In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
data = pd.read_csv(r"C:\Users\ASUS\Downloads\aluminum_wire_rod_synthetic.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	CastingTemperature(C)	RollingSpeed(m/min)	CoolingRate(C/s)	ChemicalComposition(Al%)	ChemicalComposition(Cu%)	Chemical(
0	517.738762	2.670609	2.124924	97.462870	1.343437	
1	516.073516	4.669137	1.951931	96.631344	0.341770	
2	514.097405	4.114840	2.440798	97.577301	0.732053	
3	458.507464	4.946501	1.770175	98.004922	1.253575	
4	502.823147	3.941969	0.611059	99.038671	0.430471	

In [4]:
```
data.tail()
```

Out[4]:

	CastingTemperature(C)	RollingSpeed(m/min)	CoolingRate(C/s)	ChemicalComposition(Al%)	ChemicalComposition(Cu%)	Chemi
9995	450.896112	1.591570	2.469588	95.141012	1.371089	
9996	454.236645	2.620234	2.536631	99.230694	0.268305	
9997	496.007223	2.725959	2.837583	97.023051	1.240611	
9998	503.590094	1.449964	1.766031	96.396007	1.087624	
9999	472.481583	4.524516	2.319007	99.419067	1.365180	

In [5]:
```
data.sample(15)
```

Out[5]:

	CastingTemperature(C)	RollingSpeed(m/min)	CoolingRate(C/s)	ChemicalComposition(Al%)	ChemicalComposition(Cu%)	Chemi
8037	517.920005	3.280149	1.821916	98.700120	0.348810	
7861	514.579737	3.032714	0.788460	95.716454	0.696204	
8886	510.780125	3.989576	2.456717	97.810669	0.531648	
922	514.936032	2.834234	0.957876	96.918848	0.338144	
3273	501.616053	1.590271	1.565883	98.225198	1.222651	
9724	484.352095	1.010581	0.719883	96.045009	1.221341	
3085	468.645254	2.218474	2.121610	97.226357	0.506376	
2882	489.539359	2.131989	1.326833	98.737385	0.805712	
5531	480.142943	3.504636	0.779221	99.261509	1.404801	
8302	494.142108	4.753582	2.417318	97.554057	0.448481	
5731	466.124682	2.326465	1.186723	96.066290	1.199724	
2253	501.632395	1.683443	2.172336	98.380392	0.521238	
8881	485.144121	4.271593	2.603102	96.818379	1.284283	
1483	486.078155	1.465562	2.399372	96.979185	1.227539	
1987	485.576334	4.141639	2.805205	96.858726	1.222797	

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 19 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   CastingTemperature(C)        10000 non-null  float64
 1   RollingSpeed(m/min)          10000 non-null  float64
 2   CoolingRate(C/s)             10000 non-null  float64
 3   ChemicalComposition(Al%)     10000 non-null  float64
 4   ChemicalComposition(Cu%)     10000 non-null  float64
 5   ChemicalComposition(Mg%)     10000 non-null  float64
 6   ChemicalComposition(Si%)     10000 non-null  float64
 7   ChemicalComposition(Fe%)     10000 non-null  float64
 8   CastBarEntryTemperature(C)   10000 non-null  float64
 9   EmulsionTemperature(C)       10000 non-null  float64
 10  AgingTemperature(C)          10000 non-null  float64
 11  EmulsionPressure(Bar)        10000 non-null  float64
 12  EmulsionConcentration(%)     10000 non-null  float64
 13  RodQuenchWaterPressure(Bar)  10000 non-null  float64
 14  FirstRollingTemperature(C)   10000 non-null  float64
 15  QuenchingTemperature(C)      10000 non-null  float64
 16  Elongation(%)                10000 non-null  float64
 17  Conductivity(%IACS)          10000 non-null  float64
 18  UTS(Mpa)                     10000 non-null  float64
dtypes: float64(19)
memory usage: 1.4 MB
```

In [7]:
```
data.describe()
```

Out[7]:

	CastingTemperature(C)	RollingSpeed(m/min)	CoolingRate(C/s)	ChemicalComposition(Al%)	ChemicalComposition(Cu%)	Chen
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	
mean	485.100301	2.999822	1.741335	97.236129	0.797193	
std	20.200657	1.160946	0.724717	1.302841	0.403958	
min	450.005874	1.000230	0.500046	95.000018	0.100020	
25%	467.728565	1.989422	1.111139	96.099753	0.442297	
50%	484.980155	3.009923	1.731530	97.220408	0.803133	
75%	502.420843	3.993998	2.370437	98.379422	1.145900	
max	519.994796	4.999955	2.999944	99.499657	1.499440	

In [8]:
```
data.shape
```

Out[8]:
```
(10000, 19)
```

In [9]:
```
data.size
```

Out[9]:
```
190000
```

In [10]:
```
data.isnull().sum()
```

Out[10]:
```
CastingTemperature(C)          0
RollingSpeed(m/min)            0
CoolingRate(C/s)               0
ChemicalComposition(Al%)       0
ChemicalComposition(Cu%)       0
ChemicalComposition(Mg%)       0
ChemicalComposition(Si%)       0
ChemicalComposition(Fe%)       0
CastBarEntryTemperature(C)     0
EmulsionTemperature(C)         0
AgingTemperature(C)            0
EmulsionPressure(Bar)          0
EmulsionConcentration(%)       0
RodQuenchWaterPressure(Bar)    0
FirstRollingTemperature(C)     0
QuenchingTemperature(C)        0
Elongation(%)                  0
Conductivity(%IACS)            0
UTS(Mpa)                       0
dtype: int64
```

In [11]:
```
newdata = data.drop(["ChemicalComposition(Mg%)", "ChemicalComposition(Si%)", "CastBarEntryTemperature(C)", "Agi
newdata.head()
```

Out[11]:

	CastingTemperature(C)	RollingSpeed(m/min)	CoolingRate(C/s)	ChemicalComposition(Al%)	ChemicalComposition(Cu%)	Chemical
0	517.738762	2.670609	2.124924	97.462870	1.343437	
1	516.073516	4.669137	1.951931	96.631344	0.341770	
2	514.097405	4.114840	2.440798	97.577301	0.732053	
3	458.507464	4.946501	1.770175	98.004922	1.253575	
4	502.823147	3.941969	0.611059	99.038671	0.430471	

In [12]:
```
newdata.tail()
```

Out[12]:

	CastingTemperature(C)	RollingSpeed(m/min)	CoolingRate(C/s)	ChemicalComposition(Al%)	ChemicalComposition(Cu%)	Chemi
9995	450.896112	1.591570	2.469588	95.141012	1.371089	
9996	454.236645	2.620234	2.536631	99.230694	0.268305	
9997	496.007223	2.725959	2.837583	97.023051	1.240611	
9998	503.590094	1.449964	1.766031	96.396007	1.087624	
9999	472.481583	4.524516	2.319007	99.419067	1.365180	

In [13]:
```
newdata.sample(15)
```

Out[13]:

	CastingTemperature(C)	RollingSpeed(m/min)	CoolingRate(C/s)	ChemicalComposition(Al%)	ChemicalComposition(Cu%)	Chemi
7412	498.193682	2.680574	0.654158	97.959845	1.111553	
2022	473.556770	2.275129	0.629895	98.737382	0.968975	
7242	476.806900	1.544727	1.242547	97.946052	1.484830	
7649	502.031387	3.024806	2.172294	98.201327	0.417981	
3775	467.525299	3.627351	1.860726	98.277456	0.479656	
2788	463.783116	3.638881	0.526853	96.525821	0.647228	
3682	453.297201	4.423984	2.985406	96.387265	1.101747	
5532	507.461171	1.551542	0.838893	95.471783	1.383262	
65	513.963435	1.620497	2.982469	95.493918	0.166080	
2739	511.179993	2.586878	2.944115	98.418977	1.324801	
3544	509.738383	3.792837	2.140897	95.118957	0.703943	
9609	503.003212	2.232755	1.739349	96.898266	0.295017	
1406	496.686742	1.317943	1.495557	98.502986	1.416908	
3793	484.483321	3.115345	1.246754	95.965321	0.181714	
5154	500.815736	1.560836	1.976137	98.733744	0.408704	

In [15]:
```
newdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   CastingTemperature(C)     10000 non-null  float64
 1   RollingSpeed(m/min)       10000 non-null  float64
 2   CoolingRate(C/s)          10000 non-null  float64
 3   ChemicalComposition(Al%)  10000 non-null  float64
 4   ChemicalComposition(Cu%)  10000 non-null  float64
 5   ChemicalComposition(Fe%)  10000 non-null  float64
 6   EmulsionTemperature(C)    10000 non-null  float64
 7   EmulsionPressure(Bar)     10000 non-null  float64
 8   EmulsionConcentration(%)  10000 non-null  float64
 9   Elongation(%)             10000 non-null  float64
 10  Conductivity(%IACS)       10000 non-null  float64
 11  UTS(Mpa)                  10000 non-null  float64
dtypes: float64(12)
memory usage: 937.6 KB
```

In [16]:
```
newdata.describe()
```

Out[16]:

	CastingTemperature(C)	RollingSpeed(m/min)	CoolingRate(C/s)	ChemicalComposition(Al%)	ChemicalComposition(Cu%)	Chen
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	
mean	485.100301	2.999822	1.741335	97.236129	0.797193	
std	20.200657	1.160946	0.724717	1.302841	0.403958	
min	450.005874	1.000230	0.500046	95.000018	0.100020	
25%	467.728565	1.989422	1.111139	96.099753	0.442297	
50%	484.980155	3.009923	1.731530	97.220408	0.803133	
75%	502.420843	3.993998	2.370437	98.379422	1.145900	
max	519.994796	4.999955	2.999944	99.499657	1.499440	

In [17]:
```
newdata.shape
```

Out[17]:
```
(10000, 12)
```

In [18]:
```
newdata.size
```

Out[18]:
```
120000
```

In [19]:
```
newdata.isnull().sum()
```

Out[19]:
```
CastingTemperature(C)       0
RollingSpeed(m/min)         0
CoolingRate(C/s)            0
ChemicalComposition(Al%)    0
ChemicalComposition(Cu%)    0
ChemicalComposition(Fe%)    0
EmulsionTemperature(C)      0
EmulsionPressure(Bar)       0
EmulsionConcentration(%)    0
Elongation(%)               0
Conductivity(%IACS)         0
UTS(Mpa)                    0
dtype: int64
```

In [20]:
```
X = newdata.drop(["Elongation(%)", "Conductivity(%IACS)", "UTS(Mpa)"], axis="columns")
X.head()
```

Out[20]:

	CastingTemperature(C)	RollingSpeed(m/min)	CoolingRate(C/s)	ChemicalComposition(Al%)	ChemicalComposition(Cu%)	Chemical
0	517.738762	2.670609	2.124924	97.462870	1.343437	
1	516.073516	4.669137	1.951931	96.631344	0.341770	
2	514.097405	4.114840	2.440798	97.577301	0.732053	
3	458.507464	4.946501	1.770175	98.004922	1.253575	
4	502.823147	3.941969	0.611059	99.038671	0.430471	

In [22]:
```
Y=newdata.drop(["CastingTemperature(C)", "RollingSpeed(m/min)", "CoolingRate(C/s)", "ChemicalComposition(Al%)",
Y.head()
```

Out[22]:

	Elongation(%)	Conductivity(%IACS)	UTS(Mpa)
0	79.004379	46.183274	151.816979
1	78.148739	45.231306	149.161207
2	76.589026	43.895326	141.849935
3	66.702291	43.427165	157.580112
4	73.947378	44.451037	163.040135

In [23]:
```
X
```

Out[23]:

	CastingTemperature(C)	RollingSpeed(m/min)	CoolingRate(C/s)	ChemicalComposition(Al%)	ChemicalComposition(Cu%)	Chemi
0	517.738762	2.670609	2.124924	97.462870	1.343437	
1	516.073516	4.669137	1.951931	96.631344	0.341770	
2	514.097405	4.114840	2.440798	97.577301	0.732053	
3	458.507464	4.946501	1.770175	98.004922	1.253575	
4	502.823147	3.941969	0.611059	99.038671	0.430471	
...	...	...	...	...	...	
9995	450.896112	1.591570	2.469588	95.141012	1.371089	
9996	454.236645	2.620234	2.536631	99.230694	0.268305	
9997	496.007223	2.725959	2.837583	97.023051	1.240611	
9998	503.590094	1.449964	1.766031	96.396007	1.087624	
9999	472.481583	4.524516	2.319007	99.419067	1.365180	

10000 rows × 9 columns

In [24]:
```
Y
```

Out[24]:

	Elongation(%)	Conductivity(%IACS)	UTS(Mpa)
0	79.004379	46.183274	151.816979
1	78.148739	45.231306	149.161207
2	76.589026	43.895326	141.849935
3	66.702291	43.427165	157.580112
4	73.947378	44.451037	163.040135
...	...	...	...
9995	70.246691	42.916029	162.476887
9996	70.575571	44.014963	151.391820
9997	75.721106	45.213703	144.549297
9998	77.175560	43.742988	154.972563
9999	70.947047	44.780476	155.620165

10000 rows × 3 columns

In [25]:
```
from sklearn.model_selection import train_test_split
X_train , X_test , y_train , y_test = train_test_split(X , Y , test_size= 0.2)
```

In [26]:
```
len(X_train)
```

Out[26]:
```
8000
```

In [27]:
```
len(X_test)
```

Out[27]:
```
2000
```

In [28]:
```
len(y_train)
```

Out[28]:
```
8000
```

In [29]:
```
len(y_test)
```

Out[29]:
```
2000
```

In [30]:
```
X_train.shape
```

Out[30]:
```
(8000, 9)
```

In [31]:
```
y_train.shape
```

Out[31]:
```
(8000, 3)
```

In [32]:
```
X_train.head()
```

Out[32]:

	CastingTemperature(C)	RollingSpeed(m/min)	CoolingRate(C/s)	ChemicalComposition(Al%)	ChemicalComposition(Cu%)	Chemi
7050	518.030015	2.105835	0.569886	97.209623	1.429350	
1657	472.282236	2.151071	2.458707	98.758419	0.461259	
7238	491.926433	2.442463	1.732410	95.867587	1.386258	
2611	510.638562	3.402625	1.074309	98.899356	1.006325	
6351	474.428203	3.884996	0.596119	95.748079	0.663181	

In [33]:
```
y_train.head()
```

Out[33]:

	Elongation(%)	Conductivity(%IACS)	UTS(Mpa)
7050	77.972482	43.310006	150.007713
1657	74.197752	46.495407	154.708146
7238	72.014071	43.921526	162.574850
2611	76.336956	46.070336	153.700056
6351	71.261941	42.928196	157.370614

In [34]:
```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
```

In [35]:
```
lrmodel = LinearRegression()
lrmodel.fit(X_train , y_train)
```

Out[35]:
```
▾ LinearRegression
LinearRegression()
```

In [36]:
```
lrmodel.predict(X_test)
```

Out[36]:
```
array([[ 68.51397968,  43.26639124, 150.33868295],
       [ 74.06857663,  42.47013111, 150.44708633],
       [ 72.24660263,  43.67170559, 150.6585119 ],
       ...,
       [ 71.98014763,  44.0275805 , 150.42579936],
       [ 68.39684916,  41.58886318, 150.54916922],
       [ 71.90451103,  45.21852225, 150.90154453]])
```

In [38]:
```
lrmodel.predict([[517.738762,2.670609,2.124924,97.462870,1.343437,0.490581,57.784047,2.021540,10.924512]])
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[38]:
```
array([[ 78.62474658,  45.82896893, 150.93429528]])
```

In [39]:
```
lrmodel.predict([[514.097405,4.114840,2.440798,97.577301,0.732053,0.114230,35.371321,1.189000,15.647929]])
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[39]:
```
array([[ 78.41126671,  43.79325602, 150.65577783]])
```

In [40]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
X_train_prediction = lrmodel.predict(X_train)
mae = mean_absolute_error(X_train_prediction , y_train)
mse = mean_squared_error(X_train_prediction , y_train)
rmse = np.sqrt(mse)
r2 = r2_score(X_train_prediction , y_train)
print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Squared Error (MSE): {mse}')
print(f'Root Mean Squared Error (RMSE): {rmse}')
print(f'R-squared (R²): {r2}')
```

```
Mean Absolute Error (MAE): 2.5923326541610954
Mean Squared Error (MSE): 19.88405950063489
Root Mean Squared Error (RMSE): 4.459154572408865
R-squared (R²): -159.1555014886038
```

In [41]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
X_test_prediction = lrmodel.predict(X_test)
mae = mean_absolute_error(X_test_prediction , y_test)
mse = mean_squared_error(X_test_prediction , y_test)
rmse = np.sqrt(mse)
```

```
r2 = r2_score(X_test_prediction , y_test)
print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Squared Error (MSE): {mse}')
print(f'Root Mean Squared Error (RMSE): {rmse}')
print(f'R-squared (R²): {r2}')
```

```
Mean Absolute Error (MAE): 2.5866962099711546
Mean Squared Error (MSE): 19.905639283351924
Root Mean Squared Error (RMSE): 4.46157363307521
R-squared (R²): -165.1712302515526
```

In [42]:
```
dtmodel = DecisionTreeRegressor()
dtmodel.fit(X_train, y_train)
```

Out[42]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [43]:
```
dtmodel.predict(X_test)
```

Out[43]:
```
array([[ 68.50891178,  43.25797233, 146.3174641 ],
       [ 74.70620734,  43.3933103 , 149.1574487 ],
       [ 70.77204412,  44.15217283, 164.1021499 ],
       ...,
       [ 71.64033769,  44.60281898, 163.4417619 ],
       [ 66.19912887,  42.15300446, 144.646028  ],
       [ 73.0731815 ,  46.3172765 , 155.4169527 ]])
```

In [44]:
```
dtmodel.predict([[517.738762,2.670609,2.124924,97.462870,1.343437,0.490581,57.784047,2.021540,10.924512]])
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but DecisionTreeRegressor was fitted with feature names
  warnings.warn(
```

Out[44]:
```
array([[ 79.00437948,  46.18327437, 151.8169788 ]])
```

In [45]:
```
dtmodel.predict([[514.097405,4.114840,2.440798,97.577301,0.732053,0.114230,35.371321,1.189000,15.647929]])
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but DecisionTreeRegressor was fitted with feature names
  warnings.warn(
```

Out[45]:
```
array([[ 76.58902554,  43.89532644, 141.8499348 ]])
```

In [46]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
X_train_prediction = dtmodel.predict(X_train)
mae = mean_absolute_error(X_train_prediction , y_train)
mse = mean_squared_error(X_train_prediction , y_train)
rmse = np.sqrt(mse)
r2 = r2_score(X_train_prediction , y_train)
print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Squared Error (MSE): {mse}')
print(f'Root Mean Squared Error (RMSE): {rmse}')
print(f'R-squared (R²): {r2}')
```

```
Mean Absolute Error (MAE): 0.0
Mean Squared Error (MSE): 0.0
Root Mean Squared Error (RMSE): 0.0
R-squared (R²): 1.0
```

In [47]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
X_test_prediction = dtmodel.predict(X_test)
mae = mean_absolute_error(X_test_prediction , y_test)
mse = mean_squared_error(X_test_prediction , y_test)
rmse = np.sqrt(mse)
r2 = r2_score(X_test_prediction , y_test)
print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Squared Error (MSE): {mse}')
print(f'Root Mean Squared Error (RMSE): {rmse}')
print(f'R-squared (R²): {r2}')
```

```
Mean Absolute Error (MAE): 3.8445204677383296
Mean Squared Error (MSE): 42.86960743992051
Root Mean Squared Error (RMSE): 6.547488636104725
R-squared (R²): 0.015127555354571554
```

In [48]:
```
rfmodel =  RandomForestRegressor()
rfmodel.fit(X_train, y_train)
```

Out[48]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [49]:
```
rfmodel.predict(X_test)
```

Out[49]:
```
array([[ 68.82923784,  43.58864887, 148.31994091],
       [ 74.38146406,  43.23207409, 150.10416497],
       [ 72.00485572,  43.75341134, 149.62845743],
       ...,
       [ 72.06114666,  43.91096723, 150.66605728],
       [ 68.80806798,  42.85511391, 148.77953419],
       [ 71.7006735 ,  45.37466648, 149.02098435]])
```

In [50]:
```
rfmodel.predict([[517.738762,2.670609,2.124924,97.462870,1.343437,0.490581,57.784047,2.021540,10.924512]])
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
```

Out[50]:
```
array([[ 78.72836877,  46.05207978, 151.32642541]])
```

In [51]:
```
rfmodel.predict([[514.097405,4.114840,2.440798,97.577301,0.732053,0.114230,35.371321,1.189000,15.647929]])
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
```

Out[51]:
```
array([[ 77.17853916,  43.81938758, 146.35253339]])
```

In [52]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
X_train_prediction = rfmodel.predict(X_train)
mae = mean_absolute_error(X_train_prediction , y_train)
mse = mean_squared_error(X_train_prediction , y_train)
rmse = np.sqrt(mse)
r2 = r2_score(X_train_prediction , y_train)
print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Squared Error (MSE): {mse}')
print(f'Root Mean Squared Error (RMSE): {rmse}')
print(f'R-squared (R²): {r2}')
```

```
Mean Absolute Error (MAE): 1.0036787323022127
Mean Squared Error (MSE): 2.942534550852804
Root Mean Squared Error (RMSE): 1.715381750763603
R-squared (R²): 0.8572671247496796
```

In [53]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
X_test_prediction = rfmodel.predict(X_test)
mae = mean_absolute_error(X_test_prediction , y_test)
mse = mean_squared_error(X_test_prediction , y_test)
rmse = np.sqrt(mse)
r2 = r2_score(X_test_prediction , y_test)
print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Squared Error (MSE): {mse}')
print(f'Root Mean Squared Error (RMSE): {rmse}')
print(f'R-squared (R²): {r2}')
```

```
Mean Absolute Error (MAE): 2.682137363471233
Mean Squared Error (MSE): 20.962550928138782
Root Mean Squared Error (RMSE): 4.578487842960684
R-squared (R²): -5.617551754445555
```

In [54]:
```
input_data = (514.097405,4.114840,2.440798,97.577301,0.732053,0.114230,35.371321,1.189000,15.647929)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1 , -1)
prediction = lrmodel.predict(input_data_reshaped)
prediction
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[54]:
```
array([[ 78.41126671,  43.79325602, 150.65577783]])
```

In [55]:
```
CastingTemperature = float(input(""))
RollingSpeed = float(input(""))
CoolingRate = float(input(""))
ChemicalCompositionAl = float(input(""))
ChemicalCompositionCu = float(input(""))
ChemicalCompositionFe = float(input(""))
EmulsionTemperature = float(input(""))
EmulsionPressure = float(input(""))
EmulsionConcentration = float(input(""))
```

In [56]:
```
input_data = (CastingTemperature,RollingSpeed,CoolingRate,ChemicalCompositionAl,ChemicalCompositionCu,ChemicalC
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1 , -1)
prediction = lrmodel.predict(input_data_reshaped)
prediction
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[56]:
```
array([[ 78.41126671,  43.79325602, 150.65577783]])
```

In [57]:
```
import pickle
with open("aluminium_model_pickle" , "wb") as file:
     pickle.dump(lrmodel , file)
```

In [58]:
```
import numpy as np
import pickle
with open("aluminium_model_pickle" , "rb") as file:
     lrmodel = pickle.load(file)
```

In [59]:
```
input_data = (514.097405,4.114840,2.440798,97.577301,0.732053,0.114230,35.371321,1.189000,15.647929)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1 , -1)
prediction = lrmodel.predict(input_data_reshaped)
prediction
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[59]:
```
array([[ 78.41126671,  43.79325602, 150.65577783]])
```

In [ ]:

In [ ]: