DAFTAR PUSTAKA

I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.

M. Anthony and P. L. Bartlett. Neural network learning: theoretical foundations. Cambridge University Press, Cambridge, 1999.

Lippmann, R.P. An introduction to computing with neural nets. IEEE Trans. ASSP Mag. 1987, 4, 4– 22. [CrossRef].

Makhoul, J. Pattern recognition properties of neural networks. In Proceedings of the 1991 IEEE Workshop on Neural Networks for Signal Processing, Princeton, NJ, USA, 29 September–2 October 1991. [CrossRef].

Hush, D.R.; Horne, B.G. Progress in supervised neural networks. IEEE Signal Process. Mag. 1993, 10, 8–39. [CrossRef]

Bishop, C.M. Pattern Recognition and Machine Learning; Springer: Berlin, Germany, 2006.

Riedmiller, M.; Braun, H. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In Proceedings of the IEEE International Conference on Neural Networks (ICNN'93), San Francisco, CA, USA, 28 March–1 April 1993. [CrossRef].

J. W. G. Putra, "Pengenalan Konsep Pembelajaran Mesin dan Deep Learning," *Comput. Linguist. Nat. Lang. Process. Lab.*, vol. 4, pp. 1–235, 2019, [Online]. Available: https://www.researchgate.net/publication/323700644.

S. Sena, "Pengenalan Deep Learning," *Medium*, pp. 1–7, 2017, [Online]. Available: https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94.

D. Riesel, “A brief Introduction on Neural Networks,” *Springer-Verlag, Berlin*, p. Second edition, 2007. D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber.

Sandi Baressi Šegota, Nikola Anđelić, Jan Kudláček, Robert Čep. Artificial neural network for predicting values of residuary resistance per unit weight of displacement. Pomorski zbornik 57 (2019), 9-22

Ziółkowski, J.; Oszczypała, M.; Małachowski, J.; Szkutnik-Rogoz, J. Use of Artificial Neural Networks to Predict Fuel Consumption on the Basis of Technical Parameters of Vehicles. Energies 2021, 14, 2639. https://doi.org/10.3390/en14092639

Burak Yıldız, Prediction of Residual Resistance of a Trimaran vessel by using an Artificial Neural Network. Volume 73 Number 1, 2022. http://dx.doi.org/10.21278/brod73107

Marti´c, I.; Degiuli, N.;Majeti´c, D.; Farkas, A. ArtificialNeural Network Model for the

Evaluation of Added Resistance of Container Ships in Head Waves. J.Mar. Sci. Eng. 2021, 9, 826. https:// doi.org/10.3390/jmse9080826

Moreira, Lc.; Vettor, R.; Guedes Soares, C. Neural Network Approach for Predicting Ship Speed and Fuel Consumption. J.Mar.Sci. Eng.2021, 9, 119. https://doi.org/ 10.3390/jmse9020119

Postprint of: Grabowska K., Szczuko P., Ship resistance prediction with Artificial Neural Networks, Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Poznan (2015), pp. 168-173, doi: 10.1109/SPA.2015.7365154

Ema Muk-Pavic, Resistance Prediction using Artificial Neural Networks for Preliminary Tri-SWACH Design. ans RINA, Vol XXX, Part A1, Intl J Maritime Engineering 2013 Jun-Dec.

Google.com, 10 April 2022. https://www.analyticsvidhya. com/blog/2021/10/ implementing - artificial-neural-networkclassification-in-python-from-scratch/

Google.com, 10 April 2022. https://www.analyticsvidhya. com/ blog /2022/01/ introduction- to -neural- networks/

Google.com, 10 April 2022. https://www. analyticsvidhya. com /blog /2021 /07 /understanding-the-basics-of-artificial-neural-network-ann/

Google.com,10April2022.https://www.kdnuggets.com/2018/10/simple-neural-network-python.html

Google.com, 11 April 2022. https://realpython.com/python-ai-neural-network/https://www.kdnuggets.com/2018/10/simple-neural-network-python.html

Google.com, 11 April 2022. https://python-course.eu/machine-learning/simple-neural-network-from-scratch-in-python.php

Google.com, 12 April 2022. https://towardsdatascience.com/deep-learning-with-python-neural-networks-complete-tutorial-6b53c0b06af0

Google.com, 12 April 2022. https://www.geeksforgeeks.org/activation-functions-neural-networks/?ref=lbp

https://docs.python.org/3/license.html

https://www.mathworks.com/products/matlab/student.html?s_tid=products

LAMPIRAN

LAMPIRAN CODING MATLAB ANN

```
% This script assumes these variables are defined:
% This script prepared by Ardyan Safiu. Student of Hasanuddin
University:
% Faculty of Engineering, Departement of Naval Architect 2022:
%
%   X - input data.c
%   Y - target data.
clear
clc
Data = readtable('dataset2.xlsx');
[m,n] = size(Data);
X = (table2array(Data(1:m,2:6)));
Y = (table2array(Data(1:m,7)));
x = X';
t = Y';

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging
problems.
% 'trainscg' uses less memory. Suitable in low memory
situations.
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.
```

```
% Create a Fitting Network
hiddenLayerSize = ([30 25 15]);
net = fitnet(hiddenLayerSize,trainFcn);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.input.processFcns = {'removeconstantrows','mapminmax'};
net.output.processFcns = {'removeconstantrows','mapminmax'};

% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help
nndivision
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Choose a Performance Function
% For a list of all performance functions type: help
nnperformance
net.performFcn = 'mse';  % Mean Squared Error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist',
...
    'plotregression', 'plotfit'};

% Train the Network
```

```
net.trainparam.epochs = 1000;
net.trainparam.goal = 1e-25;
net.trainparam.lr = 0.001;
[net,tr] = train(net,x,t);
save net

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)

% Recalculate Training, Validation and Test Performance
trainTargets = t .* tr.trainMask{1};
valTargets = t .* tr.valMask{1};
testTargets = t .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotregression(t,y)
%figure, plotfit(net,x,t)
```

```
% Deployment
% Change the (false) values to (true) to enable the following
code blocks.
% See the help for each generation function for more
information.
if (false)
    % Generate MATLAB function for neural network for
application
    % deployment in MATLAB scripts or with MATLAB Compiler and
Builder
    % tools, or simply to examine the calculations your
trained neural
    % network performs.
    y = myNeuralNetworkFunction(x);
end
if (false)
    % Generate a matrix-only MATLAB function for neural
network code
    % generation with MATLAB Coder tools.

genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');
    y = myNeuralNetworkFunction(x);
end
if (false)
    % Generate a Simulink diagram for simulation or deployment
with.
    % Simulink Coder tools.
    gensim(net);
end
```

LAMPIRAN CODING APLIKASI GUI MATLAB

```
function varargout = Aplikasi(varargin)
% APLIKASI MATLAB code for Aplikasi.fig
%      APLIKASI, by itself, creates a new APLIKASI or raises
the existing
%      singleton*.
%
%      H = APLIKASI returns the handle to a new APLIKASI or
the handle to
%      the existing singleton*.
%
%      APLIKASI('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in APLIKASI.M with the given
input arguments.
%
%      APLIKASI('Property','Value',...) creates a new APLIKASI
or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before Aplikasi_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes
property application
%      stop.  All inputs are passed to Aplikasi_OpeningFcn via
varargin.
%
```

```
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Aplikasi

% Last Modified by GUIDE v2.5 24-Aug-2022 18:38:06

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Aplikasi_OpeningFcn, ...
                   'gui_OutputFcn',  @Aplikasi_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before Aplikasi is made visible.
function Aplikasi_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to Aplikasi (see VARARGIN)

% Choose default command line output for Aplikasi
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
load net
handles.net =net
% UIWAIT makes Aplikasi wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command
line.
function varargout = Aplikasi_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
load net
lwl = str2num(get(handles.LWL,'string'))
B= str2num(get(handles.B,'string'))
H =str2num(get(handles.H,'string'))
summerDraft=str2num(get(handles.draft,'string'))
Speed=str2num(get(handles.speed,'string'))
z =[lwl, B, H, summerDraft, ...
    Speed]
z =transpose(z);
 hasil = net(z)
set(handles.power, 'String',hasil);

function LWL_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to LWL (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of LWL as text
%        str2double(get(hObject,'String')) returns contents of
LWL as a double


% --- Executes during object creation, after setting all
properties.
function LWL_CreateFcn(hObject, eventdata, handles)
% hObject    handle to LWL (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function B_Callback(hObject, eventdata, handles)
% hObject    handle to B (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of B as text
%        str2double(get(hObject,'String')) returns contents of
B as a double


% --- Executes during object creation, after setting all
properties.
function B_CreateFcn(hObject, eventdata, handles)
% hObject    handle to B (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function H_Callback(hObject, eventdata, handles)
% hObject    handle to H (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of H as text
%        str2double(get(hObject,'String')) returns contents of
H as a double


% --- Executes during object creation, after setting all
properties.
function H_CreateFcn(hObject, eventdata, handles)
% hObject    handle to H (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function draft_Callback(hObject, eventdata, handles)
% hObject    handle to draft (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of draft as
text
%        str2double(get(hObject,'String')) returns contents of
draft as a double


% --- Executes during object creation, after setting all
properties.
function draft_CreateFcn(hObject, eventdata, handles)
% hObject    handle to draft (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end


function speed_Callback(hObject, eventdata, handles)
% hObject    handle to speed (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of speed as
text
%        str2double(get(hObject,'String')) returns contents of
speed as a double


% --- Executes during object creation, after setting all
properties.
function speed_CreateFcn(hObject, eventdata, handles)
% hObject    handle to speed (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end


function power_Callback(hObject, eventdata, handles)
% hObject    handle to power (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of power as
text
%        str2double(get(hObject,'String')) returns contents of
power as a double


% --- Executes during object creation, after setting all
properties.
function power_CreateFcn(hObject, eventdata, handles)
% hObject    handle to power (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
```

LAMPIRAN I_DATA KAPAL

No.	SHIP	LoA	Lwl	Breadth	Depth	Draft	Freeboard	DWT	Rpm	Speed	L/B	B/T	L/H	Power
1	MAR DE CORTES	115.33	109.08	19.6	11.4	8.632	2.813	12274	210	15.2	5.565	2.271	9.568	3603
2	BLUE LOTUS	134.04	127.2	21.2	10.8	7.941	2.892	14187	158	16.1	6	2.67	11.778	4413
3	ATAYAL STAR	135.01	128.02	23	11.5	8.448	3.098	16805	173	14.7	5.566	2.723	11.132	4440
4	COQUIMBO I	134.98	128.35	23	11.5	8.54	2.991	17013	173	15.2	5.58	2.693	11.161	4440
	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1463	CHINA STEEL ENDEAVOR	299.99	296.02	50	25	18.43	6.635	209749	84	17	5.92	2.713	11.841	16040
1464	CHINA STEEL SUCCESS	299.99	296.02	50	25	18.43	6.635	209988	84	16.7	5.92	2.713	11.841	16040
1465	CAPE NORMANDY	291.9	296.9	45	24.5	18.068	6.49	180646	91	17.2	6.598	2.491	12.118	17690
1466	AWOBASAN MARU	319.58	309.15	54	25.1	18.122	7.036	226371	74	17.2	5.725	2.98	12.317	20445
1467	CAPE RALIANCE	329.95	321.78	57	25.1	18.025	7.138	250877	76	17.5	5.645	3.162	12.82	23280

	LoA	Lwl	Breadth	Depth	Summer-draft	Freeboard	Rpm	Speed	L/B	B/T	L/H	Power
count	1468	1468	1468	1468	1468	1468	1468	1468	1468	1468	1468	1468
mean	213.6498	207.6015	34.10821	18.67157	13.36372	5.366858	107.6444	16.20525	6.119328	2.566195	11.10464	9831.67
std	39.95897	39.84042	6.114442	3.226971	2.488205	0.807869	18.66167	0.550559	1.395707	0.227613	0.733032	3624.134
min	113.33	107.02	19.4	10.4	7.941	2.411	12	14.2	5.291	0.277	5.823	1686
25%	189.9	182.47	32.24	17.15	12.149	5.026	91	15.9	5.759	2.468	10.505	7775
50%	199.98	195	32.26	18.5	12.948	5.433	108	16.1	5.967	2.545	11.1745	8880
75%	228.99	224.23	32.27	20	14.429	5.675	116	16.5	6.317	2.71425	11.509	10223
max	329.95	321.78	57	25.4	18.473	10.171	210	18.3	56.819	3.551	23.406	23280

LAMPIRAN_II_PENGELOMPOKAN DATA KAPAL

No.	L107-L185	B107-B185	H107-H185	T107-T185	V107-V185	P107-P185	L185-L221	B185-B221	H185-H221	T185-T221	V185-V221	P185-P221	L221-L321	B221-B321	H221-H321	T221-T321	V221-V321	P221-P321
0	107.02	19.4	10.4	8.023	16.3	3883	185	32.26	17.8	12.53	16	8200	221.31	38	19.9	13.819	17	12240
1	109.08	19.6	11.4	8.632	15.2	3603	185	32.26	17.8	12.53	16.3	8200	221.31	38	19.9	13.819	16.9	12240
2	127.2	21.2	10.8	7.941	16.1	4413	185	32.26	17.8	12.522	16.3	8200	221.31	38	19.9	13.819	17.1	12240
3	128.02	23	11.5	8.448	14.7	4440	185	32.26	17.8	12.53	15.9	8200	221.31	38	19.9	13.819	16.8	12268
4	128.35	23	11.5	8.54	15.2	4440	185	32.26	17.8	12.522	16.2	8200	221.31	38	19.9	13.819	16.9	12240
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
484	185	32.26	17.8	12.522	16.1	8200	221.31	38	19.9	13.819	16.8	12240	296.02	50	25	18.43	17	16040
485	185	32.26	17.8	12.53	16	8200	221.31	38	19.9	13.819	16.9	12240	296.02	50	25	18.43	16.7	16040
486	185	32.26	17.8	12.522	16.1	8200	221.31	38	19.9	13.819	16.9	12240	296.9	45	24.5	18.068	17.2	17690
487	185	32.26	17.8	12.522	16.2	8200	221.31	38	19.9	13.819	16.9	12240	309.15	54	25.1	18.122	17.2	20445
488	185	32.26	17.8	12.522	16	8200	221.31	38	19.9	13.819	17.1	12240	321.78	57	25.1	18.025	17.5	23280

LAMPIRAN_III_DATA UNIQUE VARIABEL X DAN VARIABEL Y

1. *Length Water Line (Lwl)*

Data Length water line ditunjukan pada matriks berikut ini.

```
array([182.97, 195.39, 217.68, 175.33, 296.02, 172.32, 221.  , 222.54,
       217.97, 224.23, 194.85, 221.31, 195.38, 193.61, 194.  , 132.48,
       173.52, 167.76, 185.75, 195.  , 182.99, 172.33, 185.64, 163.8 ,
       281.19, 185.78, 217.11, 220.18, 169.35, 136.51, 229.45, 168.56,
       197.  , 181.71, 181.99, 181.9 , 185.  , 179.91, 178.08, 281.61,
       186.48, 150.19, 150.75, 282.96, 236.  , 183.06, 182.87, 286.59,
       227.51, 249.91, 295.17, 182.11, 161.21, 187.  , 175.93, 196.09,
       145.54, 178.27, 230.23, 217.4 , 164.33, 170.58, 181.64, 226.17,
       172.56, 218.84, 225.1 , 185.96, 217.81, 280.45, 223.  , 173.91,
       227.71, 193.57, 291.53, 218.03, 284.28, 128.54, 221.1 , 180.37,
       217.71, 183.05, 221.61, 143.32, 182.66, 128.02, 161.22, 170.57,
       172.88, 227.18, 309.15, 183.61, 247.64, 216.87, 292.17, 294.27,
       181.62, 233.04, 179.36, 165.7 , 238.74, 224.69, 187.69, 222.02,
       196.13, 174.54, 146.13, 291.95, 284.19, 286.9 , 148.85, 170.01,
       219.48, 146.75, 295.  , 127.2 , 143.97, 184.13, 173.69, 179.95,
       218.33, 205.43, 280.16, 218.63, 182.48, 217.28, 173.97, 179.6 ,
       279.45, 169.45, 225.  , 291.94, 288.  , 292.96, 295.2 , 284.23,
       283.39, 283.03, 293.05, 296.9 , 321.78, 287.2 , 196.5 , 189.99,
       185.94, 182.65, 220.53, 184.28, 223.28, 130.11, 216.08, 219.4 ,
       169.93, 178.93, 194.57, 176.59, 222.74, 222.58, 107.02, 128.35,
       223.11, 223.32, 186.35, 223.42, 132.38, 222.64, 134.86, 216.02,
       185.92, 185.95, 185.72, 283.74, 154.69, 241.6 , 169.17, 183.12,
       182.47, 217.79, 218.68, 228.12, 225.4 , 140.7 , 181.  , 225.52,
       220.31, 164.19, 218.  , 216.25, 287.92, 281.  , 172.95, 136.9 ,
       161.35, 149.67, 149.81, 128.64, 287.5 , 228.02, 175.77, 180.38,
       219.19, 177.47, 230.17, 241.85, 183.15, 225.17, 221.75, 178.5 ,
       284.2 , 225.02, 173.  , 218.82, 245.09, 225.45, 177.02, 284.29,
       154.68, 166.06, 220.16, 173.45, 169.09, 196.11, 295.5 , 280.2 ,
       178.04, 134.14, 162.45, 205.17, 151.18, 182.  , 174.53, 166.45])
```

2. Breadth

Data Breadth ditunjukan pada matriks berikut ini.

```
array([32.26 , 32.24 , 30.95 , 50.   , 28.2  , 38.   , 36.   , 25.   ,
       29.8  , 29.4  , 32.25 , 30.5  , 27.   , 47.   , 28.8  , 22.8  ,
       43.   , 32.2  , 31.   , 30.4  , 45.   , 26.   , 44.98 , 27.2  ,
       29.5  , 28.6  , 30.   , 23.   , 54.   , 25.8  , 21.2  , 28.4  ,
       28.   , 49.   , 57.   , 36.5  , 27.8  , 19.4  , 35.   , 24.   ,
       32.296, 23.6  , 22.   , 32.   , 22.4  , 26.8  ])
```

3. Depth

Data Depth ditunjukan pada matriks berikut ini.

```
array([17.9 , 18.6 , 19.3 , 16.4 , 25.  , 14.3 , 19.85, 20.05, 19.5 ,
       20.2 , 18.5 , 19.9 , 19.15, 18.45, 18.1 , 11.5 , 15.  , 13.7 ,
       18.  , 15.3 , 13.8 , 24.4 , 19.25, 19.39, 15.2 , 12.2 , 20.  ,
       13.5 , 16.5 , 16.9 , 17.8 , 17.15, 20.03, 24.7 , 13.3 , 19.95,
       17.3 , 17.  , 18.55, 16.7 , 13.6 , 16.  , 17.45, 14.8 , 19.8 ,
       17.87, 14.5 , 16.3 , 18.7 , 17.2 , 13.2 , 17.1 , 13.9 , 14.4 ,
       25.1 , 19.1 , 24.5 , 24.1 , 20.7 , 21.65, 20.01, 16.8 , 18.33,
       14.05, 13.35, 24.8 , 12.7 , 10.8 , 13.75, 19.2 , 24.65, 14.25,
       24.98, 17.62, 19.89, 11.4 , 32.26, 13.4 , 15.6 , 20.1 , 10.4 ,
       22.05, 29.8 , 12.8 , 18.4 , 14.  , 15.8 , 25.4 , 12.4 , 14.2 ,
       14.7 , 12.5 , 14.63, 16.55, 14.6 ,  9.3 , 17.6 , 16.24, 27.2 ,
       19.79, 19.22, 17.7 , 24.75, 16.47])
```

4. Summer Draft

Data Summer Draft ditunjukan pada matrix di bawah ini.

```
array([12.575, 13.01 , 13.994, 11.807, 18.43 , 10.031, 14.328, 14.429,
       14.139, 14.598, 13.318, 13.819, 13.418, 12.925, 12.676,  8.444,
       10.54 ,  9.569, 12.828, 13.025, 12.715, 10.931, 12.818,  9.718,
       17.975, 12.827, 13.928, 14.189, 10.718,  9.12 , 14.424,  9.562,
       13.026, 11.623, 11.62 , 11.925, 12.522, 12.163, 11.6  , 13.036,
       18.17 , 12.735, 10.541,  9.518,  9.801, 13.065, 17.985, 14.479,
       12.3  , 12.024, 18.236, 12.861, 13.599, 18.23 , 12.303, 11.769,
        9.819, 12.527, 12.826, 10.366, 12.955, 10.045, 12.573, 12.331,
       13.054, 13.842, 10.418, 12.825, 10.869, 11.739, 14.475, 12.564,
       12.747, 10.1  , 14.221, 12.569, 14.028, 17.955, 14.555, 17.973,
        9.929, 12.936, 14.195, 14.468, 12.948, 18.473, 12.53 , 14.078,
        9.64 , 14.038, 18.235,  8.47 , 14.375, 11.373, 13.553, 12.508,
       14.526,  9.572, 12.041,  8.448,  9.82 ,  9.77 , 10.121, 18.174,
        9.641, 14.225, 12.572, 18.122, 12.724, 13.465, 13.87 , 18.105,
       17.91 , 12.291, 14.73 , 12.151,  9.568, 10.101, 10.542, 15.404,
       14.486,  9.716, 10.419, 12.225, 14.466, 14.194, 14.408, 12.85 ,
        9.822,  9.717, 18.27 , 18.322, 18.237, 18.068,  9.115,  9.415,
       14.4  , 12.568, 14.379, 10.536,  9.553, 18.225,  7.941,  9.517,
        9.668,  8.496,  8.446, 12.109, 10.488, 10.027,  9.566, 12.022,
       14.122, 12.789, 14.446, 12.026, 13.871, 14.378, 10.538, 11.614,
       18.22 , 10.019, 13.518, 18.226, 18.171, 18.222, 18.28 , 18.369,
       18.025, 18.246, 18.173, 13.023, 12.502, 12.486, 12.815, 13.02 ,
       13.972,  8.325, 13.066, 14.126, 14.407, 12.82 ,  9.538, 10.998,
       12.136, 11.705, 13.323, 10.367, 14.568, 14.52 , 18.372,  8.023,
        8.54 , 13.995, 13.904, 13.969, 13.982,  8.316, 13.055, 12.149,
       14.467,  8.473,  9.065, 12.173,  9.56 , 14.013,  9.779, 12.447,
       18.37 , 12.951,  9.67 , 17.85 ,  9.899, 12.487, 12.845, 12.536,
       12.907, 10.416, 10.02 , 11.276, 14.464, 18.249, 13.317, 14.029,
       13.521, 14.009, 15.268, 13.978, 18.221,  9.086, 10.674, 12.578,
       12.725,  9.778, 14.518, 14.519, 14.435, 10.043, 13.787, 13.952,
       18.18 , 18.029, 18.224, 18.223, 10.116,  9.414,  8.918,  9.748,
       14.136, 12.172,  9.616, 14.426,  9.764,  8.8  ,  9.745, 17.95 ,
       12.917, 10.329, 10.032, 13.059, 11.695, 12.832, 12.821, 10.016,
       13.073, 12.11 , 11.651, 14.648, 12.   , 12.424, 12.8  , 18.324,
       11.576, 13.821, 12.863, 14.477, 13.869, 14.469, 10.769, 18.308,
        9.895, 10.624, 10.034, 14.427, 10.115, 13.507, 14.228, 14.421,
       18.425, 17.81 , 11.778,  8.4  , 13.499, 12.824, 12.93 ,  9.104,
        9.114,  9.76 , 12.448, 11.671, 11.617, 10.348])
```

5. Speed

Data *Unique Speed* ditunjukan pada matriks di berikut ini.

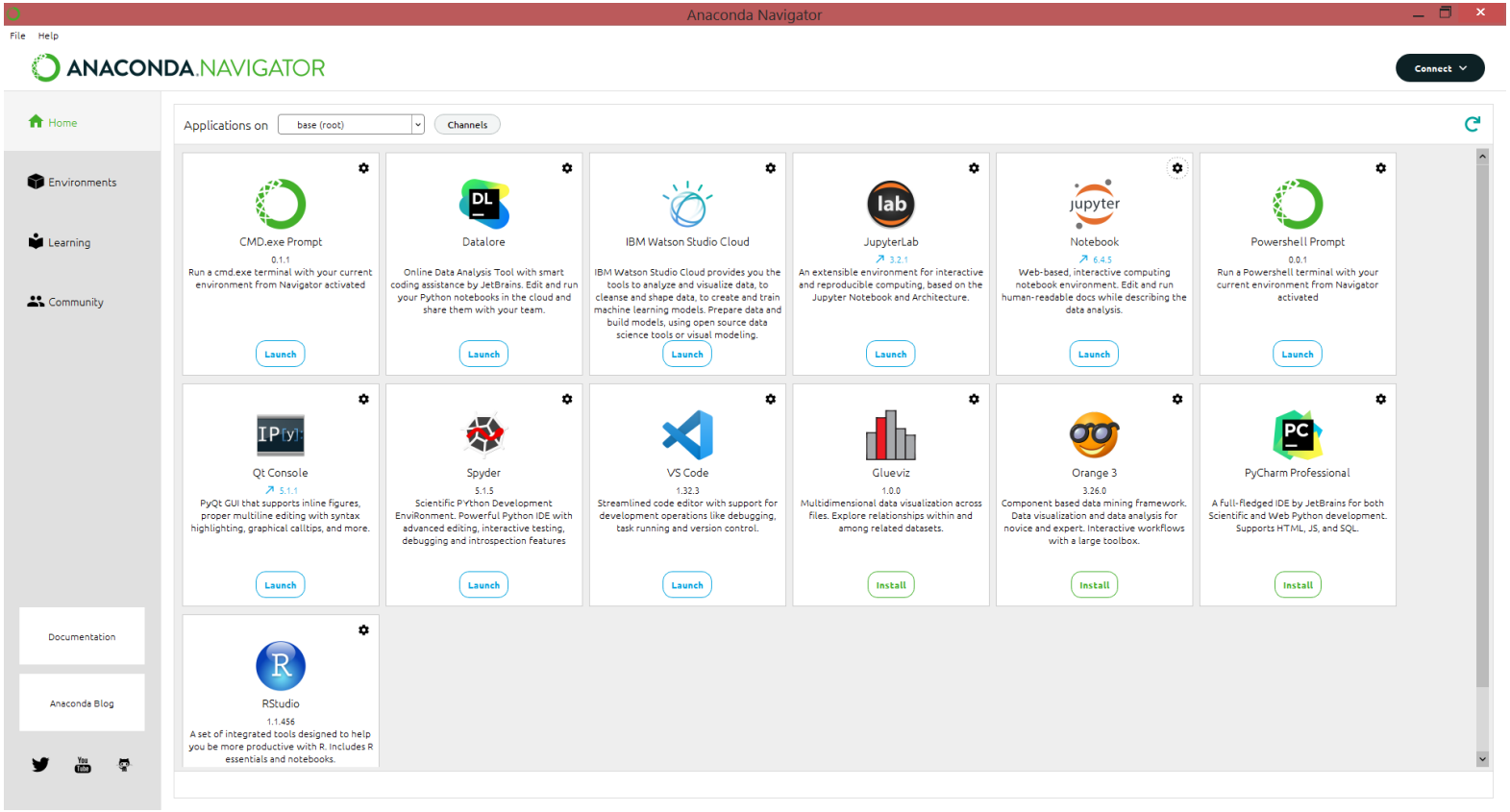
```
array([16.3, 15.2, 16.1, 14.7, 15.5, 15.3, 15.7, 15.8, 15.9, 15.1, 14.8,
       15. , 16.2, 14.4, 16.9, 15.4, 16.5, 15.6, 16. , 16.4, 16.8, 16.6,
       14.5, 14.6, 16.7, 17. , 14.3, 17.2, 17.1, 14.2, 17.3, 14.9, 17.5,
       17.4, 17.6, 17.8, 17.7, 18.3, 17.9])
```

6. Power

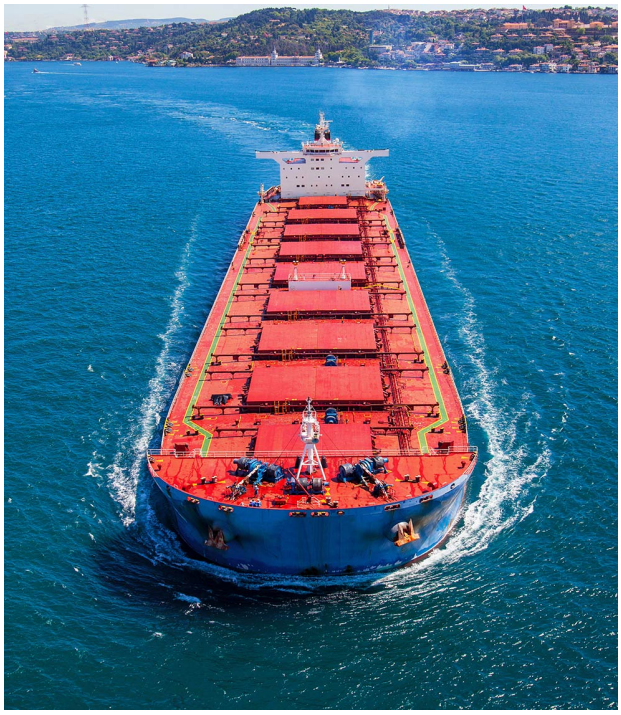
Data *Unique Power* ditunjukan pada matriks berikut ini

```
array([ 3883,  3603,  4413,  4440,  4300,  4200,  4280,  4635,  5180,
        5589,  4983,  4891,  5641,  4900,  6150,  5295,  6230,  6178,
        5185,  4589,  5000,  5296,  6156,  5850,  6480,  5847,  5736,
        5884,  6803,  6545,  5800,  5729,  7060,  7080,  5714,  6031,
        6840,  6620,  6780,  6656,  7300,  6800,  5940,  6619,  5148,
        7470,  5970,  7050,  6250,  5810,  5580,  6000,  5972,  7900,
        7240,  6820,  7860,  7487,  5720,  7024,  6050,  5650,  7250,
        6370,  6100,  7038,  7171,  8311,  7640,  9480,  7700,  7650,
        6695,  7760,  7980,  6830,  7686,  8046,  8045,  7685,  7870,
        7207,  8100,  8995,  8090,  7428,  8164,  7800,  7391,  7796,
        8250,  8209,  9466,  9465,  7722,  7649,  9070,  8000,  7390,
        8200,  8500,  8400,  8450,  8480,  8208,  8708,  7135,  7612,
        7260,  7500,  8530,  8125,  7134,  8470,  7910,  8890,  7730,
       10473,  8630,  8300,  8050,  9120,  9000,  8370,  7850,  7560,
        8260,  7510,  8605,  8201,  7880,  8130, 10115,  8110,  8826,
       10517,  8789,  8678,  8899,  8877,  8385,  8880,  8380, 10220,
        9260,  9800,  8973,  9340,  9780, 10223,  9145,  8830,  8550,
       10500,  9010,  9350, 10320,  9930, 10003,  8330,  9760,  9230,
        8700,  9855,  9560,  9318,  9326,  9319, 11032,  9170, 12100,
       12240, 12268,  9378, 10150,  9660,  9700,  9710, 11000, 11200,
       10738, 12000,  9960, 10480,  9966,  9414, 11620, 11289, 10740,
       10170,  9590,  9750,  8740, 11040, 11915, 11910, 10400, 12950,
       13200, 15150,  9880, 10335, 11106, 11033, 13560, 12700, 10820,
       11400, 12270, 11547, 17640, 16860, 14710, 16090, 18630, 18660,
       18600,  1686, 15483, 15450, 16700, 15980, 18780, 17690, 15800,
       15700, 16580, 17780,  7780, 15860, 16200, 16420, 17140, 16650,
       16020, 17470, 19000, 16610, 16810, 16830, 15250, 16040, 20445,
       23280], dtype=int64)
```

LAMPIRAN CODING PYTHON



THESIS- Bulk Carrier



In [2]:
```
# Import library
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Import dataset
df=pd.read_csv("ANN_Data_Check_Rev_8.csv")
df
# If we want to take the first 5 rows, use this: data_visualization.head()
# If we want to take the last 5 rows, use this: data_visualization.tail()
# if we want to know total row: len(data_visualization)
```

Out[2]:

	Ships-Name	LoA	Lwl	B	H	Summer-Draft	Summer-Freeboard	Dead-Weight	Rpm	Speed	L/B	B/T	L/H	Power
0	COMET	113.33	107.02	19.4	10.4	8.023	2.411	10095	210	16.3	5.516	2.418	10.290	3883
1	MAR DE CORTES	115.33	109.08	19.6	11.4	8.632	2.813	12274	210	15.2	5.565	2.271	9.568	3603
2	BLUE LOTUS	134.04	127.20	21.2	10.8	7.941	2.892	14187	158	16.1	6.000	2.670	11.778	4413
3	ATAYAL STAR	135.01	128.02	23.0	11.5	8.448	3.098	16805	173	14.7	5.566	2.723	11.132	4440
4	COQUIMBO I	134.98	128.35	23.0	11.5	8.540	2.991	17013	173	15.2	5.580	2.693	11.161	4440
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1463	CHINA STEEL ENDEAVOR	299.99	296.02	50.0	25.0	18.430	6.635	209749	84	17.0	5.920	2.713	11.841	16040
1464	CHINA STEEL SUCCESS	299.99	296.02	50.0	25.0	18.430	6.635	209988	84	16.7	5.920	2.713	11.841	16040
1465	CAPE NORMANDY	291.90	296.90	45.0	24.5	18.068	6.490	180646	91	17.2	6.598	2.491	12.118	17690
1466	AWOBASAN MARU	319.58	309.15	54.0	25.1	18.122	7.036	226371	74	17.2	5.725	2.980	12.317	20445
1467	CAPE RALIANCE	329.95	321.78	57.0	25.1	18.025	7.138	250877	76	17.5	5.645	3.162	12.820	23280

In [3]:
```
# create headers list
headers = ["Ships-name","LoA","Lwl","Breadth", "Depth","Summer-draft",
           "Freeboard","Dead-weight","Rpm", "Speed","L/B","B/T","L/H","Power"]
df.columns = headers
df.head()
```

Out[3]:

	Ships-name	LoA	Lwl	Breadth	Depth	Summer-draft	Freeboard	Dead-weight	Rpm	Speed	L/B	B/T	L/H	Power
0	COMET	113.33	107.02	19.4	10.4	8.023	2.411	10095	210	16.3	5.516	2.418	10.290	3883
1	MAR DE CORTES	115.33	109.08	19.6	11.4	8.632	2.813	12274	210	15.2	5.565	2.271	9.568	3603
2	BLUE LOTUS	134.04	127.20	21.2	10.8	7.941	2.892	14187	158	16.1	6.000	2.670	11.778	4413
3	ATAYAL STAR	135.01	128.02	23.0	11.5	8.448	3.098	16805	173	14.7	5.566	2.723	11.132	4440
4	COQUIMBO I	134.98	128.35	23.0	11.5	8.540	2.991	17013	173	15.2	5.580	2.693	11.161	4440

In [4]:
```
df.head()
```

Out[4]:

	Ships-name	LoA	Lwl	Breadth	Depth	Summer-draft	Freeboard	Dead-weight	Rpm	Speed	L/B	B/T	L/H	Power
0	COMET	113.33	107.02	19.4	10.4	8.023	2.411	10095	210	16.3	5.516	2.418	10.290	3883
1	MAR DE CORTES	115.33	109.08	19.6	11.4	8.632	2.813	12274	210	15.2	5.565	2.271	9.568	3603
2	BLUE LOTUS	134.04	127.20	21.2	10.8	7.941	2.892	14187	158	16.1	6.000	2.670	11.778	4413
3	ATAYAL STAR	135.01	128.02	23.0	11.5	8.448	3.098	16805	173	14.7	5.566	2.723	11.132	4440
4	COQUIMBO I	134.98	128.35	23.0	11.5	8.540	2.991	17013	173	15.2	5.580	2.693	11.161	4440

In [5]:
```
df.tail()
```

Out[5]:

	Ships-name	LoA	Lwl	Breadth	Depth	Summer-draft	Freeboard	Dead-weight	Rpm	Speed	L/B	B/T	L/H	Power
1463	CHINA STEEL ENDEAVOR	299.99	296.02	50.0	25.0	18.430	6.635	209749	84	17.0	5.920	2.713	11.841	16040
1464	CHINA STEEL SUCCESS	299.99	296.02	50.0	25.0	18.430	6.635	209988	84	16.7	5.920	2.713	11.841	16040
1465	CAPE NORMANDY	291.90	296.90	45.0	24.5	18.068	6.490	180646	91	17.2	6.598	2.491	12.118	17690
1466	AWOBASAN MARU	319.58	309.15	54.0	25.1	18.122	7.036	226371	74	17.2	5.725	2.980	12.317	20445
1467	CAPE RALIANCE	329.95	321.78	57.0	25.1	18.025	7.138	250877	76	17.5	5.645	3.162	12.820	23280

In [6]:
```
len (df)
```

Out[6]: 1468

In [7]:
```
df.dtypes
```

Out[7]:
```
Ships-name       object
LoA             float64
Lwl             float64
Breadth         float64
Depth           float64
Summer-draft    float64
Freeboard       float64
Dead-weight      object
Rpm               int64
Speed           float64
L/B             float64
B/T             float64
L/H             float64
Power             int64
dtype: object
```

In [8]:
```
df['LoA'].unique()
```

Out[8]:
```
array([113.33, 115.33, 134.04, 135.01, 134.98, 134.93, 135.8 , 137.03,
       139.91, 139.92, 141.03, 145.  , 148.17, 149.17, 150.53, 150.52,
       153.2 , 153.78, 154.38, 154.5 , 154.35, 153.5 , 157.7 , 157.5 ,
       157.03, 158.  , 157.26, 159.92, 157.23, 160.8 , 161.  , 169.37,
       169.26, 163.03, 169.03, 169.36, 169.54, 170.  , 170.7 , 169.99,
       171.59, 172.  , 171.99, 176.5 , 175.53, 175.5 , 176.82, 176.85,
       177.  , 175.6 , 169.93, 176.62, 176.6 , 178.41, 177.85, 179.99,
       172.32, 180.8 , 179.9 , 180.  , 181.16, 179.97, 179.94, 176.99,
       183.04, 179.96, 183.  , 181.  , 179.95, 182.  , 185.73, 185.74,
       186.  , 182.98, 182.93, 186.4 , 182.99, 188.5 , 182.87, 188.33,
       187.3 , 190.44, 189.96, 189.33, 189.9 , 189.8 , 189.82, 189.94,
       189.99, 189.6 , 189.95, 190.  , 189.93, 194.94, 199.99, 197.  ,
       199.9 , 198.  , 199.98, 199.92, 210.  , 209.28, 224.97, 225.  ,
       224.89, 224.95, 224.94, 224.99, 224.79, 224.98, 229.  , 230.  ,
       229.93, 224.86, 224.9 , 228.99, 229.98, 228.41, 229.02, 228.05,
       228.95, 228.94, 228.92, 234.88, 235.  , 234.98, 234.94, 234.93,
       234.99, 240.  , 239.99, 245.  , 249.94, 249.91, 249.98, 255.22,
       254.99, 254.62, 289.  , 288.97, 289.98, 290.  , 288.93, 292.  ,
       291.92, 291.97, 291.98, 291.9 , 291.95, 299.97, 299.94, 299.95,
       300.  , 299.7 , 299.99, 319.58, 329.95])
```

In [9]:
```
df['Lwl'].unique()
```

Out[9]:
```
array([107.02, 109.08, 127.2 , 128.02, 128.35, 128.54, 128.64, 130.11,
       132.38, 132.48, 134.14, 134.86, 136.51, 136.9 , 140.7 , 143.32,
       143.97, 145.54, 146.13, 146.75, 148.85, 149.67, 149.81, 150.19,
       150.75, 151.18, 154.68, 154.69, 161.21, 161.22, 161.35, 162.45,
       162.77, 162.86, 163.8 , 164.19, 164.33, 165.7 , 166.06, 166.45,
       167.76, 168.56, 169.09, 169.17, 169.35, 169.45, 169.93, 170.01,
       170.57, 170.58, 172.32, 172.33, 172.56, 172.88, 172.95, 173.  ,
       173.45, 173.52, 173.69, 173.91, 173.97, 174.53, 174.54, 175.33,
       175.54, 175.77, 175.93, 176.59, 177.02, 177.47, 178.04, 178.08,
       178.27, 178.5 , 178.81, 178.93, 179.36, 179.6 , 179.91, 179.95,
       180.37, 180.38, 181.  , 181.62, 181.64, 181.71, 181.9 , 181.99,
       182.  , 182.11, 182.47, 182.48, 182.65, 182.66, 182.87, 182.97,
       182.99, 183.05, 183.06, 183.12, 183.15, 183.61, 184.13, 184.28,
       185.  , 185.64, 185.72, 185.75, 185.78, 185.92, 185.94, 185.95,
       185.96, 186.35, 186.48, 187.  , 187.69, 189.99, 193.57, 193.61,
       194.  , 194.57, 194.85, 195.  , 195.38, 195.39, 196.05, 196.09,
       196.11, 196.13, 196.5 , 197.  , 205.17, 205.43, 216.02, 216.08,
       216.25, 216.87, 217.11, 217.28, 217.4 , 217.68, 217.71, 217.79,
       217.81, 217.97, 218.  , 218.03, 218.33, 218.63, 218.68, 218.82,
       218.84, 219.19, 219.4 , 219.48, 220.16, 220.18, 220.31, 220.53,
       221.  , 221.1 , 221.31, 221.61, 221.75, 222.02, 222.54, 222.58,
       222.64, 222.74, 223.  , 223.11, 223.28, 223.32, 223.42, 224.23,
       224.69, 225.  , 225.02, 225.1 , 225.17, 225.4 , 225.45, 225.5 ,
       225.52, 226.17, 227.18, 227.51, 227.71, 228.02, 228.12, 229.45,
       230.17, 230.23, 233.04, 236.  , 238.74, 241.6 , 241.85, 245.09,
       247.64, 249.91, 279.45, 280.16, 280.2 , 280.45, 281.  , 281.19,
       281.61, 282.96, 283.03, 283.39, 283.74, 284.19, 284.2 , 284.23,
       284.28, 284.29, 286.59, 286.9 , 287.2 , 287.5 , 287.92, 288.  ,
       288.42, 291.53, 291.94, 291.95, 292.17, 292.96, 293.05, 294.27,
       295.  , 295.17, 295.2 , 295.5 , 296.02, 296.9 , 309.15, 321.78])
```

In [10]:
```
df['Breadth'].unique()
```

Out[10]:
```
array([19.4 , 19.6 , 21.2 , 23.  , 22.  , 25.  , 22.4 , 24.  , 22.8 ,
       26.  , 25.8 , 27.  , 26.8 , 27.2 , 26.6 , 28.  , 29.4 , 29.5 ,
       28.4 , 28.8 , 28.6 , 28.2 , 30.5 , 30.  , 31.  , 29.8 , 30.95,
       32.  , 30.4 , 32.26, 30.6 , 27.8 , 32.2 , 23.6 , 32.25, 32.3 ,
       36.  , 32.24, 36.5 , 38.  , 35.  , 43.  , 45.  , 47.  , 44.98,
       50.  , 49.  , 54.  , 57.  ])
```

In [11]:
```
df['Depth'].unique()
```

Out[11]:
```
array([10.4 , 11.4 , 10.8 , 11.5 , 12.2 , 12.8 , 12.5 , 12.4 , 13.2 ,
       13.75, 16.  , 13.35, 13.3 , 12.7 , 13.5 , 14.  , 13.6 , 13.8 ,
       14.2 , 14.8 , 15.  , 14.6 , 13.7 , 14.5 , 15.2 , 14.25, 13.4 ,
       13.9 , 14.3 , 15.3 , 14.4 , 14.7 , 16.24, 16.47, 14.05, 16.4 ,
       14.63, 16.5 , 17.45, 17.6 , 15.6 , 17.15, 16.3 , 16.55, 17.3 ,
       16.9 , 17.  , 16.7 , 15.8 , 17.62, 17.1 , 17.9 , 18.1 , 17.2 ,
       18.5 , 17.8 , 18.  , 18.4 , 17.87, 16.8 , 18.45, 18.6 , 19.15,
       18.54, 19.22, 18.33, 18.7 , 17.7 , 19.3 , 19.39, 19.1 , 19.25,
       19.2 , 19.5 , 19.9 , 19.8 , 19.79, 19.85, 20.05, 22.05, 20.  ,
       20.03, 20.1 , 20.2 , 19.89, 20.01, 18.55, 25.4 , 20.7 , 19.95,
       21.65, 24.65, 24.4 , 24.1 , 24.7 , 24.8 , 24.75, 24.5 , 24.55,
       25.  , 24.98, 25.1 ])
```

In [12]:
```
df['Summer-draft'].unique()
```

Out[12]:
```
array([ 8.023,  8.632,  7.941,  8.448,  8.54 ,  8.47 ,  8.8  ,  8.325,
        8.316,  8.473,  8.444,  8.496,  8.446,  8.4  ,  9.065,  9.12 ,
        9.104,  8.918,  9.086,  9.572,  9.566,  9.517, 10.045,  9.717,
        9.518,  9.668,  9.837,  9.553,  9.115,  9.77 ,  9.616,  9.801,
        9.114,  9.895,  9.899,  9.819,  9.779,  9.67 ,  9.778,  9.745,
        9.76 ,  9.82 ,  9.748,  9.764,  9.869,  9.75 ,  9.718,  9.716,
       10.043, 10.418, 10.419,  9.568, 10.624, 10.348,  9.569,  9.64 ,
        9.641,  9.562,  9.56 , 10.115, 10.416, 10.718, 10.019, 10.02 ,
       10.018,  9.538,  9.415,  9.414, 10.869, 10.031, 10.1  , 10.101,
       10.032, 10.931, 10.121, 10.116,  9.917, 11.576, 10.034, 10.54 ,
       10.541, 10.542, 10.536, 10.538, 10.488,  9.929, 11.671,  9.822,
       11.807, 11.81 , 10.621, 10.329, 10.366, 10.367, 10.769, 10.519,
       10.016, 11.778, 11.6  , 11.62 , 11.617, 12.331, 12.424, 10.021,
       10.998, 12.151, 12.149, 11.614, 12.163, 12.173, 12.172, 10.027,
       11.373, 11.695, 10.674, 12.291, 11.739, 11.623, 11.705, 11.925,
       12.303, 12.022, 11.769, 11.276, 12.026, 12.486, 12.487, 12.041,
       12.024, 12.   , 12.575, 12.573, 12.572, 12.715, 12.747, 12.508,
       12.3  , 11.651, 12.724, 12.725, 12.109, 12.136, 13.02 , 12.522,
       12.564, 12.53 , 12.818, 12.8  , 12.951, 12.925, 12.828, 12.827,
       12.826, 12.825, 12.447, 12.448, 12.502, 12.569, 12.568, 12.536,
       12.578, 12.735, 12.527, 12.225, 12.948, 12.676, 13.323, 13.318,
       13.317, 13.025, 13.036, 13.065, 12.936, 13.418, 13.01 , 13.059,
       12.955, 13.507, 13.499, 12.85 , 12.845, 13.023, 13.026, 12.824,
       12.789, 14.013, 14.009, 14.126, 13.952, 13.87 , 13.928, 13.871,
       13.842, 13.994, 13.995, 13.553, 14.029, 14.028, 14.038, 14.139,
       13.787, 14.078, 14.122, 14.446, 13.521, 13.821, 14.221, 14.225,
       14.136, 12.821, 12.82 , 14.379, 14.427, 14.189, 14.195, 14.194,
       14.435, 12.815, 14.328, 14.378, 14.375, 13.819, 14.526, 14.426,
       14.429, 14.408, 14.407, 14.4  , 14.464, 14.52 , 14.568, 14.555,
       13.904, 13.972, 13.969, 13.982, 13.978, 14.518, 13.869, 14.467,
       14.598, 14.486, 13.518, 14.469, 14.648, 14.477, 14.668, 14.519,
       14.475, 14.228, 12.861, 12.93 , 12.929, 14.468, 14.466, 12.832,
       14.421, 12.917, 15.268, 14.424, 13.073, 13.054, 13.066, 13.055,
       14.73 , 14.479, 15.404, 12.907, 12.11 , 12.863, 13.465, 13.599,
       18.22 , 18.249, 17.975, 17.81 , 17.955, 17.95 , 18.029, 17.973,
       18.17 , 18.174, 18.171, 17.985, 18.322, 18.222, 17.85 , 18.324,
       18.221, 18.224, 18.223, 18.235, 18.237, 18.28 , 18.173, 18.308,
       18.236, 18.068, 18.246, 18.18 , 18.226, 18.225, 13.18 , 18.473,
       18.27 , 18.105, 18.369, 18.372, 18.37 , 17.91 , 18.23 , 18.425,
       18.43 , 18.122, 18.025])
```

In [13]:
```
df['Freeboard'].unique()
```

Out[13]:
```
array([ 2.411,  2.813,  2.892,  3.098,  2.991,  3.073,  3.424,  3.13 ,
        3.135,  3.061,  3.092,  3.04 ,  3.142,  3.768,  3.111,  3.63 ,
        3.344,  3.672,  3.675,  4.27 ,  8.995,  3.667,  3.867,  3.732,
        3.785,  3.625,  3.777,  3.922,  3.815,  3.745,  4.13 ,  4.139,
        4.142,  3.825,  9.819,  3.864,  3.882,  3.897,  3.834,  4.097,
        4.081,  4.181,  4.169,  4.074,  4.125,  4.127,  4.202,  4.437,
        4.44 ,  4.072,  4.301,  4.173,  4.105,  3.977,  3.979,  4.438,
        4.429,  4.53 ,  4.282,  4.292,  4.283,  3.905,  3.937,  4.176,
        4.192,  4.19 ,  4.336,  4.266,  4.337,  4.407,  4.267,  4.626,
        4.826,  4.699,  4.5  ,  4.501,  4.376,  4.63 ,  4.84 ,  4.278,
        4.276,  4.631,  4.628,  4.62 ,  4.352,  4.688,  4.28 ,  4.659,
        4.917,  4.918,  5.165,  5.223,  4.525,  4.636,  5.053,  5.055,
        4.945,  5.026,  5.028,  4.542,  4.967,  4.898,  4.57 ,  5.066,
        4.701,  4.922,  4.837,  4.919,  5.018,  5.037,  5.024,  4.97 ,
        4.564,  5.173,  5.174,  5.103,  5.023,  5.37 ,  5.375,  5.378,
        5.433,  5.401,  5.434,  4.731,  4.733,  4.94 ,  5.424,  5.098,
        5.073,  5.519,  5.32 ,  5.294,  5.328,  5.326,  5.224,  5.225,
        5.498,  5.524,  5.226,  5.221,  5.159,  5.351,  5.354,  5.384,
        5.346,  5.228,  5.427,  5.426,  5.425,  5.608,  5.577,  5.477,
        5.624,  5.627,  5.598,  5.726,  5.782,  5.64 ,  5.561,  5.554,
        5.786,  5.798,  5.537,  5.543,  5.729,  7.425,  5.593,  5.331,
        5.33 ,  5.31 ,  5.391,  5.271,  5.369,  5.368,  5.35 ,  5.349,
        5.187,  5.186,  5.329,  5.404,  5.371,  5.484,  5.419,  5.502,
        5.42 ,  5.62 ,  5.717,  5.725,  5.72 ,  5.723,  5.576,  5.323,
        5.422,  5.247,  5.249,  5.728,  5.582,  5.532,  5.579,  6.118,
        5.583,  5.672,  5.63 ,  5.645,  5.647,  5.675,  5.657,  5.676,
        5.677,  5.656,  5.629,  5.526,  5.673,  5.706,  6.045,  5.969,
        5.978,  5.965,  5.625,  6.067,  6.066,  5.48 ,  5.662,  5.587,
        5.641,  5.58 ,  5.732,  5.609,  5.678,  5.586,  5.574,  5.833,
        5.828,  5.83 ,  5.745,  5.483,  6.511,  5.53 ,  6.526, 10.171,
        5.546,  5.56 ,  5.556,  6.029,  5.529,  6.3  ,  5.844,  6.639,
        5.707,  5.99 ,  5.856,  6.482,  6.453,  6.352,  6.504,  6.506,
        6.733,  6.485,  6.578,  6.786,  6.534,  6.585,  6.583,  6.314,
        6.536,  6.516,  6.58 ,  6.551,  6.532,  6.49 ,  6.492,  6.505,
        6.385,  6.538,  6.431,  6.59 ,  6.447,  6.665,  6.255,  6.54 ,
        6.835,  6.64 ,  6.635,  7.036,  7.138])
```

In [14]:
```
df['Dead-weight'].unique()
```

Out[14]:
```
array(['10095', '12274', '14187', ..., '209988', '226371', '250877'],
      dtype=object)
```

In [15]:
```
df['Speed']
```

Out[15]:
```
0       16.3
1       15.2
2       16.1
3       14.7
4       15.2
        ...
1463    17.0
1464    16.7
1465    17.2
1466    17.2
1467    17.5
Name: Speed, Length: 1468, dtype: float64
```

In [16]:
```
df['Power']
```

Out[16]:
```
0        3883
1        3603
2        4413
3        4440
4        4440
        ...
1463    16040
1464    16040
1465    17690
1466    20445
1467    23280
Name: Power, Length: 1468, dtype: int64
```

In [17]:
```
df.describe()
```

Out[17]:

	LoA	Lwl	Breadth	Depth	Summer-draft	Freeboard	Rpm	Speed	L/B	B/T	L/H	
count	1468.000000	1468.000000	1468.000000	1468.000000	1468.000000	1468.000000	1468.000000	1468.000000	1468.000000	1468.000000	1468.000000	1468.
mean	213.649816	207.601465	34.108208	18.671567	13.363719	5.366858	107.644414	16.205245	6.119328	2.566195	11.104636	9848.
std	39.958974	39.840419	6.114442	3.226971	2.488205	0.807869	18.661670	0.550559	1.395707	0.227613	0.733032	3628.
min	113.330000	107.020000	19.400000	10.400000	7.941000	2.411000	12.000000	14.200000	5.291000	0.277000	5.823000	3603.
25%	189.900000	182.470000	32.240000	17.150000	12.149000	5.026000	91.000000	15.900000	5.759000	2.468000	10.505000	7796.
50%	199.980000	195.000000	32.260000	18.500000	12.948000	5.433000	108.000000	16.100000	5.967000	2.545000	11.174500	8880.
75%	228.990000	224.230000	32.270000	20.000000	14.429000	5.675000	116.000000	16.500000	6.317000	2.714250	11.509000	10320.
max	329.950000	321.780000	57.000000	25.400000	18.473000	10.171000	210.000000	18.300000	56.819000	3.551000	23.406000	23280.

Interval pengelompokan data variable (x) dan Variabel (y)

1. Interval pertama yaitu dikelompokan dari Lwl kapal 107-185 meter
2. Interval pertama yaitu dikelompokan dari Lwl kapal 185-221 meter
3. Interval pertama yaitu dikelompokan dari Lwl kapal 221-321 meter

Variabel x antara lain adalah sebagai berikut:

1. Lwl
2. Breadth
3. Depth
4. Summer-draft
5. Speed

Variabel output (y) adalah

1. Power

In [18]:
```
# Import library
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Import dataset
df1=pd.read_csv("ANN_Data_Check_Rev_7b.csv")
df1
# If we want to take the first 5 rows, use this: data_visualization.head()
# If we want to take the last 5 rows, use this: data_visualization.tail()
# if we want to know total row: len(data_visualization)
```

Out[18]:

	L107-L185	B107-B185	H107-H185	T107-T185	V107-V185	P107-P185	L185-L221	B185-B221	H185-H221	T185-T221	V185-V221	P185-P221	L221-L321	B221-B321	H221-H321	T221-T321	V221-V321	P221-P321
0	107.02	19.40	10.4	8.023	16.3	3883	185.00	32.26	17.8	12.530	16.0	8200	221.31	38.0	19.9	13.819	17.0	12240
1	109.08	19.60	11.4	8.632	15.2	3603	185.00	32.26	17.8	12.530	16.3	8200	221.31	38.0	19.9	13.819	16.9	12240
2	127.20	21.20	10.8	7.941	16.1	4413	185.00	32.26	17.8	12.522	16.3	8200	221.31	38.0	19.9	13.819	17.1	12240
3	128.02	23.00	11.5	8.448	14.7	4440	185.00	32.26	17.8	12.530	15.9	8200	221.31	38.0	19.9	13.819	16.8	12268
4	128.35	23.00	11.5	8.540	15.2	4440	185.00	32.26	17.8	12.522	16.2	8200	221.31	38.0	19.9	13.819	16.9	12240
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
484	185.00	32.26	17.8	12.522	16.1	8200	221.31	38.00	19.9	13.819	16.8	12240	296.02	50.0	25.0	18.430	17.0	16040
485	185.00	32.26	17.8	12.530	16.0	8200	221.31	38.00	19.9	13.819	16.9	12240	296.02	50.0	25.0	18.430	16.7	16040
486	185.00	32.26	17.8	12.522	16.1	8200	221.31	38.00	19.9	13.819	16.9	12240	296.90	45.0	24.5	18.068	17.2	17690
487	185.00	32.26	17.8	12.522	16.2	8200	221.31	38.00	19.9	13.819	16.9	12240	309.15	54.0	25.1	18.122	17.2	20445
488	185.00	32.26	17.8	12.522	16.0	8200	221.31	38.00	19.9	13.819	17.1	12240	321.78	57.0	25.1	18.025	17.5	23280

489 rows × 18 columns

In [19]:
```
df1.head()
```

Out[19]:

	L107-L185	B107-B185	H107-H185	T107-T185	V107-V185	P107-P185	L185-L221	B185-B221	H185-H221	T185-T221	V185-V221	P185-P221	L221-L321	B221-B321	H221-H321	T221-T321	V221-V321	P221-P321
0	107.02	19.4	10.4	8.023	16.3	3883	185.0	32.26	17.8	12.530	16.0	8200	221.31	38.0	19.9	13.819	17.0	12240
1	109.08	19.6	11.4	8.632	15.2	3603	185.0	32.26	17.8	12.530	16.3	8200	221.31	38.0	19.9	13.819	16.9	12240
2	127.20	21.2	10.8	7.941	16.1	4413	185.0	32.26	17.8	12.522	16.3	8200	221.31	38.0	19.9	13.819	17.1	12240
3	128.02	23.0	11.5	8.448	14.7	4440	185.0	32.26	17.8	12.530	15.9	8200	221.31	38.0	19.9	13.819	16.8	12268
4	128.35	23.0	11.5	8.540	15.2	4440	185.0	32.26	17.8	12.522	16.2	8200	221.31	38.0	19.9	13.819	16.9	12240

In [20]:
```
df1.tail()
```

Out[20]:

	L107-L185	B107-B185	H107-H185	T107-T185	V107-V185	P107-P185	L185-L221	B185-B221	H185-H221	T185-T221	V185-V221	P185-P221	L221-L321	B221-B321	H221-H321	T221-T321	V221-V321	P221-P321
484	185.0	32.26	17.8	12.522	16.1	8200	221.31	38.0	19.9	13.819	16.8	12240	296.02	50.0	25.0	18.430	17.0	16040
485	185.0	32.26	17.8	12.530	16.0	8200	221.31	38.0	19.9	13.819	16.9	12240	296.02	50.0	25.0	18.430	16.7	16040
486	185.0	32.26	17.8	12.522	16.1	8200	221.31	38.0	19.9	13.819	16.9	12240	296.90	45.0	24.5	18.068	17.2	17690
487	185.0	32.26	17.8	12.522	16.2	8200	221.31	38.0	19.9	13.819	16.9	12240	309.15	54.0	25.1	18.122	17.2	20445
488	185.0	32.26	17.8	12.522	16.0	8200	221.31	38.0	19.9	13.819	17.1	12240	321.78	57.0	25.1	18.025	17.5	23280

Multiple Linear Regression

Setelah melakukan proses korelasi antara variabel-variabel x dan y, dapat diputuskan ada beberapa variabel yang memiliki korelasi nilai yang kuat yang akan dijasi sebagai variabel input untuk proses selanjutnya, yakni:

1. Lwl
2. Breadth
3. Height
4. Summer-draft
5. Speed

In [21]:
```
df
```

Out[21]:

	Ships-name	LoA	Lwl	Breadth	Depth	Summer-draft	Freeboard	Dead-weight	Rpm	Speed	L/B	B/T	L/H	Power
0	COMET	113.33	107.02	19.4	10.4	8.023	2.411	10095	210	16.3	5.516	2.418	10.290	3883
1	MAR DE CORTES	115.33	109.08	19.6	11.4	8.632	2.813	12274	210	15.2	5.565	2.271	9.568	3603
2	BLUE LOTUS	134.04	127.20	21.2	10.8	7.941	2.892	14187	158	16.1	6.000	2.670	11.778	4413
3	ATAYAL STAR	135.01	128.02	23.0	11.5	8.448	3.098	16805	173	14.7	5.566	2.723	11.132	4440
4	COQUIMBO I	134.98	128.35	23.0	11.5	8.540	2.991	17013	173	15.2	5.580	2.693	11.161	4440
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1463	CHINA STEEL ENDEAVOR	299.99	296.02	50.0	25.0	18.430	6.635	209749	84	17.0	5.920	2.713	11.841	16040
1464	CHINA STEEL SUCCESS	299.99	296.02	50.0	25.0	18.430	6.635	209988	84	16.7	5.920	2.713	11.841	16040
1465	CAPE NORMANDY	291.90	296.90	45.0	24.5	18.068	6.490	180646	91	17.2	6.598	2.491	12.118	17690
1466	AWOBASAN MARU	319.58	309.15	54.0	25.1	18.122	7.036	226371	74	17.2	5.725	2.980	12.317	20445
1467	CAPE RALIANCE	329.95	321.78	57.0	25.1	18.025	7.138	250877	76	17.5	5.645	3.162	12.820	23280

1468 rows × 14 columns

In [22]:
```
# Import library
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Import dataset
X_regression = df.loc[:,['Lwl','Breadth','Depth','Summer-draft','Speed']].values
y_regression = df.iloc[:,13].values # df.columns.get_loc("Power")

X_regression
```

Out[22]:
```
array([[107.02 ,  19.4  ,  10.4  ,   8.023,  16.3  ],
       [109.08 ,  19.6  ,  11.4  ,   8.632,  15.2  ],
       [127.2  ,  21.2  ,  10.8  ,   7.941,  16.1  ],
       ...,
       [296.9  ,  45.   ,  24.5  ,  18.068,  17.2  ],
       [309.15 ,  54.   ,  25.1  ,  18.122,  17.2  ],
       [321.78 ,  57.   ,  25.1  ,  18.025,  17.5  ]])
```

In [29]:
```
# Be aware that the scala of each independent variables are different
# Let's do feature scaling using MinMaxScaler()
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler() # another one is StandardScaler()
X_regression = scaler.fit_transform(X_regression)
X_regression
```

Out[29]:
```
array([[0.        , 0.        , 0.        , 0.0077858 , 0.51219512],
       [0.0095921 , 0.00531915, 0.06666667, 0.06560957, 0.24390244],
       [0.09396536, 0.04787234, 0.02666667, 0.        , 0.46341463],
       ...,
       [0.88414975, 0.68085106, 0.94      , 0.96154577, 0.73170732],
       [0.94119017, 0.92021277, 0.98      , 0.966673  , 0.73170732],
       [1.        , 1.        , 0.98      , 0.95746297, 0.80487805]])
```

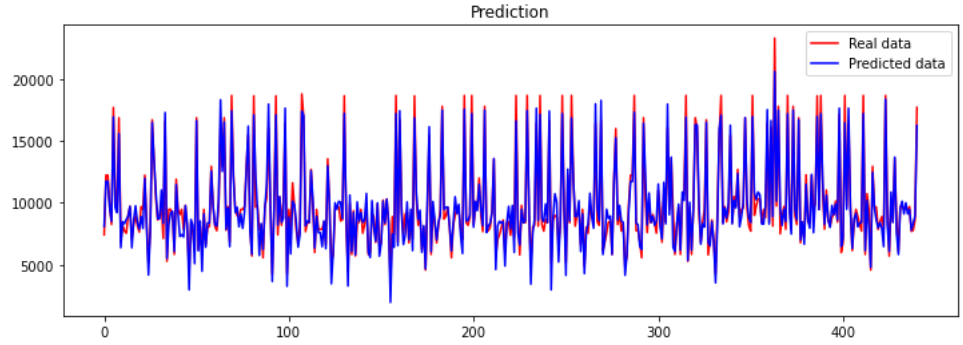
In [30]:
```
# Split Data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_regression, y_regression, test_size = 0.3, random_state = 0)

# Fitting Multiple Linear Regression to Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predict Testing data
y_pred = regressor.predict(X_test)
print("Coefficient: ", regressor.coef_)
print("Intercept: ", regressor.intercept_)
```

```
Coefficient:  [ 3526.08679662  9137.2660714  -6205.0622194  10104.42472568
  4784.60542742]
Intercept:  481.6062799218071
```

In [31]:
```
# Because there are too many variables, we do not need to visualize it, we can directly use metrics evaluation
# Evaluate Metrics
from sklearn import metrics
print("MAE =",metrics.mean_absolute_error(y_test,y_pred))
print("MSE =",metrics.mean_squared_error(y_test,y_pred))
print("R2 =",metrics.r2_score(y_test,y_pred))
```

```
MAE = 626.6025161812488
MSE = 687903.7877685308
R2 = 0.945389260604941
```

In [32]:
```
# Show how good the prediction from visualization
plt.plot(y_test, color = 'red', label = 'Real data')
plt.plot(y_pred, color = 'blue', label = 'Predicted data')
plt.title('Prediction')
plt.rcParams['figure.figsize']=(12, 4)
plt.legend()
plt.show()
```



Polynomial

In [33]:
```
# Fitting Polynomial Regression to dataset
from sklearn.preprocessing import PolynomialFeatures
poly_reg2 = PolynomialFeatures(degree = 2)  # We can change this degree
X_poly = poly_reg2.fit_transform(X_regression)
X_poly
```

Out[33]:
```
array([[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        6.06186142e-05, 3.98784656e-03, 2.62343843e-01],
       [1.00000000e+00, 9.59210281e-03, 5.31914894e-03, ...,
        4.30461578e-03, 1.60023343e-02, 5.94883998e-02],
       [1.00000000e+00, 9.39653567e-02, 4.78723404e-02, ...,
        0.00000000e+00, 0.00000000e+00, 2.14753123e-01],
       ...,
       [1.00000000e+00, 8.84149749e-01, 6.80851064e-01, ...,
        9.24570259e-01, 7.03570072e-01, 5.35395598e-01],
       [1.00000000e+00, 9.41190166e-01, 9.20212766e-01, ...,
        9.34456682e-01, 7.07321705e-01, 5.35395598e-01],
       [1.00000000e+00, 1.00000000e+00, 1.00000000e+00, ...,
        9.16735339e-01, 7.70640927e-01, 6.47828673e-01]])
```

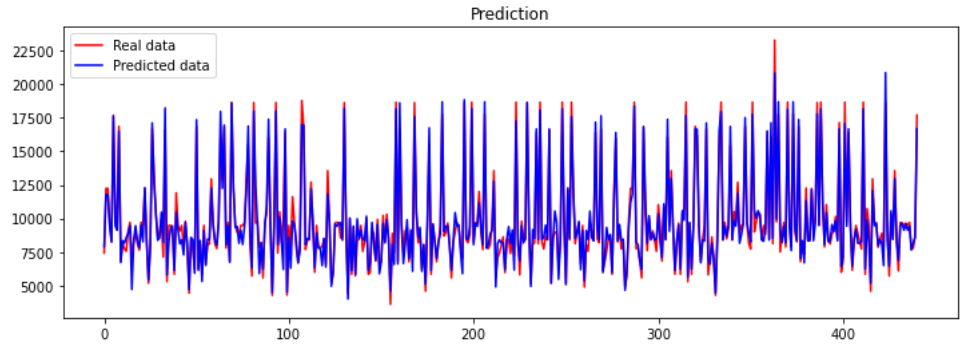
In [34]:
```
# Split Data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_poly, y_regression, test_size = 0.3, random_state = 0)

# Fitting Multiple Linear Regression to Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predict Testing data
y_pred = regressor.predict(X_test)
print("Coefficient: ", regressor.coef_)
print("Intercept: ", regressor.intercept_)
```

```
Coefficient:  [      0.           -3454.47786056    2925.94386996  -17499.10510875
   26029.40312859   -8959.20383325   -3590.00543272   49884.98080342
  -70874.99190738   44643.06096628    6232.74125815  -29149.60417388
  117419.92062934 -125180.21934706    9622.75672358  -14873.71467099
   -3371.3699859    31519.82885128   36753.90236267  -40056.64861663
    9915.18560752]
Intercept:  5720.077630788421
```

In [35]:
```
# Because there are too many variables, we do not need to visualize it, we can directly use metrics evaluation
# Evaluate Metrics
from sklearn import metrics
print("MAE =",metrics.mean_absolute_error(y_test,y_pred))
print("MSE =",metrics.mean_squared_error(y_test,y_pred))
print("R2 =",metrics.r2_score(y_test,y_pred))
```

```
MAE = 504.0318802443081
MSE = 489368.2018147939
R2 = 0.9611504402029418
```

In [36]:
```
# Show how good the prediction from visualization
plt.plot(y_test, color = 'red', label = 'Real data')
plt.plot(y_pred, color = 'blue', label = 'Predicted data')
plt.title('Prediction')
plt.rcParams['figure.figsize']=(12, 4)
plt.legend()
plt.show()
```



In [37]:
```
df.corr()
```

Out[37]:

	LoA	Lwl	Breadth	Depth	Summer-draft	Freeboard	Rpm	Speed	L/B	B/T	L/H	Power
LoA	1.000000	0.998966	0.930943	0.955798	0.963280	0.836179	-0.752547	0.598084	0.168003	-0.211333	0.314932	0.949610
Lwl	0.998966	1.000000	0.930302	0.961096	0.967071	0.845228	-0.763117	0.586964	0.167861	-0.221340	0.302704	0.945816
Breadth	0.930943	0.930302	1.000000	0.896669	0.893815	0.820032	-0.657605	0.599014	0.050488	0.059873	0.261115	0.940666
Depth	0.955798	0.961096	0.896669	1.000000	0.993908	0.915181	-0.761620	0.545834	0.159585	-0.360533	0.078113	0.921209
Summer-draft	0.963280	0.967071	0.893815	0.993908	1.000000	0.881216	-0.746450	0.557058	0.165487	-0.375818	0.111554	0.930105
Freeboard	0.836179	0.845228	0.820032	0.915181	0.881216	1.000000	-0.717198	0.464658	0.122398	-0.281290	-0.027541	0.804277
Rpm	-0.752547	-0.763117	-0.657605	-0.761620	-0.746450	-0.717198	1.000000	-0.328371	-0.163927	0.290029	-0.168457	-0.600508
Speed	0.598084	0.586964	0.599014	0.545834	0.557058	0.464658	-0.328371	1.000000	0.090733	-0.013832	0.228900	0.687732
L/B	0.168003	0.167861	0.050488	0.159585	0.165487	0.122398	-0.163927	0.090733	1.000000	-0.482057	0.067908	0.121189
B/T	-0.211333	-0.221340	0.059873	-0.360533	-0.375818	-0.281290	0.290029	-0.013832	-0.482057	1.000000	0.327193	-0.125768
L/H	0.314932	0.302704	0.261115	0.078113	0.111554	-0.027541	-0.168457	0.228900	0.067908	0.327193	1.000000	0.244990
Power	0.949610	0.945816	0.940666	0.921209	0.930105	0.804277	-0.600508	0.687732	0.121189	-0.125768	0.244990	1.000000

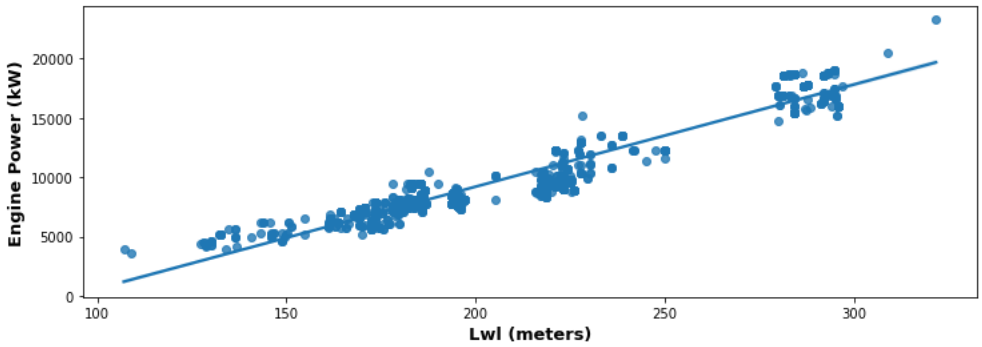
In [31]:
```
df[["Lwl", "Power"]].corr()
```

Out[31]:

	Lwl	Power
Lwl	1.000000	0.945816
Power	0.945816	1.000000

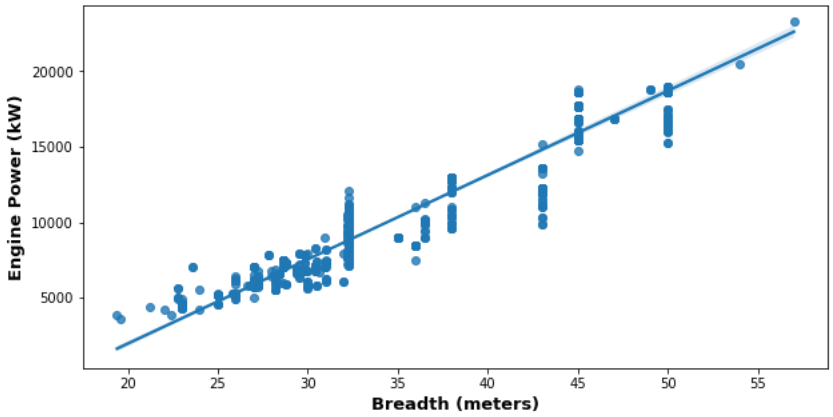
In [32]:
```
import seaborn as sns
# Engine size as potential predictor variable of price
sns.regplot(x="Lwl", y="Power", data=df)
plt.rcParams['figure.figsize']=(10, 5)
plt.xlabel('Lwl (meters)', fontsize=13, fontweight="bold")
plt.ylabel('Engine Power (kW)', fontsize=13, fontweight="bold")
```

Out[32]: Text(0, 0.5, 'Engine Power (kW)')



In [33]:
```
import seaborn as sns
# Engine size as potential predictor variable of price
sns.regplot(x="Breadth", y="Power", data=df)
plt.rcParams['figure.figsize']=(10, 5)
plt.xlabel('Breadth (meters)', fontsize=13, fontweight="bold")
plt.ylabel('Engine Power (kW)', fontsize=13, fontweight="bold")
#plt.title("Korelasi Breadth vs Power", fontsize=13, fontweight="bold")
```

Out[33]: Text(0, 0.5, 'Engine Power (kW)')



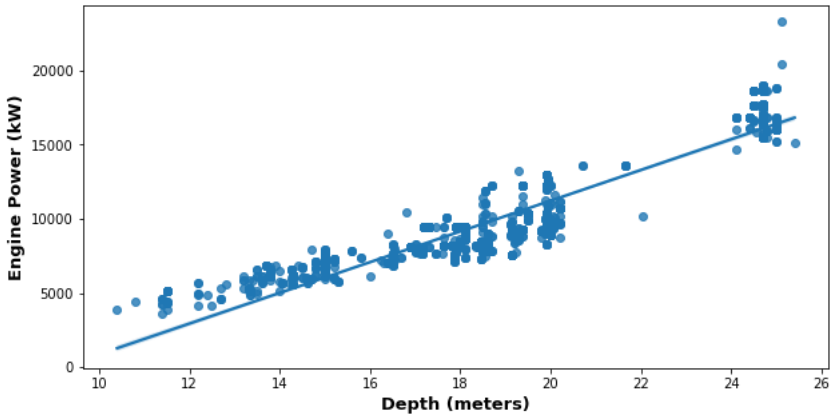
In [34]:
```
df[["Breadth", "Power"]].corr()
```

Out[34]:

	Breadth	Power
Breadth	1.000000	0.940666
Power	0.940666	1.000000

In [35]:
```
import seaborn as sns
# Engine size as potential predictor variable of price
sns.regplot(x="Depth", y="Power", data=df)
plt.rcParams['figure.figsize']=(10, 5)
plt.xlabel('Depth (meters)', fontsize=13, fontweight="bold")
plt.ylabel('Engine Power (kW)', fontsize=13, fontweight="bold")
#plt.title("Korelasi Depth vs Power", fontsize=13, fontweight="bold")
```

Out[35]:
```
Text(0, 0.5, 'Engine Power (kW)')
```



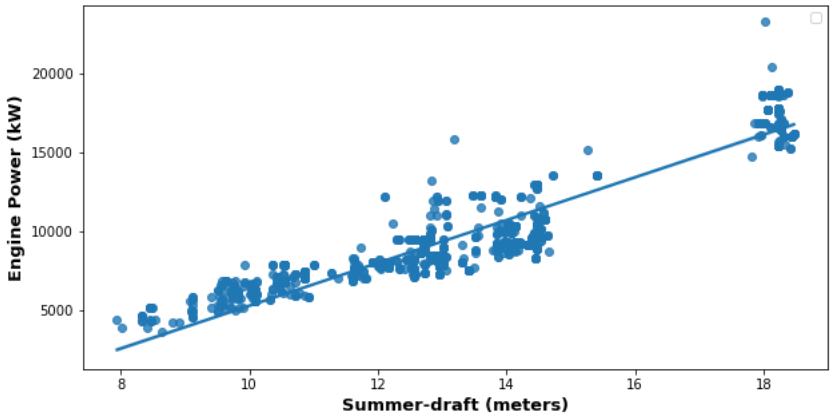
In [36]:
```
df[["Depth", "Power"]].corr()
```

Out[36]:

	Depth	Power
Depth	1.000000	0.921209
Power	0.921209	1.000000

In [37]:
```
import seaborn as sns
# Engine size as potential predictor variable of price
sns.regplot(x="Summer-draft", y="Power", data=df)
plt.rcParams['figure.figsize']=(10, 5)
plt.xlabel('Summer-draft (meters)', fontsize=13, fontweight="bold")
plt.ylabel('Engine Power (kW)', fontsize=13, fontweight="bold")
#plt.title("Korelasi Summer-draft vs Power", fontsize=13, fontweight="bold")
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



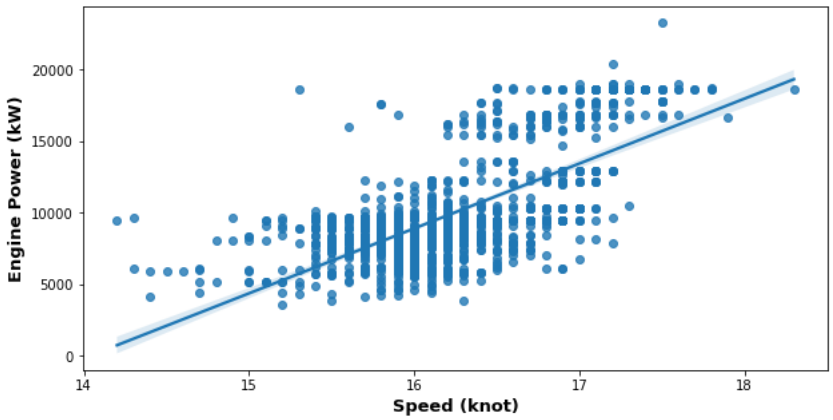
In [38]:
```
df[["Summer-draft", "Power"]].corr()
```

Out[38]:

	Summer-draft	Power
Summer-draft	1.000000	0.930105
Power	0.930105	1.000000

In [39]:
```
import seaborn as sns
# Engine size as potential predictor variable of price
sns.regplot(x="Speed", y="Power", data=df)
plt.rcParams['figure.figsize']=(10, 5)
plt.xlabel('Speed (knot)', fontsize=13, fontweight="bold")
plt.ylabel('Engine Power (kW)', fontsize=13, fontweight="bold")
#plt.title("Korelasi Speed vs Power", fontsize=13, fontweight="bold")
```

Out[39]: Text(0, 0.5, 'Engine Power (kW)')



In [40]:
```
df[["Speed", "Power"]].corr()
```

Out[40]:

	Speed	Power
Speed	1.000000	0.687732
Power	0.687732	1.000000

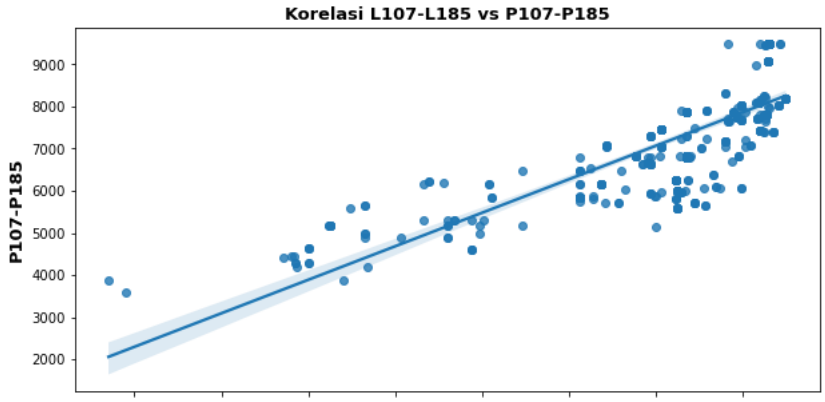
In [41]:
```
df1
```

Out[41]:

	L107-L185	B107-B185	H107-H185	T107-T185	V107-V185	P107-P185	L185-L221	B185-B221	H185-H221	T185-T221	V185-V221	P185-P221	L221-L321	B221-B321	H221-H321	T221-T321	V221-V321	P221-P321
0	107.02	19.40	10.4	8.023	16.3	3883	185.00	32.26	17.8	12.530	16.0	8200	221.31	38.0	19.9	13.819	17.0	12240
1	109.08	19.60	11.4	8.632	15.2	3603	185.00	32.26	17.8	12.530	16.3	8200	221.31	38.0	19.9	13.819	16.9	12240
2	127.20	21.20	10.8	7.941	16.1	4413	185.00	32.26	17.8	12.522	16.3	8200	221.31	38.0	19.9	13.819	17.1	12240
3	128.02	23.00	11.5	8.448	14.7	4440	185.00	32.26	17.8	12.530	15.9	8200	221.31	38.0	19.9	13.819	16.8	12268
4	128.35	23.00	11.5	8.540	15.2	4440	185.00	32.26	17.8	12.522	16.2	8200	221.31	38.0	19.9	13.819	16.9	12240
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
484	185.00	32.26	17.8	12.522	16.1	8200	221.31	38.00	19.9	13.819	16.8	12240	296.02	50.0	25.0	18.430	17.0	16040
485	185.00	32.26	17.8	12.530	16.0	8200	221.31	38.00	19.9	13.819	16.9	12240	296.02	50.0	25.0	18.430	16.7	16040
486	185.00	32.26	17.8	12.522	16.1	8200	221.31	38.00	19.9	13.819	16.9	12240	296.90	45.0	24.5	18.068	17.2	17690
487	185.00	32.26	17.8	12.522	16.2	8200	221.31	38.00	19.9	13.819	16.9	12240	309.15	54.0	25.1	18.122	17.2	20445
488	185.00	32.26	17.8	12.522	16.0	8200	221.31	38.00	19.9	13.819	17.1	12240	321.78	57.0	25.1	18.025	17.5	23280

In [42]:
```
import seaborn as sns
# Correlation variable x and y
sns.regplot(x="L107-L185", y="P107-P185", data=df1)
plt.rcParams['figure.figsize']=(8, 4)
plt.xlabel('L107-L185', fontsize=13, fontweight="bold")
plt.ylabel('P107-P185', fontsize=13, fontweight="bold")
plt.title("Korelasi L107-L185 vs P107-P185", fontsize=13, fontweight="bold")
```

Out[42]: Text(0.5, 1.0, 'Korelasi L107-L185 vs P107-P185')



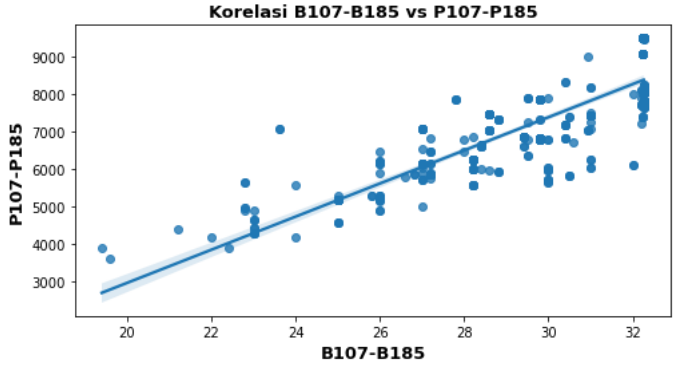
In [43]:
```
df1[["L107-L185","P107-P185"]].corr()
```

Out[43]:

	L107-L185	P107-P185
L107-L185	1.000000	0.793767
P107-P185	0.793767	1.000000

In [44]:
```
# Correlation variable x and y
sns.regplot(x="B107-B185", y="P107-P185", data=df1)
plt.rcParams['figure.figsize']=(8, 4)
plt.xlabel('B107-B185', fontsize=13, fontweight="bold")
plt.ylabel('P107-P185', fontsize=13, fontweight="bold")
plt.title("Korelasi B107-B185 vs P107-P185", fontsize=13, fontweight="bold")
```

Out[44]: Text(0.5, 1.0, 'Korelasi B107-B185 vs P107-P185')



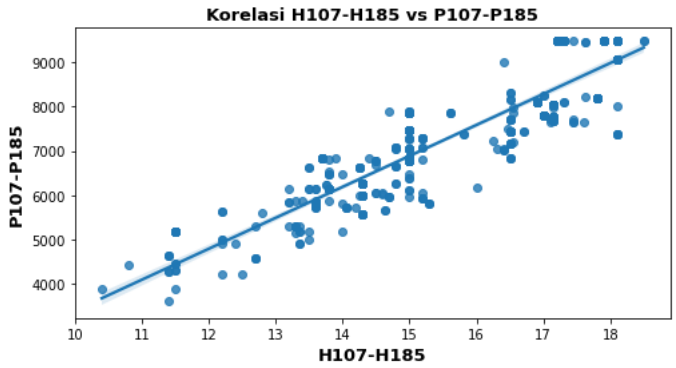
In [45]:
```
df1[["B107-B185","P107-P185"]].corr()
```

Out[45]:

	B107-B185	P107-P185
B107-B185	1.000000	0.843771
P107-P185	0.843771	1.000000

In [46]:
```
# Correlation variable x and y
sns.regplot(x="H107-H185", y="P107-P185", data=df1)
plt.rcParams['figure.figsize']=(8, 4)
plt.xlabel('H107-H185', fontsize=13, fontweight="bold")
plt.ylabel('P107-P185', fontsize=13, fontweight="bold")
plt.title("Korelasi H107-H185 vs P107-P185", fontsize=13, fontweight="bold")
```

Out[46]: Text(0.5, 1.0, 'Korelasi H107-H185 vs P107-P185')



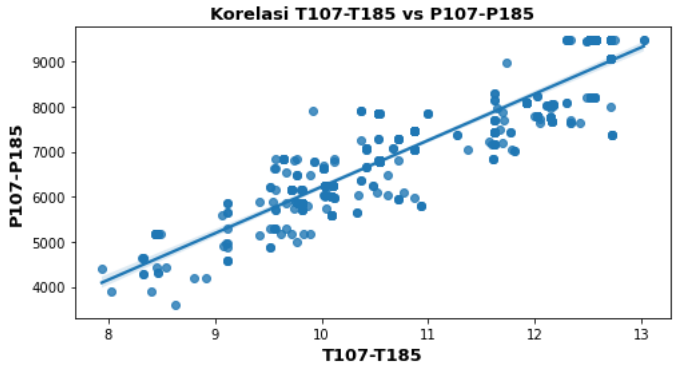
In [47]:
```
df1[["H107-H185","P107-P185"]].corr()
```

Out[47]:

	H107-H185	P107-P185
H107-H185	1.00000	0.91017
P107-P185	0.91017	1.00000

In [48]:
```
# Correlation variable x and y
sns.regplot(x="T107-T185", y="P107-P185", data=df1)
plt.rcParams['figure.figsize']=(8, 4)
plt.xlabel('T107-T185', fontsize=13, fontweight="bold")
plt.ylabel('P107-P185', fontsize=13, fontweight="bold")
plt.title("Korelasi T107-T185 vs P107-P185", fontsize=13, fontweight="bold")
```

Out[48]: Text(0.5, 1.0, 'Korelasi T107-T185 vs P107-P185')



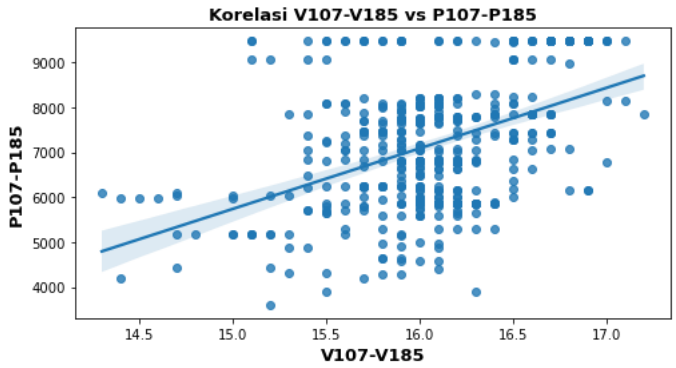
In [49]:
```
df1[["T107-T185","P107-P185"]].corr()
```

Out[49]:

	T107-T185	P107-P185
T107-T185	1.000000	0.910912
P107-P185	0.910912	1.000000

In [50]:
```
# Correlation variable x and y
sns.regplot(x="V107-V185", y="P107-P185", data=df1)
plt.rcParams['figure.figsize']=(8, 4)
plt.xlabel('V107-V185', fontsize=13, fontweight="bold")
plt.ylabel('P107-P185', fontsize=13, fontweight="bold")
plt.title("Korelasi V107-V185 vs P107-P185", fontsize=13, fontweight="bold")
```

Out[50]:
```
Text(0.5, 1.0, 'Korelasi V107-V185 vs P107-P185')
```



In [51]:
```
df1[["V107-V185","P107-P185"]].corr()
```

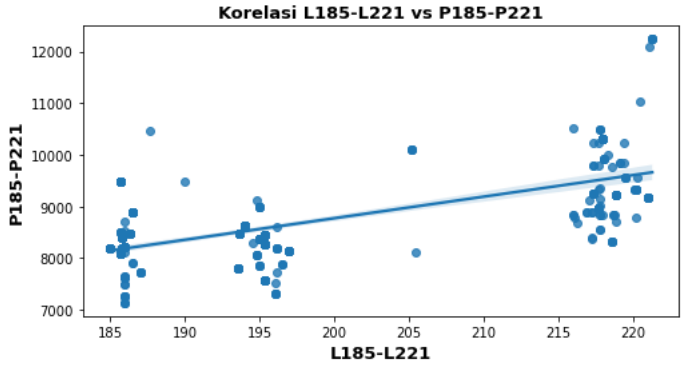
Out[51]:

	V107-V185	P107-P185
V107-V185	1.000000	0.452588
P107-P185	0.452588	1.000000

KELOMPOK KAPAL L185-L221

In [52]:
```
import seaborn as sns
# Correlation variable x and y
sns.regplot(x="L185-L221", y="P185-P221", data=df1)
plt.rcParams['figure.figsize']=(8, 4)
plt.xlabel('L185-L221', fontsize=13, fontweight="bold")
plt.ylabel('P185-P221', fontsize=13, fontweight="bold")
plt.title("Korelasi L185-L221 vs P185-P221", fontsize=13, fontweight="bold")
```

Out[52]:
```
Text(0.5, 1.0, 'Korelasi L185-L221 vs P185-P221')
```



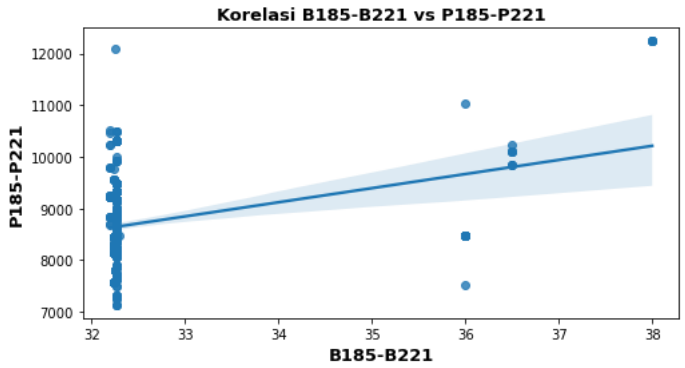
In [53]:
```
df1[["L185-L221","P185-P221"]].corr()
```

Out[53]:

	L185-L221	P185-P221
L185-L221	1.000000	0.650597
P185-P221	0.650597	1.000000

In [54]:
```
# Correlation variable x and y
sns.regplot(x="B185-B221", y="P185-P221", data=df1)
plt.rcParams['figure.figsize']=(8, 4)
plt.xlabel('B185-B221', fontsize=13, fontweight="bold")
plt.ylabel('P185-P221', fontsize=13, fontweight="bold")
plt.title("Korelasi B185-B221 vs P185-P221", fontsize=13, fontweight="bold")
```

Out[54]:
```
Text(0.5, 1.0, 'Korelasi B185-B221 vs P185-P221')
```



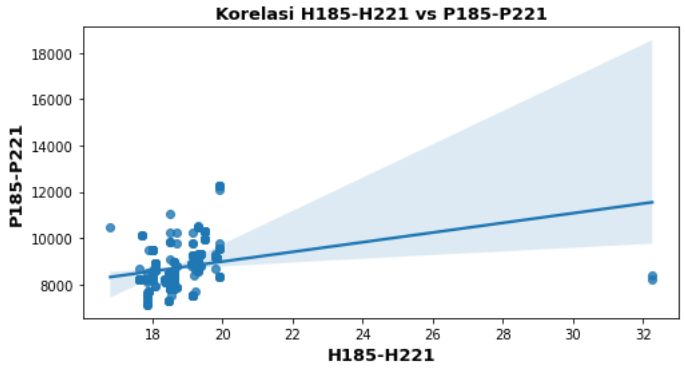
In [55]:
```
df1[["B185-B221","P185-P221"]].corr()
```

Out[55]:

	B185-B221	P185-P221
B185-B221	1.000000	0.328008
P185-P221	0.328008	1.000000

In [56]:
```
# Correlation variable x and y
sns.regplot(x="H185-H221", y="P185-P221", data=df1)
plt.rcParams['figure.figsize']=(8, 4)
plt.xlabel('H185-H221', fontsize=13, fontweight="bold")
plt.ylabel('P185-P221', fontsize=13, fontweight="bold")
plt.title("Korelasi H185-H221 vs P185-P221", fontsize=13, fontweight="bold")
```

Out[56]:
```
Text(0.5, 1.0, 'Korelasi H185-H221 vs P185-P221')
```



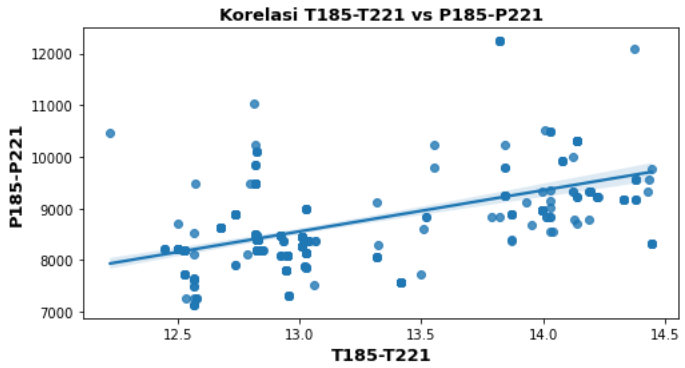
In [57]:
```
df1[["H185-H221","P185-P221"]].corr()
```

Out[57]:

	H185-H221	P185-P221
H185-H221	1.000000	0.267742
P185-P221	0.267742	1.000000

In [58]:
```
# Correlation variable x and y
sns.regplot(x="T185-T221", y="P185-P221", data=df1)
plt.rcParams['figure.figsize']=(8, 4)
plt.xlabel('T185-T221', fontsize=13, fontweight="bold")
plt.ylabel('P185-P221', fontsize=13, fontweight="bold")
plt.title("Korelasi T185-T221 vs P185-P221", fontsize=13, fontweight="bold")
```

Out[58]:
```
Text(0.5, 1.0, 'Korelasi T185-T221 vs P185-P221')
```



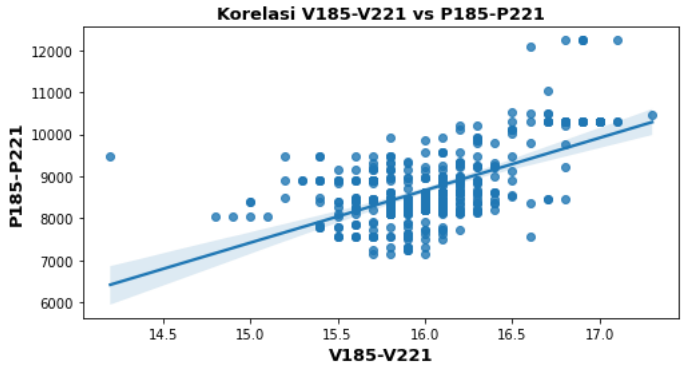
In [59]:
```
df1[["T185-T221","P185-P221"]].corr()
```

Out[59]:

	T185-T221	P185-P221
T185-T221	1.000000	0.552842
P185-P221	0.552842	1.000000

In [60]:
```
# Correlation variable x and y
sns.regplot(x="V185-V221", y="P185-P221", data=df1)
plt.rcParams['figure.figsize']=(8, 4)
plt.xlabel('V185-V221', fontsize=13, fontweight="bold")
plt.ylabel('P185-P221', fontsize=13, fontweight="bold")
plt.title("Korelasi V185-V221 vs P185-P221", fontsize=13, fontweight="bold")
```

Out[60]:
```
Text(0.5, 1.0, 'Korelasi V185-V221 vs P185-P221')
```



In [61]:
```
df1[["V185-V221","P185-P221"]].corr()
```

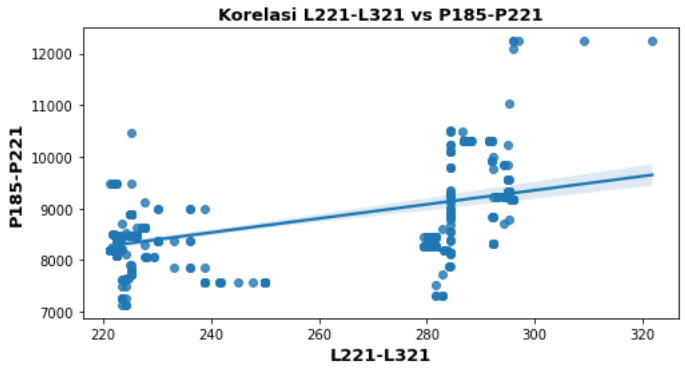
Out[61]:

	V185-V221	P185-P221
V185-V221	1.000000	0.589703
P185-P221	0.589703	1.000000

KELOMPOK KAPAL L221-L321

In [62]:
```
import seaborn as sns
# Correlation variable x and y
sns.regplot(x="L221-L321", y="P185-P221", data=df1)
plt.rcParams['figure.figsize']=(8, 4)
plt.xlabel('L221-L321', fontsize=13, fontweight="bold")
plt.ylabel('P185-P221', fontsize=13, fontweight="bold")
plt.title("Korelasi L221-L321 vs P185-P221", fontsize=13, fontweight="bold")
```

Out[62]: Text(0.5, 1.0, 'Korelasi L221-L321 vs P185-P221')



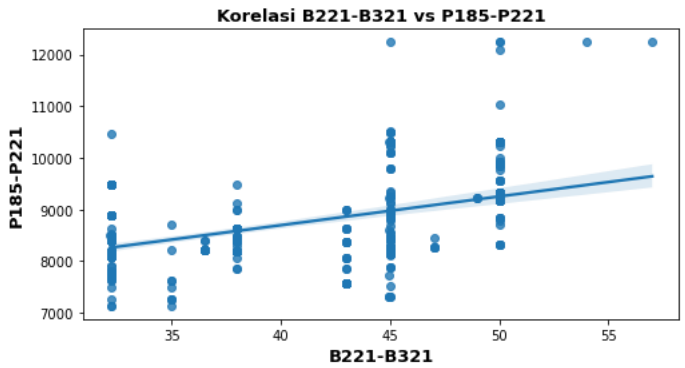
In [63]:
```
df1[["L221-L321","P221-P321"]].corr()
```

Out[63]:

	L221-L321	P221-P321
L221-L321	1.000000	0.939574
P221-P321	0.939574	1.000000

In [64]:
```
# Correlation variable x and y
sns.regplot(x="B221-B321", y="P185-P221", data=df1)
plt.rcParams['figure.figsize']=(8, 4)
plt.xlabel('B221-B321', fontsize=13, fontweight="bold")
plt.ylabel('P185-P221', fontsize=13, fontweight="bold")
plt.title("Korelasi B221-B321 vs P185-P221", fontsize=13, fontweight="bold")
```

Out[64]: Text(0.5, 1.0, 'Korelasi B221-B321 vs P185-P221')



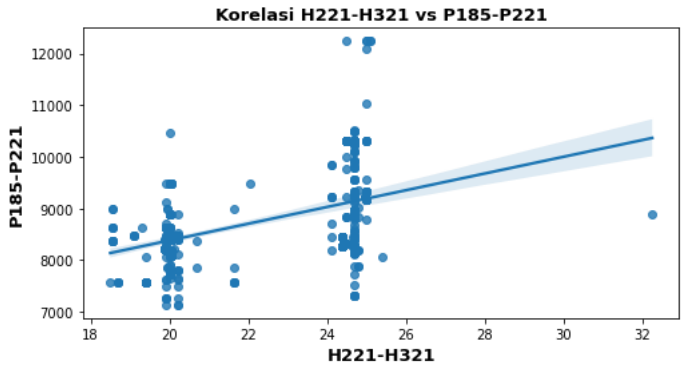
In [65]:
```
df1[["B221-B321","P221-P321"]].corr()
```

Out[65]:

	B221-B321	P221-P321
B221-B321	1.000000	0.890894
P221-P321	0.890894	1.000000

In [66]:
```
# Correlation variable x and y
sns.regplot(x="H221-H321", y="P185-P221", data=df1)
plt.rcParams['figure.figsize']=(8, 4)
plt.xlabel('H221-H321', fontsize=13, fontweight="bold")
plt.ylabel('P185-P221', fontsize=13, fontweight="bold")
plt.title("Korelasi H221-H321 vs P185-P221", fontsize=13, fontweight="bold")
```

Out[66]:
```
Text(0.5, 1.0, 'Korelasi H221-H321 vs P185-P221')
```



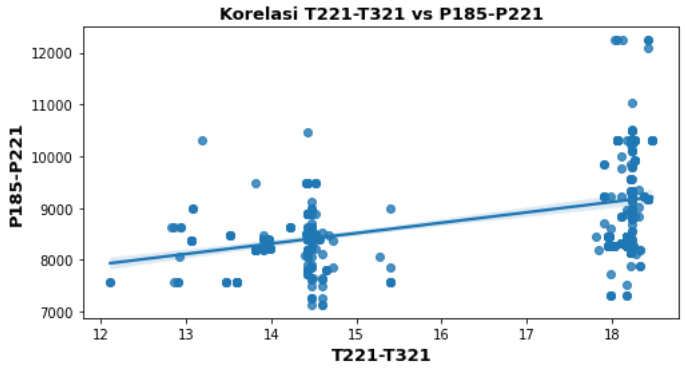
In [67]:
```
df1[["H221-H321","P221-P321"]].corr()
```

Out[67]:

	H221-H321	P221-P321
H221-H321	1.000000	0.897273
P221-P321	0.897273	1.000000

In [68]:
```
# Correlation variable x and y
sns.regplot(x="T221-T321", y="P185-P221", data=df1)
plt.rcParams['figure.figsize']=(8, 4)
plt.xlabel('T221-T321', fontsize=13, fontweight="bold")
plt.ylabel('P185-P221', fontsize=13, fontweight="bold")
plt.title("Korelasi T221-T321 vs P185-P221", fontsize=13, fontweight="bold")
```

Out[68]:
```
Text(0.5, 1.0, 'Korelasi T221-T321 vs P185-P221')
```



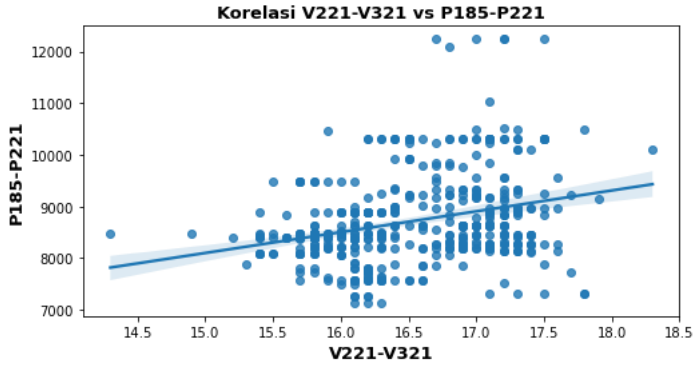
In [69]:
```
df1[["T221-T321","P221-P321"]].corr()
```

Out[69]:

	T221-T321	P221-P321
T221-T321	1.000000	0.907677
P221-P321	0.907677	1.000000

In [70]:
```
# Correlation variable x and y
sns.regplot(x="V221-V321", y="P185-P221", data=df1)
plt.rcParams['figure.figsize']=(8, 4)
plt.xlabel('V221-V321', fontsize=13, fontweight="bold")
plt.ylabel('P185-P221', fontsize=13, fontweight="bold")
plt.title("Korelasi V221-V321 vs P185-P221", fontsize=13, fontweight="bold")
```

Out[70]:
```
Text(0.5, 1.0, 'Korelasi V221-V321 vs P185-P221')
```

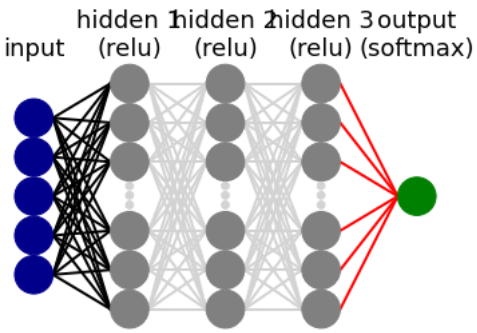


In [71]:
```
df1[["V221-V321","P221-P321"]].corr()
```

Out[71]:

	V221-V321	P221-P321
V221-V321	1.000000	0.809552
P221-P321	0.809552	1.000000

In [72]:
```
# !pip install nnv
from nnv import NNV

layersList = [
    {"title":"input", "units": 5, "color": "darkBlue","edges_color":"black", "edges_width":2},
    {"title":"hidden 1\n(relu)", "units": 30, "edges_width":2},
    {"title":"hidden 2\n(relu)", "units": 25, "edges_width":2},
    {"title":"hidden 3\n(relu)", "units": 15, "edges_color":"red", "edges_width":2},
    {"title":"output\n(softmax)", "units": 1,"color": "green"},
]

NNV(layersList,max_num_nodes_visible=6).render()
plt.rcParams['figure.figsize']=(15, 6)
```



In [73]:
```
! pip install pandoc
```

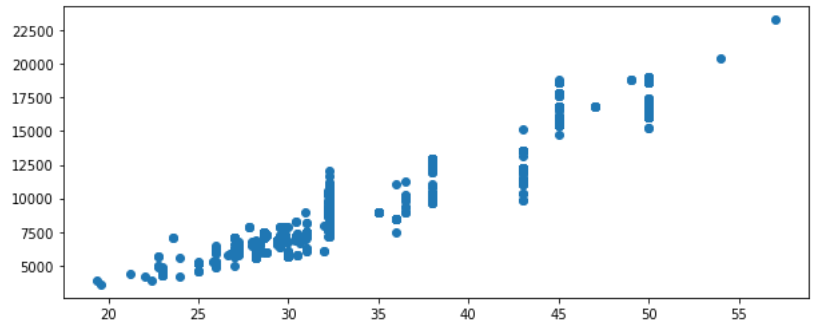
```
Requirement already satisfied: pandoc in c:\programdata\anaconda3\lib\site-packages (2.2)
Requirement already satisfied: plumbum in c:\programdata\anaconda3\lib\site-packages (from pandoc) (1.7.2)
Requirement already satisfied: ply in c:\programdata\anaconda3\lib\site-packages (from pandoc) (3.11)
Requirement already satisfied: pywin32 in c:\programdata\anaconda3\lib\site-packages (from plumbum->pandoc) (228)
```

In [74]:
```
df
```

Out[74]:

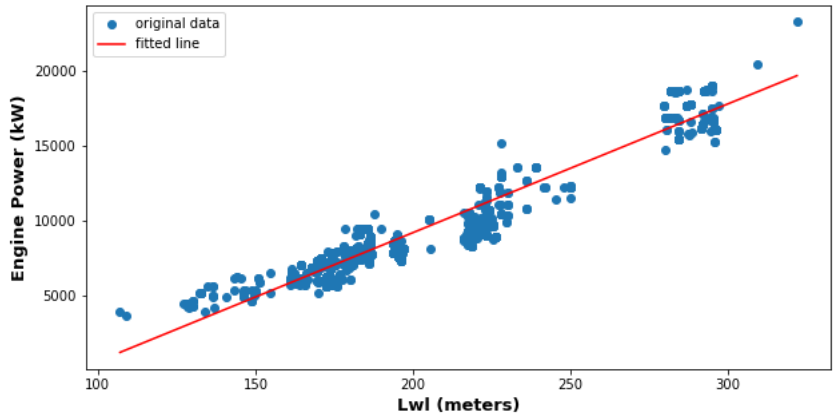
	Ships-name	LoA	Lwl	Breadth	Depth	Summer-draft	Freeboard	Dead-weight	Rpm	Speed	L/B	B/T	L/H	Power
0	COMET	113.33	107.02	19.4	10.4	8.023	2.411	10095	210	16.3	5.516	2.418	10.290	3883
1	MAR DE CORTES	115.33	109.08	19.6	11.4	8.632	2.813	12274	210	15.2	5.565	2.271	9.568	3603
2	BLUE LOTUS	134.04	127.20	21.2	10.8	7.941	2.892	14187	158	16.1	6.000	2.670	11.778	4413
3	ATAYAL STAR	135.01	128.02	23.0	11.5	8.448	3.098	16805	173	14.7	5.566	2.723	11.132	4440
4	COQUIMBO I	134.98	128.35	23.0	11.5	8.540	2.991	17013	173	15.2	5.580	2.693	11.161	4440
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1463	CHINA STEEL ENDEAVOR	299.99	296.02	50.0	25.0	18.430	6.635	209749	84	17.0	5.920	2.713	11.841	16040
1464	CHINA STEEL SUCCESS	299.99	296.02	50.0	25.0	18.430	6.635	209988	84	16.7	5.920	2.713	11.841	16040
1465	CAPE NORMANDY	291.90	296.90	45.0	24.5	18.068	6.490	180646	91	17.2	6.598	2.491	12.118	17690
1466	AWOBASAN MARU	319.58	309.15	54.0	25.1	18.122	7.036	226371	74	17.2	5.725	2.980	12.317	20445
1467	CAPE RALIANCE	329.95	321.78	57.0	25.1	18.025	7.138	250877	76	17.5	5.645	3.162	12.820	23280

In [75]:
```
import numpy as np
import pandas as np
import matplotlib.pyplot as plt
x = df['Breadth']
y = df['Power']
plt.figure(figsize=(10,4))
plt.scatter(x,y)
plt.show()
```



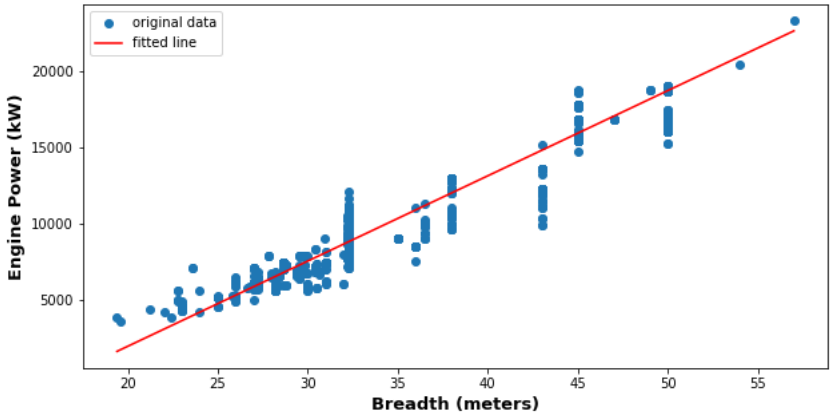
In [76]:
```
import matplotlib.pyplot as plt
from scipy import stats
rgn = df
x = df['Lwl']
y = df['Power']
res = stats.linregress(x, y)
print(f"R-squared: {res.rvalue**2:.6f}")
plt.rcParams['figure.figsize']=(10, 5)
plt.plot(x, y, 'o', label='original data')
plt.plot(x, res.intercept + res.slope*x, 'r', label='fitted line')
plt.xlabel('Lwl (meters)', fontsize=13, fontweight="bold")
plt.ylabel('Engine Power (kW)', fontsize=13, fontweight="bold")
plt.legend()
plt.show()
```

```
R-squared: 0.894567
```



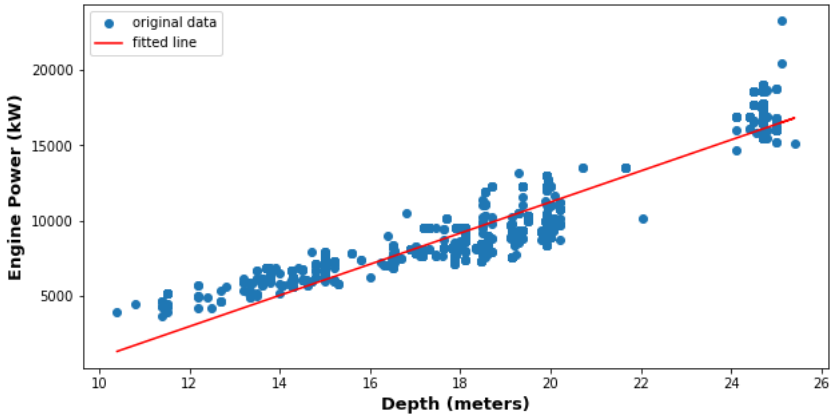
In [77]:
```
import matplotlib.pyplot as plt
from scipy import stats
rgn = df
x = df['Breadth']
y = df['Power']
res = stats.linregress(x, y)
print(f"R-squared: {res.rvalue**2:.6f}")
plt.rcParams['figure.figsize']=(10, 5)
plt.plot(x, y, 'o', label='original data')
plt.plot(x, res.intercept + res.slope*x, 'r', label='fitted line')
plt.xlabel('Breadth (meters)', fontsize=13, fontweight="bold")
plt.ylabel('Engine Power (kW)', fontsize=13, fontweight="bold")
plt.legend()
plt.show()
```

```
R-squared: 0.884852
```



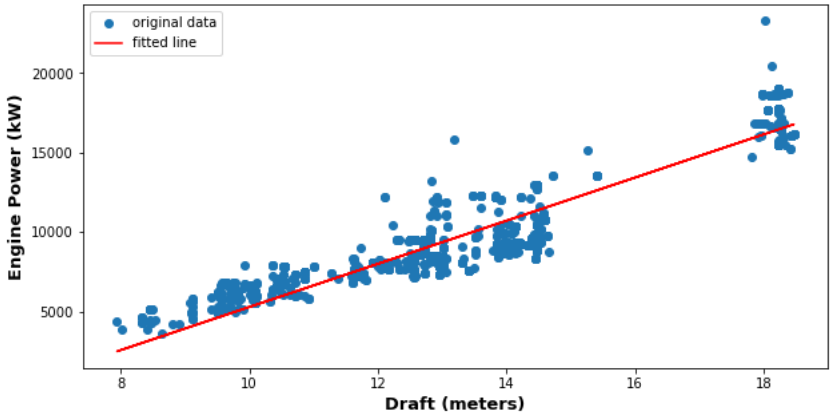
In [78]:
```
import matplotlib.pyplot as plt
from scipy import stats
rgn = df
x = df['Depth']
y = df['Power']
res = stats.linregress(x, y)
print(f"R-squared: {res.rvalue**2:.6f}")
plt.rcParams['figure.figsize']=(10, 5)
plt.plot(x, y, 'o', label='original data')
plt.plot(x, res.intercept + res.slope*x, 'r', label='fitted line')
plt.xlabel('Depth (meters)', fontsize=13, fontweight="bold")
plt.ylabel('Engine Power (kW)', fontsize=13, fontweight="bold")
plt.legend()
plt.show()
```

```
R-squared: 0.848626
```



In [79]:
```
import matplotlib.pyplot as plt
from scipy import stats
rgn = df
x = df['Summer-draft']
y = df['Power']
res = stats.linregress(x, y)
print(f"R-squared: {res.rvalue**2:.6f}")
plt.rcParams['figure.figsize']=(10, 5)
plt.plot(x, y, 'o', label='original data')
plt.plot(x, res.intercept + res.slope*x, 'r', label='fitted line')
plt.xlabel('Draft (meters)', fontsize=13, fontweight="bold")
plt.ylabel('Engine Power (kW)', fontsize=13, fontweight="bold")
plt.legend()
plt.show()
```

```
R-squared: 0.865096
```



In [80]:
```
import matplotlib.pyplot as plt
from scipy import stats
rgn = df
x = df['Speed']
y = df['Power']
res = stats.linregress(x, y)
print(f"R-squared: {res.rvalue**2:.6f}")
plt.rcParams['figure.figsize']=(10, 5)
plt.plot(x, y, 'o', label='original data')
plt.plot(x, res.intercept + res.slope*x, 'r', label='fitted line')
plt.xlabel('Speed (Knot)', fontsize=13, fontweight="bold")
plt.ylabel('Engine Power (kW)', fontsize=13, fontweight="bold")
plt.legend()
plt.show()
```

```
R-squared: 0.472976
```