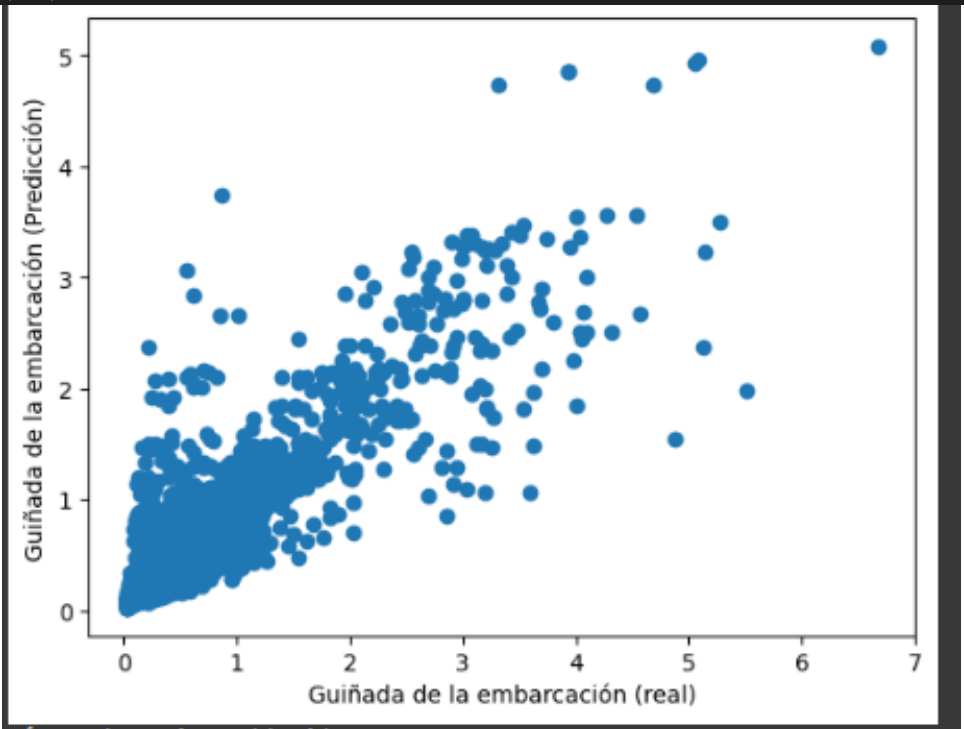


Ilustración 11. Arfada de los buques (KNN)

3.2.2 Guiñada (Rotación Eje Z)

```
# Se accede a las carpetas de Google drive
from google.colab import drive
drive.mount('/content/drive')

# Se habilita entrada de código
from ast import Mult

# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline


import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Se accede al archivo CSV
yaw_train = '/content/drive/MyDrive/port_coruna/yaw_train.csv'
df = pd.read_csv(yaw_train)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]
```

```
model = KNeighborsRegressor()
model.fit(x, y)
prediction_knn = model.predict(x)
r2 = r2_score(y, prediction_knn)
plt.scatter(y, prediction_knn)
plt.xlabel('Guiñada de la embarcación (real)')
plt.ylabel('Guiñada de la embarcación (Predicción)')
plt.show()
print("Número de vecinos obtenido:", model.n_neighbors)
print(r2)
```




```
Número de vecinos obtenido: 5
0.7596783449090644
```

Ilustración 12. Guiñada de los buques (KNN)

3.2.3 Marejada (Traslación Eje X)

```
# Se accede a las carpetas de Google drive
from google.colab import drive
drive.mount('/content/drive')

# Se habilita entrada de código
from ast import Mult

# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline


import pandas as pd
import numpy as np
```

```
import csv
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Se accede al archivo CSV
surge_train = '/content/drive/MyDrive/port_coruna/surge_train.csv'
df = pd.read_csv(surge_train)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]

model = KNeighborsRegressor()
model.fit(x, y)
prediction_knn = model.predict(x)
r2 = r2_score(y, prediction_knn)
plt.scatter(y, prediction_knn)
plt.xlabel('Marejada de la embarcación (real)')
plt.ylabel('Marejada de la embarcación (Predicción)')
plt.show()
print("Número de vecinos obtenido:", model.n_neighbors)
print(r2)
```

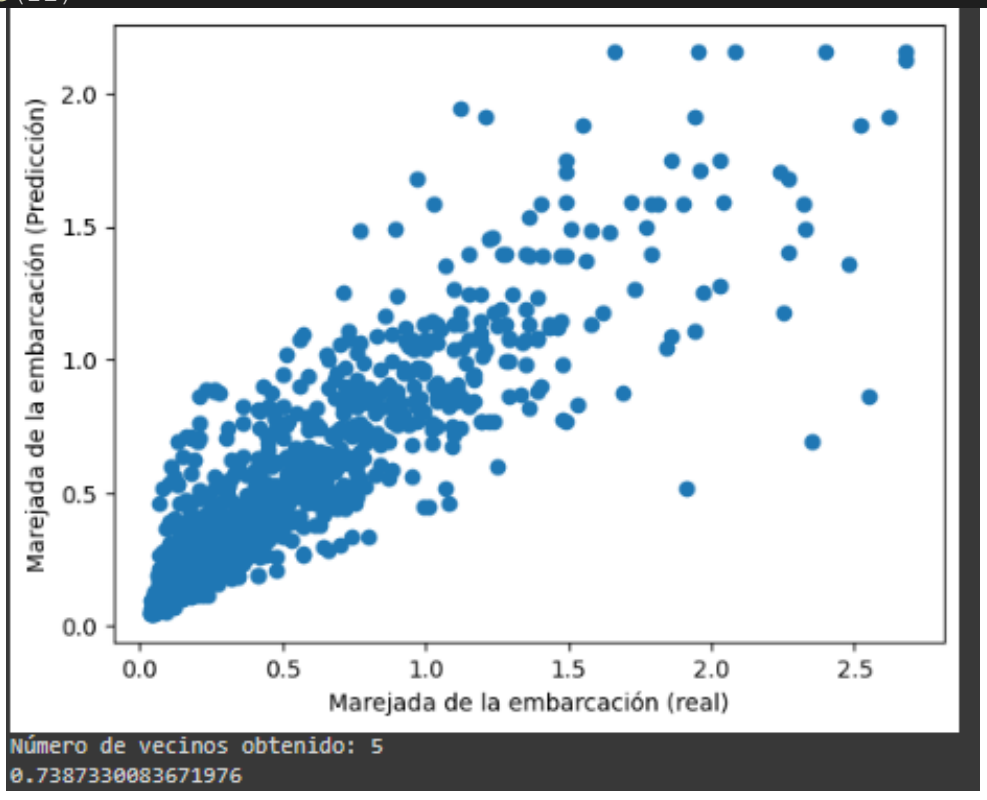


Ilustración 13. Marejada de los buques (KNN)

3.2.4 Balanceo o Escora (Rotación Eje X)

```
# Se accede a las carpetas de Google drive
```

```
from google.colab import drive
drive.mount('/content/drive')

# Se habilita entrada de código
from ast import Mult

# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline


import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Se accede al archivo CSV
roll_train = '/content/drive/MyDrive/port_coruna/roll_train.csv'
df = pd.read_csv(roll_train)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]

model = KNeighborsRegressor()
model.fit(x, y)
prediction_knn = model.predict(x)
r2 = r2_score(y, prediction_knn)
plt.scatter(y, prediction_knn)
plt.xlabel('Balanceo o Escora de la embarcación (real)')
plt.ylabel('Balanceo o Escora de la embarcación (Predicción)')
plt.show()
print("Número de vecinos obtenido:", model.n_neighbors)
print(r2)
```

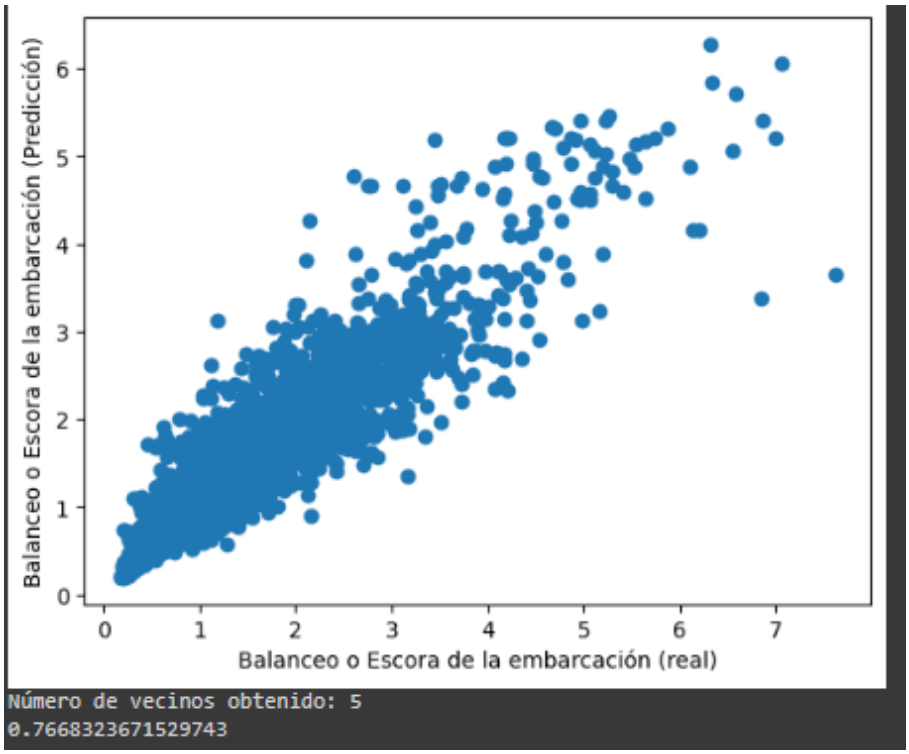


Ilustración 14. Balanceo de los buques (KNN)

3.2.5 Abatimiento / Ronza (Traslación Eje Y)

```
# Se accede a las carpetas de Google drive
from google.colab import drive
drive.mount('/content/drive')

# Se habilita entrada de código
from ast import Mult

# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Se accede al archivo CSV
sway_train = '/content/drive/MyDrive/port_coruna/sway_train.csv'
df = pd.read_csv(sway_train)
df.dropna(inplace = True)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]
```

```
model = KNeighborsRegressor()
model.fit(x, y)
prediction_knn = model.predict(x)
r2 = r2_score(y, prediction_knn)
plt.scatter(y, prediction_knn)
plt.xlabel('Balanceo lateral lineal de la embarcación (real)')
plt.ylabel('Balanceo lateral lineal de la embarcación
(Predicción)')
plt.show()
print("Número de vecinos obtenido:", model.n_neighbors)
print(r2)
```

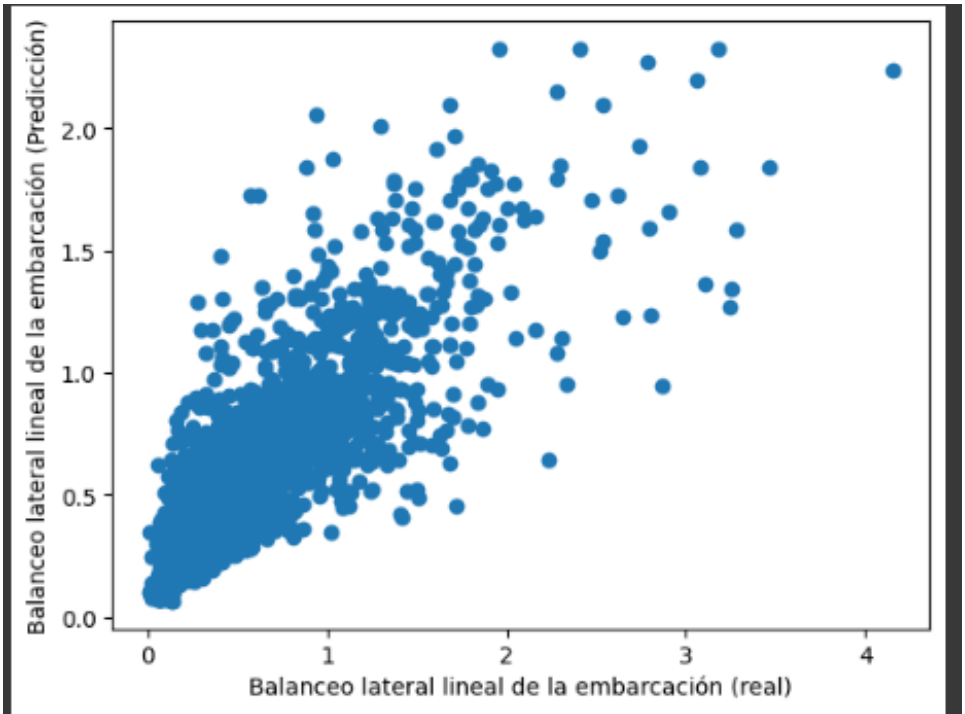


Ilustración 15. Abatimiento de los buques (KNN)

3.2.6 Cabeceo (Rotación Eje Y)

```
# Se accede a las carpetas de Google drive
from google.colab import drive
drive.mount('/content/drive')

# Se habilita entrada de código
from ast import Mult

# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

import pandas as pd
import numpy as np
import csv
```

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Se accede al archivo CSV
pitch_train = '/content/drive/MyDrive/port_coruna/pitch_train.csv'
df = pd.read_csv(pitch_train)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]

model = KNeighborsRegressor()
model.fit(x, y)
prediction_knn = model.predict(x)
r2 = r2_score(y, prediction_knn)
plt.scatter(y, prediction_knn)
plt.xlabel('Cabeceo ascendente/descendente eje lateral buque
(real)')
plt.ylabel('Cabeceo ascendente/descendente eje lateral buque
(Predicción)')
plt.show()
print("Número de vecinos obtenido:", model.n_neighbors)
print(r2)
```

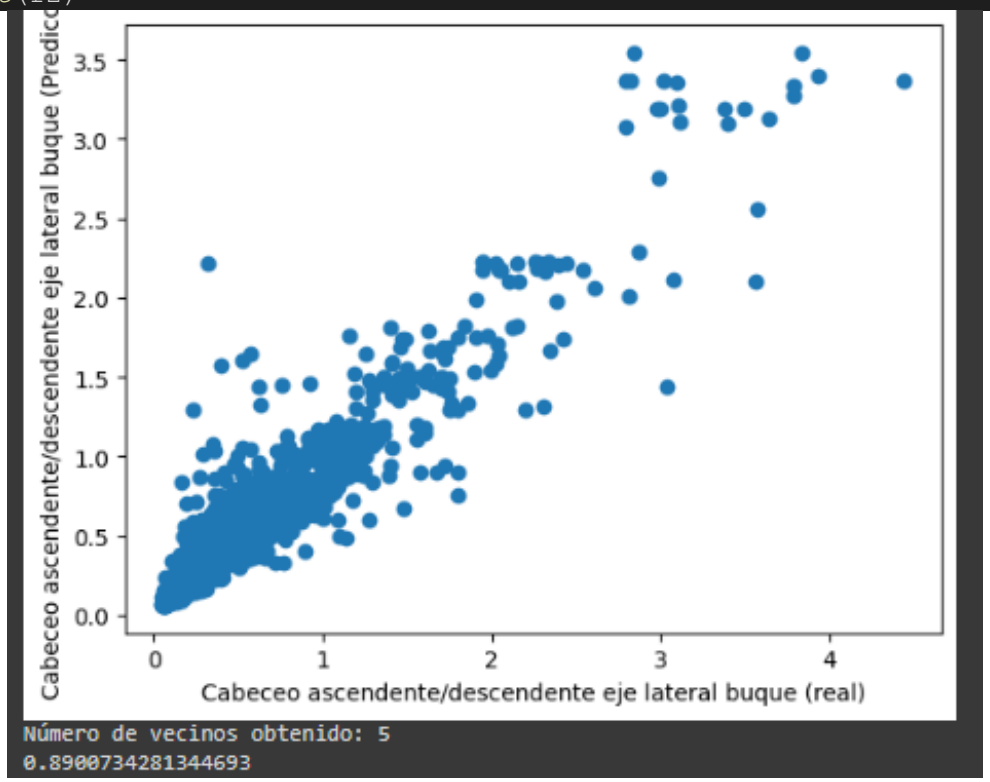


Ilustración 16. Cabeceo de los buques (KNN)

Como se observa, empleando el algoritmo KNN, el R Cuadrado mayor obtenido es igual a 0,89. Esto indica que el conjunto de variables de entrada sigue una regresión no lineal, ya que los

valores de R2, en general, son superiores en KNN al compararlos con Regresión Lineal. Pero ya que se aleja de la unidad, se descarta como modelo empleado para un aprendizaje supervisado.

3.3 Algoritmo de Random Forest

El algoritmo Random Forest es un algoritmo ensamblado, es decir, un algoritmo completo conformado por algoritmos más simples. Es un algoritmo de *bagging*, es decir, sistema donde varios algoritmos simples se emplean en paralelo. Los algoritmos base o simples son los árboles de decisión.

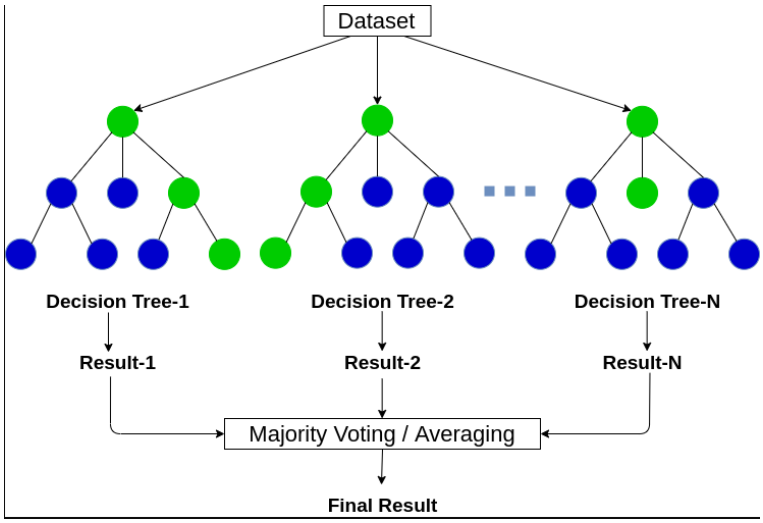


Ilustración 17. Definición de Random Forest. Fuente:

El algoritmo de Árbol de decisión es un modelo de aprendizaje jerárquico que divide el espacio empleando reglas de decisión. Es válido para modelos de regresión como para modelos de clasificación. Básicamente, el objetivo es predecir el valor de una variable dependiente aprendiendo reglas de decisión simples a partir de variables independientes.

Los árboles de decisión los forman nodos y hojas. Existen dos conceptos mediante los cuales se crean los árboles:

- Criterio de división: En cada nodo se emplea una regla para decidir que muestras caen en un lado u en otro del nodo.
- Criterio de parada: Limita el número de reglas aplicadas en cada árbol.

3.3.1 Arfada (Traslación eje Z)

```
# ALGORITMO DE RANDOM FOREST
```

```
# Se accede a las carpetas de Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Se habilita entrada de código
from ast import Mult

# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Se accede al archivo CSV
heave_train = '/content/drive/MyDrive/port_coruna/heave_train.csv'
df = pd.read_csv(heave_train)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg', 'mov_max', 'mov_sig']]

model = RandomForestRegressor()
model.fit(x, y)
prediction_rf = model.predict(x)
r2 = r2_score(y, prediction_rf)
plt.scatter(y, prediction_rf)
plt.xlabel('Oscilación vertical lineal de la embarcación (real)')
plt.ylabel('Oscilación vertical lineal de la embarcación
(Predicción)')
plt.show()
print("Número de árboles en el bosque:", model.n_estimators)
print(r2)
```

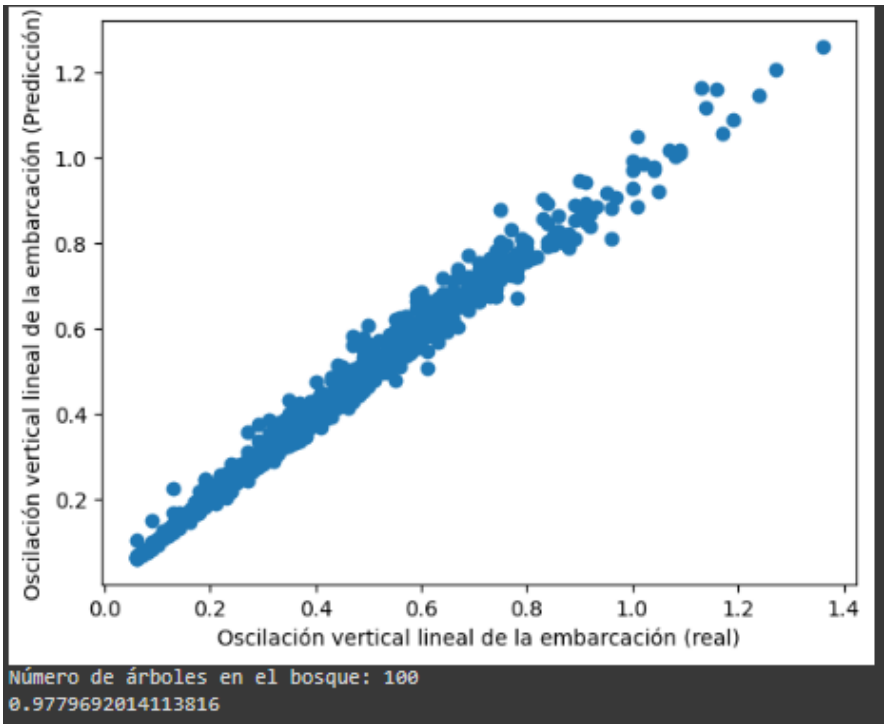
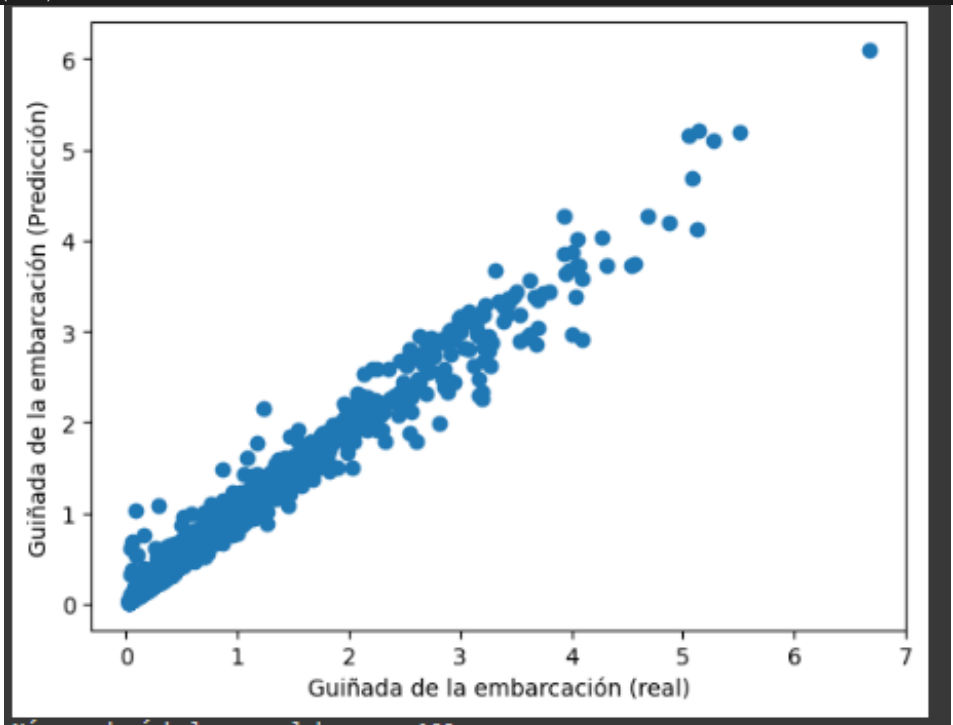


Ilustración 18. Arfada de los buques (Random Forest)

3.3.2 Guiñada (Rotación Eje Z)

```
# Se accede a las carpetas de Google Drive y se habilta la entrada
de código
from google.colab import drive
drive.mount('/content/drive')
from ast import Mult
# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

# Se importan las librerías para el acceso al dataset
import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Se accede al archivo CSV
yaw_train = '/content/drive/MyDrive/port_coruna/yaw_train.csv'
df = pd.read_csv(yaw_train)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]

# Se crea la instancia (objeto específico del modelo) y se ajusta
el modelo a los datos
```

```
model = RandomForestRegressor()
model.fit(x, y)
prediction_rf = model.predict(x)
r2 = r2_score(y, prediction_rf)

# Se realizan predicciones del modelo y se muestra

plt.scatter(y, prediction_rf)
plt.xlabel(Guiñada de la embarcación (real)')
plt.ylabel(Guiñada de la embarcación (Predicción)')
plt.show()
print("Número de árboles en el bosque:", model.n_estimators)
print(r2)
```




```
Número de árboles en el bosque: 100
0.9720003667944749
```

Ilustración 19. Guiñada de los buques (Random Forest)

3.3.3 Marejada de los buques (Traslación Eje X)

```
# Se accede a las carpetas de Google Drive y se habilta la entrada
de código
from google.colab import drive
drive.mount('/content/drive')
from ast import Mult
# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

# Se importan las librerías para el acceso al dataset
import pandas as pd
import numpy as np
```

```
import csv
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Se accede al archivo CSV
surge_train = '/content/drive/MyDrive/port_coruna/surge_train.csv'
df = pd.read_csv(surge_train)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]

# Se crea la instancia (objeto específico del modelo) y se ajusta
el modelo a los datos
model = RandomForestRegressor()
model.fit(x, y)
prediction_rf = model.predict(x)
r2 = r2_score(y, prediction_rf)

# Se realizan predicciones del modelo y se muestra

plt.scatter(y, prediction_rf)
plt.xlabel('Marejada de la embarcación (real)')
plt.ylabel('Marejada de la embarcación (Predicción)')
plt.show()
print("Número de árboles en el bosque:", model.n_estimators)
print(r2)
```

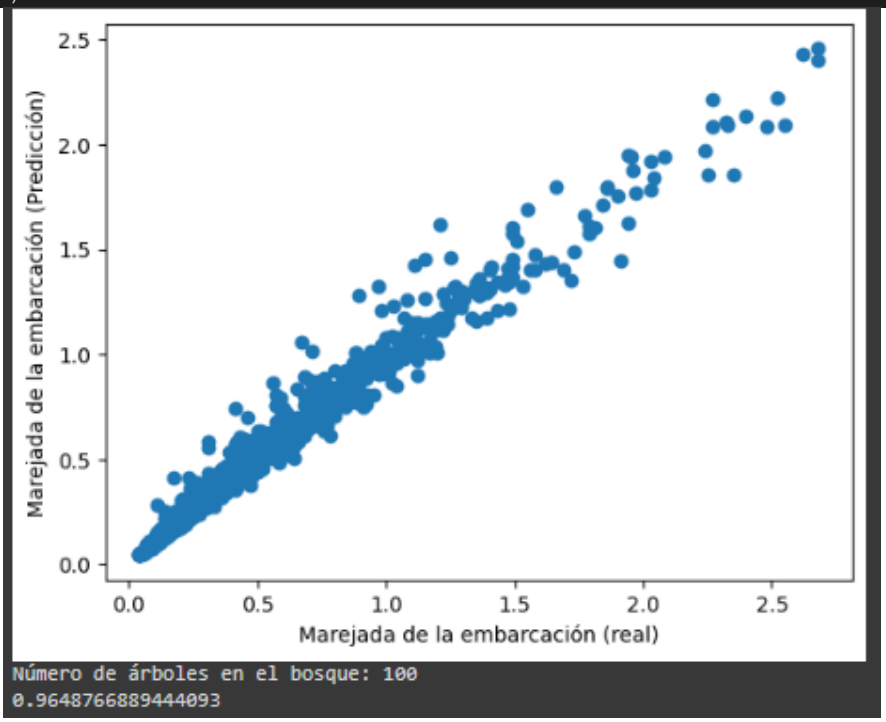


Ilustración 20. Marejada de los buques (Random Forest)

3.3.4 Escora o Balanceo (Rotación Eje X)

```
# Se accede a las carpetas de Google Drive y se habilta la entrada
de código
from google.colab import drive
drive.mount('/content/drive')
from ast import Mult
# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

# Se importan las librerías para el acceso al dataset
import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Se accede al archivo CSV
roll_train = '/content/drive/MyDrive/port_coruna/roll_train.csv'
df = pd.read_csv(roll_train)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]

# Se crea la instancia (objeto específico del modelo) y se ajusta
el modelo a los datos
model = RandomForestRegressor()
model.fit(x, y)
prediction_rf = model.predict(x)
r2 = r2_score(y, prediction_rf)

# Se realizan predicciones del modelo y se muestra

plt.scatter(y, prediction_rf)
plt.xlabel('Escora o Balanceo de la embarcación (real)')
plt.ylabel('Escora o Balanceo de la embarcación (Predicción)')
plt.show()
print("Número de árboles en el bosque:", model.n_estimators)
print(r2)
```

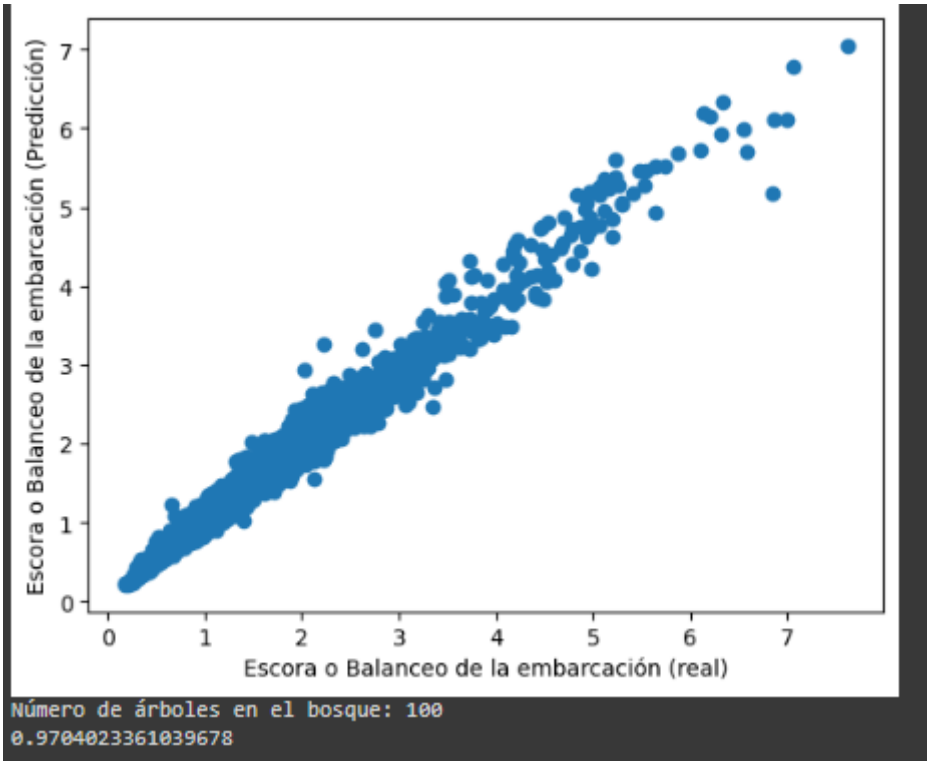


Ilustración 21. Escora o Balanceo de los buques (Random Forest)

3.3.5 Abatimiento / Ronza (Traslación Eje Y)

```
# Se accede a las carpetas de Google Drive y se habilta la entrada
de código
from google.colab import drive
drive.mount('/content/drive')
from ast import Mult
# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

# Se importan las librerías para el acceso al dataset
import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Se accede al archivo CSV
sway_train = '/content/drive/MyDrive/port_coruna/sway_train.csv'
df = pd.read_csv(sway_train)
df.dropna(inplace = True)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]

# Se crea la instancia (objeto específico del modelo) y se ajusta
el modelo a los datos
```

```
model = RandomForestRegressor()
model.fit(x, y)
prediction_rf = model.predict(x)
r2 = r2_score(y, prediction_rf)

# Se realizan predicciones del modelo y se muestra

plt.scatter(y, prediction_rf)
plt.xlabel('Abatimiento / Ronza (Traslación Eje Y)de la embarcación
(real)')
plt.ylabel('Abatimiento / Ronza (Traslación Eje Y) de la
embarcación (Predicción)')
plt.show()
print("Número de árboles en el bosque:", model.n_estimators)
print(r2)
```

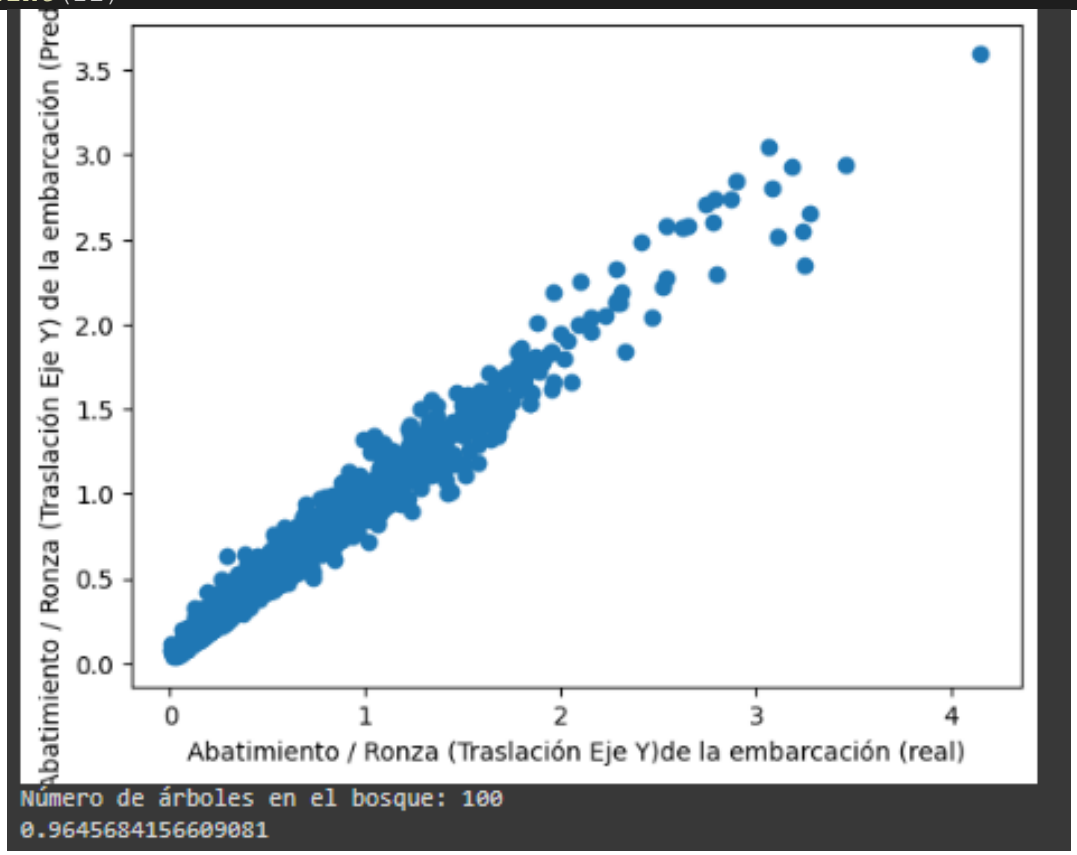


Ilustración 22. Abatimiento de los buques (Random Forest)

3.3.6 Cabeceo (Rotación Eje Y)

```
# Se accede a las carpetas de Google Drive y se habilta la entrada
de código
from google.colab import drive
drive.mount('/content/drive')
from ast import Mult
# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline
```

```
# Se importan las librerías para el acceso al dataset
import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Se accede al archivo CSV
pitch_train = '/content/drive/MyDrive/port_coruna/pitch_train.csv'
df = pd.read_csv(pitch_train)
x = df[['h_s', 't_p', 'dir', 'wind_speed', 'wind_dir', 'h_0',
'h_sm', 'length', 'breath']]
y = df[['mov_avg','mov_max','mov_sig']]

# Se crea la instancia (objeto específico del modelo) y se ajusta
el modelo a los datos
model = RandomForestRegressor()
model.fit(x, y)
prediction_rf = model.predict(x)
r2 = r2_score(y, prediction_rf)

# Se realizan predicciones del modelo y se muestra

plt.scatter(y, prediction_rf)
plt.xlabel('Cabeceo de la embarcación (real)')
plt.ylabel('Cabeceo de la embarcación (Predicción)')
plt.show()
print("Número de árboles en el bosque:", model.n_estimators)
print(r2)
```

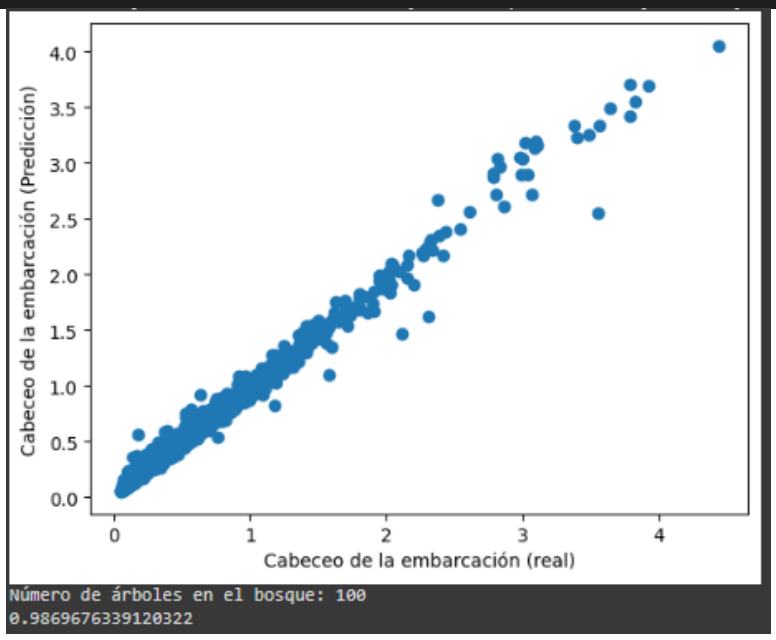


Ilustración 23. Cabeceo de los buques (Random Forest)

Como se observa, R Cuadrado ha mostrado el valor de R2 más alto, muy próximo a 1, por lo que es el que mejor se ajusta a los datos de entrenamiento, con lo cual, este modelo es el óptimo para aplicar al conjunto de datos de prueba. A continuación, se desarrolla el código empleado para aplicar Random Forest, obteniendo las predicciones correspondientes.

Es necesario mencionar que, a la hora de emplear el modelo entrenado con un conjunto de datos nuevo, R cuadrado está diseñado para medir la proporción de la varianza total en la variable dependiente que es explicada por el modelo, por tanto, el ajuste se evalúa mediante la importancia de las variables calculadas por el propio modelo, las cuales se corresponden con las variables de entrada y toma valores de 0 a 1. Esto se traduce como que se evalúa cada variable de entrada por separado según tengan mayor o menor incidencia al evaluar un nuevo conjunto de datos**. R cuadrado es una métrica útil para evaluar cómo se ajusta el modelo sólo a los datos de entrenamiento**.

Para obtener la importancia de las variables se emplea el atributo "*feature_importances_*" en Python. Las variables con una mayor importancia serán consideradas más significativas para el modelo. De tal manera que se pueden dimensionar los resultados de las mismas para identificar las características más relevantes y realizar una selección de las mismas si fuese necesario.

En este código, se carga el conjunto de prueba desde archivo CSV, del mismo modo que se hizo con el conjunto de entrenamiento. Se aplica el modelo como "model_rf" entrenando previamente el conjunto de prueba "x_test",

3.4 Resolución mediante Random Forest

En el código empleado que se muestra a continuación, se emplea el modelo entrenado ('model_rf') para realizar las predicciones sobre el conjunto de prueba (el cual incluye una variable nueva, 'dwt', que devuelve la carga del barco), llamando al método 'predict' y pasando como argumento 'x_test'. Las predicciones resultantes se almacenan en las variables 'prediction_rf'. Se utiliza el atributo 'feature_importances_' del modelo para obtener la importancia de cada variable de entrada en el proceso de predicción. El resultado se almacena en la variable 'importance'.

También es necesario destacar que se emplea el atributo GridSearchCV para optimizar todos los parámetros, mediante validación cruzada. En el caso de los algoritmos Random Forest, los parámetros más relevantes a optimizar son el número de árboles 'n_estimators' y la profundidad máxima de los árboles 'max_depth'. GridSearchCV realiza una búsqueda exhaustiva en un rango

predefinido de valores para cada parámetro, evaluando el rendimiento del modelo y devolviendo los mejores valores para cada uno.

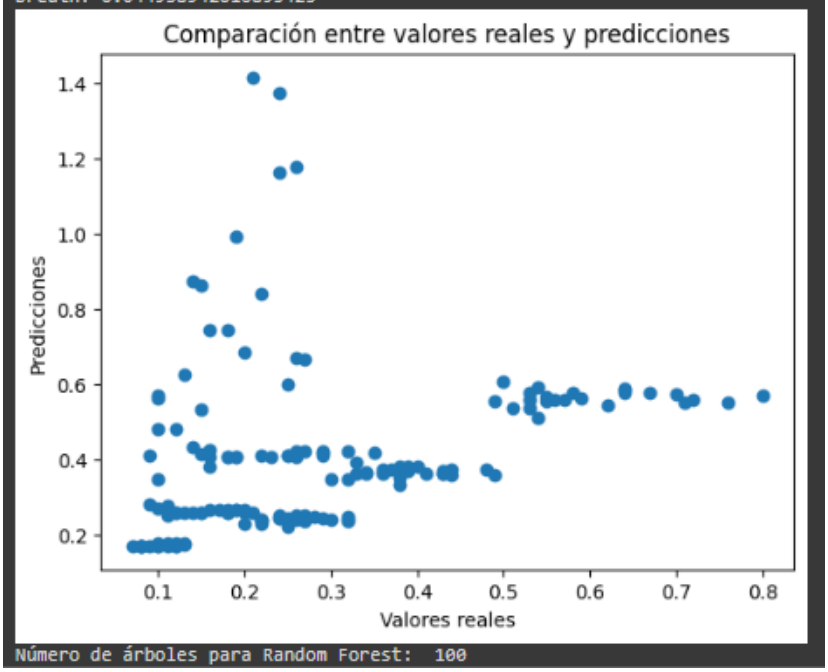
Por último, es importante nombrar que se aplica la predicción a 5 de 6 grados de libertad del buque debido a la falta de datos en el eje z (traslación, heave). Los resultados generales se comentan en el apartado *Conclusiones*.

3.4.1 Predicción en el Cabeceo del buque mediante Random Forest (Rotación Eje Y)

```
# ALGORITMO DE RANDOM FOREST
# Se accede a las carpetas de Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Se habilita entrada de código
from ast import Mult

# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

# Se importan las librerías necesarias
import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV

# Se definen los parámetros y los valores a testear
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 5, 10]
}

# Se accede al archivo CSV de entrenamiento
pitch_train =
'/content/drive/MyDrive/port_coruna/pitch_train.csv'
df = pd.read_csv(pitch_train)
df.dropna(inplace = True)
x_train = df[['dwt','h_s', 't_p', 'dir', 'wind_speed',
'wind_dir', 'h_0', 'h_sm', 'length', 'breath']]
y_train = df[['mov_avg', 'mov_max', 'mov_sig']]
# Entrenamiento del modelo Random Forest
model_rf = RandomForestRegressor()
grid_search = GridSearchCV(model_rf, param_grid, cv = 5)
model_rf.fit(x_train, y_train)
grid_search.fit(x_train, y_train)
```

```
best_estimator = grid_search.best_estimator_
# Se accede al conjunto de prueba
pitch_test = '/content/drive/MyDrive/port_coruna/pitch_test.csv'
df_test = pd.read_csv(pitch_test)
x_test = df_test[['dwt', 'h_s', 't_p', 'dir', 'wind_speed',
'wind_dir', 'h_0', 'h_sm', 'length', 'breath']]
y_test = df_test[['mov_avg', 'mov_max', 'mov_sig']]
# Se aplica el modelo Random Forest al conjunto de prueba
prediction_rf = best_estimator.predict(x_test)
r2_rf = r2_score(y_test, prediction_rf)
importance = model_rf.feature_importances_

print("Mejores parámetros:", best_estimator)
print("Importancia de características:")
for i, feature in enumerate(x_train.columns):
    print(f"{feature}: {importance[i]}")

plt.scatter(y_test, prediction_rf)
plt.xlabel('Valores reales')
plt.ylabel('Predicciones')
plt.title('Comparación entre valores reales y predicciones')
plt.show()
print("Número de árboles para Random Forest: ",
model_rf.n_estimators)
```

```
Mejores parámetros: RandomForestRegressor(max_depth=5, n_estimators=200)
Importancia de características:
dwt: 0.4434710373478867
h_s: 0.10102247573767163
t_p: 0.023388892590111725
dir: 0.08488974741055573
wind_speed: 0.033572273808209055
wind_dir: 0.03097406052355228
h_0: 0.015138656195880604
h_sm: 0.03330793868701732
length: 0.18929597568221965
breath: 0.044938942016895425
```




```
Número de árboles para Random Forest:  100
```

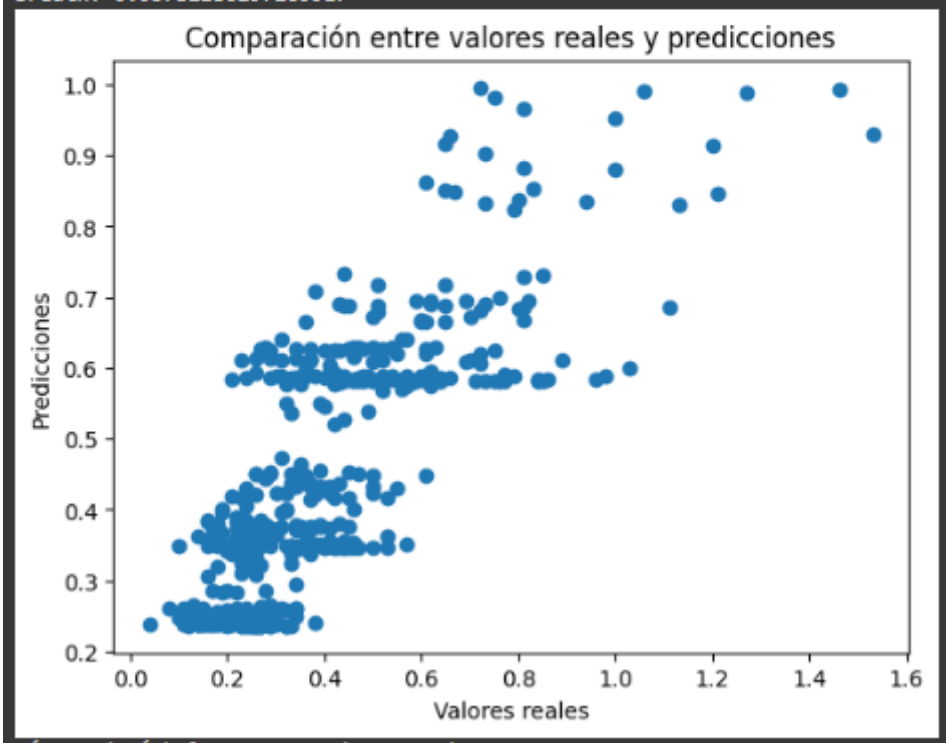
Ilustración 24. Mejores parámetros en la Guiñada de un buque

3.4.2 Predicción en el Abatimiento del buque mediante Random Forest (Traslación Eje Y)

```
# ALGORITMO DE RANDOM FOREST
# Se accede a las carpetas de Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Se habilita entrada de código
from ast import Mult

# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

# Se importan las librerías necesarias
import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV
```

```
# Se definen los parámetros y los valores a testear
param_grid = {
   'n_estimators': [100, 200, 300],
   'max_depth': [None, 5, 10]
}

# Se accede al archivo CSV de entrenamiento
sway_train = '/content/drive/MyDrive/port_coruna/sway_train.csv'
df = pd.read_csv(sway_train)
df.dropna(inplace = True)
x_train = df[['dwt','h_s', 't_p', 'dir', 'wind_speed', 'wind_dir',
'h_0', 'h_sm', 'length', 'breath']]
y_train = df[['mov_avg', 'mov_max', 'mov_sig']]
# Entrenamiento del modelo Random Forest
model_rf = RandomForestRegressor()
grid_search = GridSearchCV(model_rf, param_grid, cv = 5)
model_rf.fit(x_train, y_train)
grid_search.fit(x_train, y_train)
best_estimator = grid_search.best_estimator_
# Se accede al conjunto de prueba
sway_test = '/content/drive/MyDrive/port_coruna/sway_test.csv'
df_test = pd.read_csv(sway_test)
x_test = df_test[['dwt', 'h_s', 't_p', 'dir', 'wind_speed',
'wind_dir', 'h_0', 'h_sm', 'length', 'breath']]
y_test = df_test[['mov_avg', 'mov_max', 'mov_sig']]
# Se aplica el modelo Random Forest al conjunto de prueba
prediction_rf = best_estimator.predict(x_test)
r2_rf = r2_score(y_test, prediction_rf)
importance = model_rf.feature_importances_

print("Mejores parámetros:", best_estimator)
print("Importancia de características:")
for i, feature in enumerate(x_train.columns):
    print(f"{feature}: {importance[i]}")

plt.scatter(y_test, prediction_rf)
plt.xlabel('Valores reales')
plt.ylabel('Predicciones')
plt.title('Comparación entre valores reales y predicciones')
plt.show()
print("Número de árboles para Random Forest: ",
model_rf.n_estimators)
```

```
Mejores parámetros: RandomForestRegressor(max_depth=5)
Importancia de características:
dwt: 0.11406045229117166
h_s: 0.3683246252134038
t_p: 0.08562974222065739
dir: 0.058958944100918634
wind_speed: 0.05374400887127314
wind_dir: 0.07814340631987915
h_0: 0.0728927871513998
h_sm: 0.04624139262964739
length: 0.06418283822894985
breath: 0.05782180297269917
```




```
Número de árboles para Random Forest:  100
```

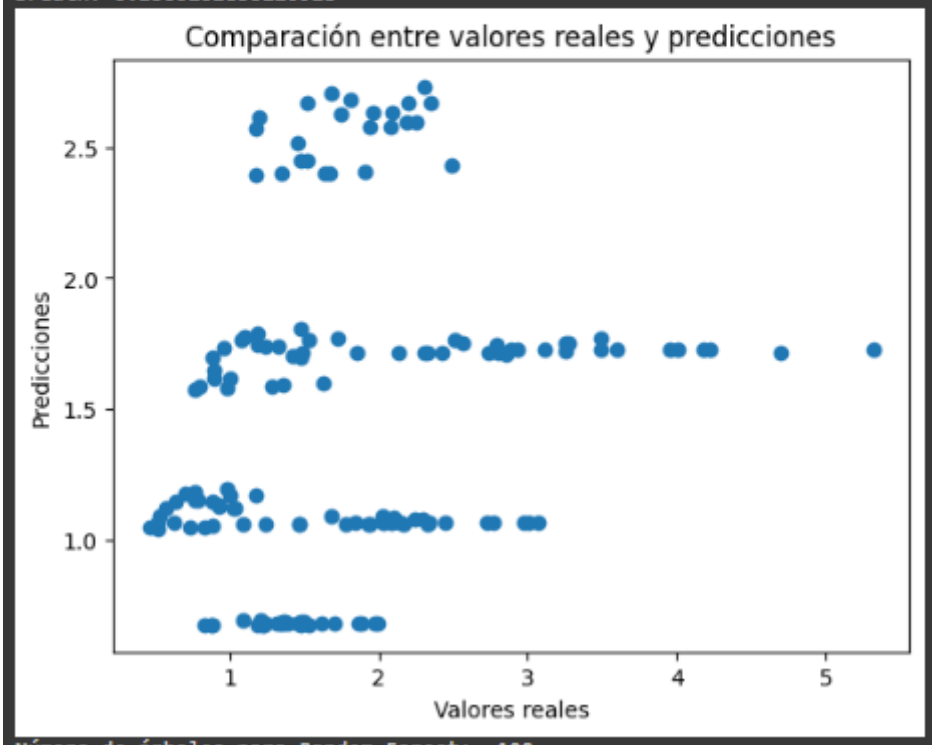
Ilustración 25. Mejores parámetros en el Abatimiento de un buque

3.4.3 Predicción en la Escora o Balance de un buque mediante Random Forest (Rotación Eje X)

```
# ALGORITMO DE RANDOM FOREST
# Se accede a las carpetas de Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Se habilita entrada de código
from ast import Mult

# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

# Se importan las librerías necesarias
import pandas as pd
import numpy as np
import csv
```

```
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV

# Se definen los parámetros y los valores a testear
param_grid = {
   'n_estimators': [100, 200, 300],
   'max_depth': [None, 5, 10]
}

# Se accede al archivo CSV de entrenamiento
roll_train = '/content/drive/MyDrive/port_coruna/roll_train.csv'
df = pd.read_csv(roll_train)
df.dropna(inplace = True)
x_train = df[['dwt','h_s', 't_p', 'dir', 'wind_speed', 'wind_dir',
'h_0', 'h_sm', 'length', 'breath']]
y_train = df[['mov_avg', 'mov_max', 'mov_sig']]
# Entrenamiento del modelo Random Forest
model_rf = RandomForestRegressor()
grid_search = GridSearchCV(model_rf, param_grid, cv = 5)
model_rf.fit(x_train, y_train)
grid_search.fit(x_train, y_train)
best_estimator = grid_search.best_estimator_
# Se accede al conjunto de prueba
roll_test = '/content/drive/MyDrive/port_coruna/roll_test.csv'
df_test = pd.read_csv(roll_test)
x_test = df_test[['dwt', 'h_s', 't_p', 'dir', 'wind_speed',
'wind_dir', 'h_0', 'h_sm', 'length', 'breath']]
y_test = df_test[['mov_avg', 'mov_max', 'mov_sig']]
# Se aplica el modelo Random Forest al conjunto de prueba
prediction_rf = best_estimator.predict(x_test)
r2_rf = r2_score(y_test, prediction_rf)
importance = model_rf.feature_importances_

print("Mejores parámetros:", best_estimator)
print("Importancia de características:")
for i, feature in enumerate(x_train.columns):
    print(f"{feature}: {importance[i]}")

plt.scatter(y_test, prediction_rf)
plt.xlabel('Valores reales')
plt.ylabel('Predicciones')
plt.title('Comparación entre valores reales y predicciones')
plt.show()
print("Número de árboles para Random Forest: ",
model_rf.n_estimators)
```

```
Mejores parámetros: RandomForestRegressor(max_depth=5, n_estimators=300)
Importancia de características:
dwt: 0.07954760059841005
h_s: 0.17734286332152036
t_p: 0.05912973787316629
dir: 0.06817337856087854
wind_speed: 0.03954282937529349
wind_dir: 0.04545324985318242
h_0: 0.0414534479029487
h_sm: 0.06294538117847867
length: 0.2703852263140287
breath: 0.1560262850220928
```




```
Número de árboles para Random Forest:  100
```

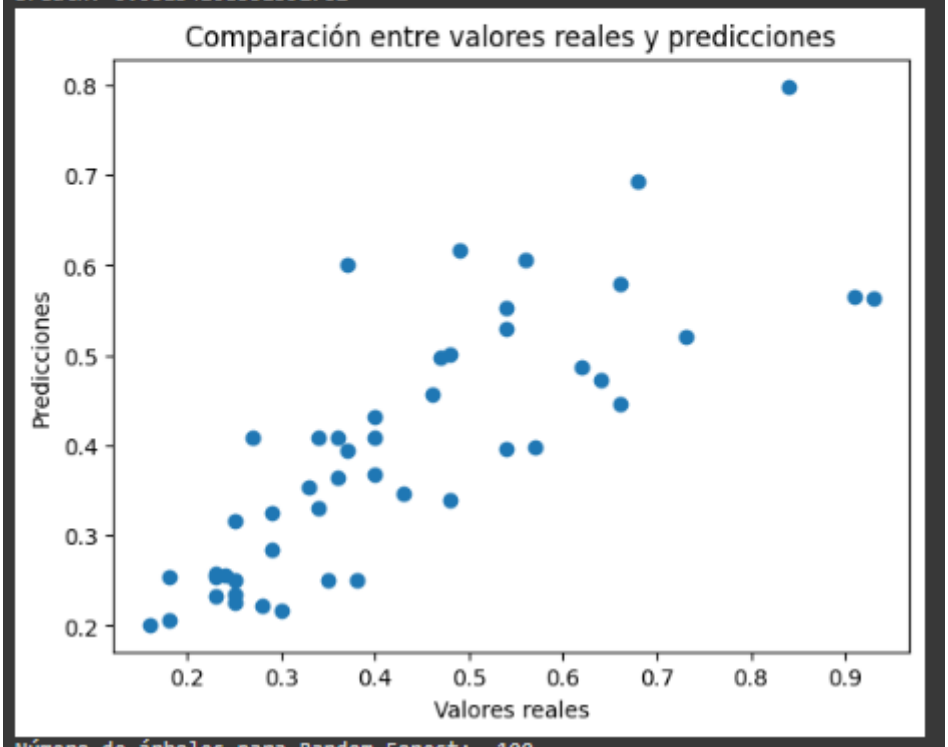
Ilustración 26. Mejores parámetros en la Escora de un buque

3.4.4 Predicción en la Marejada del buque mediante Random Forest (Traslación Eje X)

```
# ALGORITMO DE RANDOM FOREST
# Se accede a las carpetas de Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Se habilita entrada de código
from ast import Mult

# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

# Se importan las librerías necesarias
import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV

# Se definen los parámetros y los valores a testear
param_grid = {
   'n_estimators': [100, 200, 300],
   'max_depth': [None, 5, 10]
}

# Se accede al archivo CSV de entrenamiento
surge_train = '/content/drive/MyDrive/port_coruna/surge_train.csv'
df = pd.read_csv(surge_train)
df.dropna(inplace = True)
x_train = df[['dwt','h_s', 't_p', 'dir', 'wind_speed', 'wind_dir',
'h_0', 'h_sm', 'length', 'breath']]
y_train = df[['mov_avg', 'mov_max', 'mov_sig']]
# Entrenamiento del modelo Random Forest
model_rf = RandomForestRegressor()
grid_search = GridSearchCV(model_rf, param_grid, cv = 5)
model_rf.fit(x_train, y_train)
grid_search.fit(x_train, y_train)
best_estimator = grid_search.best_estimator_
# Se accede al conjunto de prueba
surge_test = '/content/drive/MyDrive/port_coruna/surge_test.csv'
df_test = pd.read_csv(surge_test)
x_test = df_test[['dwt', 'h_s', 't_p', 'dir', 'wind_speed',
'wind_dir', 'h_0', 'h_sm', 'length', 'breath']]
y_test = df_test[['mov_avg', 'mov_max', 'mov_sig']]
# Se aplica el modelo Random Forest al conjunto de prueba
prediction_rf = best_estimator.predict(x_test)
r2_rf = r2_score(y_test, prediction_rf)
importance = model_rf.feature_importances_

print("Mejores parámetros:", best_estimator)
print("Importancia de características:")
for i, feature in enumerate(x_train.columns):
    print(f"{feature}: {importance[i]}")

plt.scatter(y_test, prediction_rf)
plt.xlabel('Valores reales')
plt.ylabel('Predicciones')
plt.title('Comparación entre valores reales y predicciones')
plt.show()
print("Número de árboles para Random Forest: ",
model_rf.n_estimators)
```

```
Mejores parámetros: RandomForestRegressor(max_depth=10)
Importancia de características:
dwt: 0.06749989201192093
h_s: 0.14568532604084033
t_p: 0.3890773728073863
dir: 0.1059160587538479
wind_speed: 0.04970809359562335
wind_dir: 0.041732240403012405
h_0: 0.03816698130433136
h_sm: 0.058058212070351424
length: 0.052814737496758406
breath: 0.05134108551592762
```




```
Número de árboles para Random Forest:  100
```

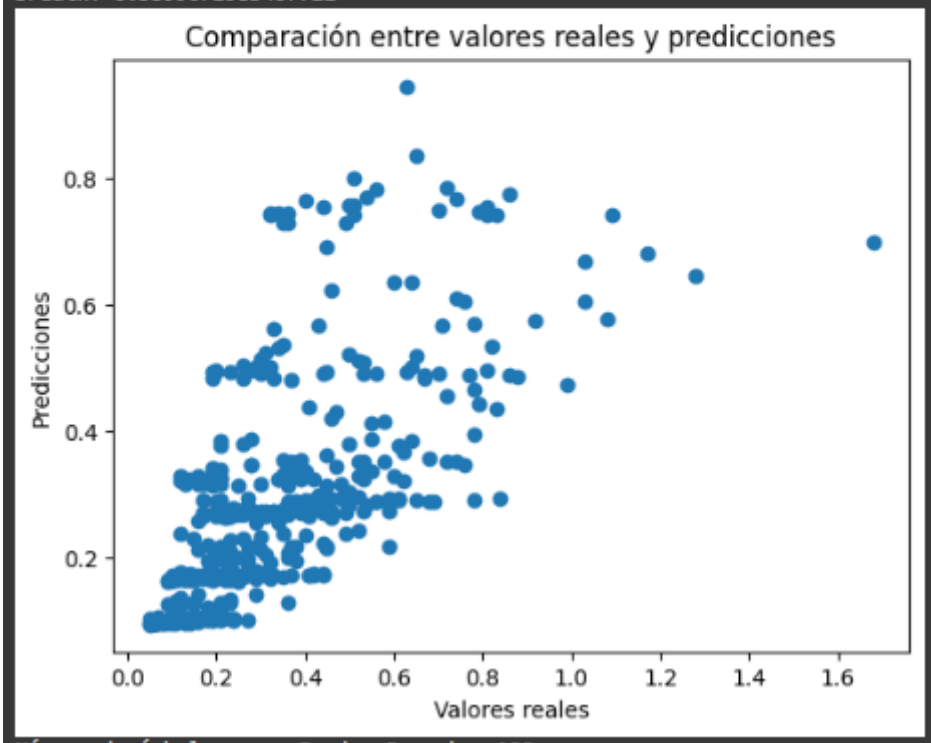
Ilustración 27. Mejores parámetros en la Marejada de un buque

3.4.5 Predicción en la Guiñada de un buque mediante Random Forest (Rotación Eje Z)

```
# ALGORITMO DE RANDOM FOREST
# Se accede a las carpetas de Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Se habilita entrada de código
from ast import Mult

# Se habilitan gráficos dentro de Jupyter Notebook
%matplotlib inline

# Se importan las librerías necesarias
import pandas as pd
import numpy as np
import csv
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV

# Se definen los parámetros y los valores a testear
param_grid = {
   'n_estimators': [100, 200, 300],
   'max_depth': [None, 5, 10]
}

# Se accede al archivo CSV de entrenamiento
yaw_train = '/content/drive/MyDrive/port_coruna/yaw_train.csv'
df = pd.read_csv(yaw_train)
df.dropna(inplace = True)
x_train = df[['dwt','h_s', 't_p', 'dir', 'wind_speed', 'wind_dir',
'h_0', 'h_sm', 'length', 'breath']]
y_train = df[['mov_avg', 'mov_max', 'mov_sig']]
# Entrenamiento del modelo Random Forest
model_rf = RandomForestRegressor()
grid_search = GridSearchCV(model_rf, param_grid, cv = 5)
model_rf.fit(x_train, y_train)
grid_search.fit(x_train, y_train)
best_estimator = grid_search.best_estimator_
# Se accede al conjunto de prueba
yaw_test = '/content/drive/MyDrive/port_coruna/yaw_test.csv'
df_test = pd.read_csv(yaw_test)
x_test = df_test[['dwt', 'h_s', 't_p', 'dir', 'wind_speed',
'wind_dir', 'h_0', 'h_sm', 'length', 'breath']]
y_test = df_test[['mov_avg', 'mov_max', 'mov_sig']]
# Se aplica el modelo Random Forest al conjunto de prueba
prediction_rf = best_estimator.predict(x_test)
r2_rf = r2_score(y_test, prediction_rf)
importance = model_rf.feature_importances_

print("Mejores parámetros:", best_estimator)
print("Importancia de características:")
for i, feature in enumerate(x_train.columns):
    print(f"{feature}: {importance[i]}")

plt.scatter(y_test, prediction_rf)
plt.xlabel('Valores reales')
plt.ylabel('Predicciones')
plt.title('Comparación entre valores reales y predicciones')
plt.show()
print("Número de árboles para Random Forest: ",
model_rf.n_estimators)
```

```
Mejores parámetros: RandomForestRegressor(max_depth=5, n_estimators=300)
Importancia de características:
dwt: 0.09481446989607593
h_s: 0.07801624552481363
t_p: 0.07490166162653268
dir: 0.03629646364127692
wind_speed: 0.06413247114127663
wind_dir: 0.04777552921392562
h_0: 0.04033427697568941
h_sm: 0.08655740782695369
length: 0.11726417581268331
breath: 0.3599072983407722
```




```
Número de árboles para Random Forest:  100
```

Ilustración 28. Mejores parámetros en la Guiñada de un buque

4. Conclusiones

Una vez desarrollado el código en Python que ejecuta el algoritmo ensamblado de Random Forest al conjunto de datos obtenidos, se presentan los resultados.

- En primer lugar, en el cabeceo del barco (rotación eje y) influye principalmente la carga del barco y en menor medida, la eslora del mismo, esto es, la longitud lineal de proa a popa.
- En el abatimiento (traslación eje y) se obtiene un mayor movimiento en la altura de la ola.
- En la escora (rotación eje x), el parámetro más influyente es la eslora o longitud del barco.
- En la marejada (traslación eje x), el tiempo entre olas es el parámetro con mayor magnitud.
- En la guiñada (rotación eje x), el parámetro más influyente es el ancho del buque.

5. Bibliografía

1. Apuntes Máster Data Science & Business Intelligence UCAV & IMF Smart Education [Módulo 4. Machine Learning / www.imf.com/]
2. Repositorio de datos en [https://github.com/aalvarell/ship-movement-dataset]
3. Algoritmo Random Forest explicación [https://towardsdatascience.com/random-forest-regression-5f605132d19d]
4. Movimientos de un buque atracado en puerto [https://es.wikipedia.org/wiki/Movimiento_y_oscilaci%C3%B3n_de_un_barco]

A Coruña, 13 de junio de 2023

Julián Vázquez Sampedro