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
2031	512.100212	4.114984	0.768487	95.415253	1.249494	
7064	486.046415	1.011904	1.861370	97.346390	1.012814	
7177	493.002795	3.020417	0.992113	95.852145	1.368790	
4422	502.335007	2.285010	0.829985	95.334744	1.297320	
9645	472.270501	3.634497	1.887266	97.555677	0.290230	
1996	484.597591	4.987751	1.433072	98.792771	0.360098	
8570	471.282078	3.641708	1.422238	96.787631	0.221290	
9201	496.786162	2.572870	2.167432	98.067174	0.964551	
3732	467.882268	3.042995	1.040646	98.449575	0.171660	
8766	519.467155	1.444343	2.242932	95.517861	0.547453	
8170	475.008006	4.983337	1.675417	95.330183	0.539358	
5110	492.613976	4.172112	1.942683	96.487123	0.174866	
9144	493.350243	4.646407	2.471678	96.558615	0.167844	
9215	489.013899	1.549020	0.956857	97.227616	0.441434	
8051	475.001149	4.200922	2.753840	96.067132	1.039078	

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
9198	463.832756	4.938848	1.319622	99.469287	0.795620	
95	474.497241	2.640713	0.949833	95.745777	0.270540	
9721	498.072065	2.435587	2.943607	98.882857	0.336209	
2469	509.676394	1.113680	1.313695	96.862746	1.191073	
9507	465.339521	3.621071	2.841663	96.286954	0.605596	
3388	509.577046	2.228630	1.371930	95.128693	1.370580	
1753	453.687235	3.485772	2.179596	95.365543	0.655866	
9564	498.764133	3.219259	1.292278	98.450415	0.957299	
9493	467.846824	4.170972	1.028801	98.643933	0.948184	
2436	504.257252	3.378568	1.061124	98.826781	0.228141	
4516	511.305888	3.389589	1.677753	97.438609	1.249815	
6629	475.738294	2.656471	2.395499	98.793396	1.152166	
5113	517.356043	2.191776	1.771043	96.918791	0.624277	
6208	478.434903	1.889639	1.909067	96.477892	0.992041	
3694	488.692848	3.365346	2.011290	98.938673	1.164052	

In [14]:
```
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 [15]:
```
newdata.describe()
```

Out[15]:

	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 [16]:
```
newdata.shape
```

Out[16]:
```
(10000, 12)
```

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

Out[17]:
```
120000
```

In [18]:
```
newdata.isnull().sum()
```

Out[18]:
```
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 [19]:
```
X=newdata.drop(["CastingTemperature(C)", "RollingSpeed(m/min)", "CoolingRate(C/s)", "ChemicalComposition(Al%)",
X.head()
```

Out[19]:

	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 [20]:
```
Y = newdata.drop(["Elongation(%)", "Conductivity(%IACS)", "UTS(Mpa)"], axis="columns")
Y.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 [21]:
```
X
```

Out[21]:

	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 [22]:
```
Y
```

Out[22]:

	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 [23]:
```
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 [24]:
```
len(X_train)
```

Out[24]: 8000

In [25]:
```
len(X_test)
```

Out[25]: 2000

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

Out[26]: 8000

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

Out[27]: 2000

In [28]:
```
X_train.shape
```

Out[28]: (8000, 3)

In [29]:
```
y_train.shape
```

Out[29]: (8000, 9)

In [30]:
```
X_train.head()
```

Out[30]:

	Elongation(%)	Conductivity(%IACS)	UTS(Mpa)
5595	78.678840	45.336680	148.886003
4969	78.022582	45.496135	163.030213
646	75.820067	44.676865	149.717593
4637	76.564725	46.666607	161.883959
6033	75.532052	44.950268	161.215067

In [31]:
```
y_train.head()
```

Out[31]:

	CastingTemperature(C)	RollingSpeed(m/min)	CoolingRate(C/s)	ChemicalComposition(Al%)	ChemicalComposition(Cu%)	Chemi
5595	507.990703	4.421008	0.918457	98.684930	0.674289	
4969	507.463897	4.482023	1.238204	96.124499	1.282339	
646	510.047020	4.008095	1.358859	96.752980	1.070494	
4637	504.664542	2.624789	2.536032	97.939341	1.444780	
6033	513.483767	2.795351	0.968323	97.202745	1.444487	

In [32]:
```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
```

In [33]:
```
lrmodel = LinearRegression()
lrmodel.fit(X_train , y_train)
```

Out[33]:
```
▾ LinearRegression
LinearRegression()
```

In [34]:
```
lrmodel.predict(X_test)
```

Out[34]:
```
array([[491.1331561 ,   2.97447795,   1.88232256, ...,  54.1774526 ,
          2.00655527,  12.6024731 ],
       [465.03525745,   3.00480411,   1.4945368 , ...,  44.7935658 ,
          2.00263861,  12.52027337],
       [499.12196138,   2.99212901,   1.53683956, ...,  34.45437479,
          1.99608291,  12.50395742],
       ...,
       [492.58819874,   2.97117435,   1.55849658, ...,  37.09096122,
          1.9964039 ,  12.56408835],
       [460.20120228,   2.99829521,   1.64756469, ...,  54.03467286,
          2.00763843,  12.5628714 ],
       [487.90187171,   2.97470321,   1.62046985, ...,  42.02197512,
          1.9994544 ,  12.571433  ]])
```

In [35]:
```
prediction = lrmodel.predict([[79.004379,46.183274,151.816979]])
prediction_ = [[f"{num:.2f}" for num in row] for row in prediction]
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[35]:
```
[['513.10', '3.00', '2.11', '97.67', '0.79', '0.30', '58.98', '2.01', '12.53']]
```

In [37]:
```
prediction = lrmodel.predict([[71.362238,44.513813,156.158610]])
prediction_ = [[f"{num:.2f}" for num in row] for row in prediction]
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[37]:
```
[['473.93', '2.97', '1.65', '97.13', '0.81', '0.30', '48.43', '2.00', '12.61']]
```

In [38]:
```
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): 2.268577305512877
Mean Squared Error (MSE): 17.639414466921213
Root Mean Squared Error (RMSE): 4.199930293102638
R-squared (R²): -2458.7523023089393
```

In [39]:
```
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): 2.256234704137137
Mean Squared Error (MSE): 17.504045032625967
Root Mean Squared Error (RMSE): 4.183783578607523
R-squared (R²): -2457.3971791698496
```

In [40]:
```
dtmodel = DecisionTreeRegressor()
dtmodel.fit(X_train, y_train)
```

Out[40]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [41]:
```
dtmodel.predict(X_test)
```

Out[41]:
```
array([[481.8876868 ,   3.54377527,   1.60216575, ...,  49.06833143,
          2.05310458,  10.04780125],
       [453.3142597 ,   1.57371906,   1.17953927, ...,  42.2691258 ,
          2.96176095,  13.90828736],
       [517.0555955 ,   1.74859173,   1.12580022, ...,  42.36852553,
          1.16751689,  12.98437956],
       ...,
       [503.5895542 ,   3.73659972,   0.68876736, ...,  44.64712084,
          1.10704769,  13.33995402],
       [452.2030989 ,   2.6754889 ,   0.73266691, ...,  58.01695   ,
          2.09349788,  12.02168878],
       [491.2363573 ,   4.26556122,   1.29533409, ...,  42.87313612,
          1.18832813,  16.88855592]])
```

In [42]:
```
prediction = dtmodel.predict([[79.004379,46.183274,151.816979]])
prediction_ = [[f"{num:.2f}" for num in row] for row in prediction]
prediction_
```

```
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[42]:
```
[['517.74', '2.67', '2.12', '97.46', '1.34', '0.49', '57.78', '2.02', '10.92']]
```

In [43]:
```
prediction = dtmodel.predict([[71.362238,44.513813,156.158610]])
prediction_ = [[f"{num:.2f}" for num in row] for row in prediction]
prediction_
```

```
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[43]:
```
[['479.17', '1.18', '2.97', '96.47', '0.12', '0.43', '42.31', '2.49', '12.90']]
```

In [44]:
```
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 [45]:
```
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): 3.0514889348682233
Mean Squared Error (MSE): 33.9648430247367
Root Mean Squared Error (RMSE): 5.827936429366462
R-squared (R²): -0.6323788969970702
```

In [46]:
```
rfmodel = RandomForestRegressor()
rfmodel.fit(X_train, y_train)
```

Out[46]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [47]:
```
rfmodel.predict(X_test)
```

Out[47]:
```
array([[491.90796933,   3.10318342,   1.906181  , ...,  55.61182774,
          2.29092171,  11.21807569],
       [458.04128724,   2.56207442,   1.53305869, ...,  42.95525764,
          2.26336186,  12.92943778],
       [502.79098325,   3.24116577,   1.48636661, ...,  37.42069535,
          1.91782837,  11.89953071],
       ...,
       [500.49878041,   3.29876936,   1.08161794, ...,  40.04933959,
          1.50842092,  13.33580036],
       [457.27887991,   3.12238988,   1.44798921, ...,  58.06060792,
          2.02984274,  13.78695477],
       [490.21672967,   2.943932  ,   1.43634694, ...,  40.52973099,
          1.832185  ,  13.20201121]])
```

In [48]:
```
prediction = rfmodel.predict([[79.004379,46.183274,151.816979]])
prediction_ = [[f"{num:.2f}" for num in row] for row in prediction]
prediction_
```

```
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[48]:
```
[['515.13', '3.15', '2.17', '97.27', '1.12', '0.44', '58.26', '2.15', '10.78']]
```

In [49]:
```
prediction = rfmodel.predict([[71.362238,44.513813,156.158610]])
prediction_ = [[f"{num:.2f}" for num in row] for row in prediction]
prediction_
```

```
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[49]:
```
[['468.60', '3.19', '1.74', '96.98', '0.62', '0.37', '48.53', '2.21', '12.85']]
```

In [50]:
```
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): 0.8584005135892219
Mean Squared Error (MSE): 2.644683253531503
Root Mean Squared Error (RMSE): 1.6262482139979442
R-squared (R²): 0.7243370454256639
```

In [51]:
```
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): 2.3091897142144626
Mean Squared Error (MSE): 18.65789484129651
Root Mean Squared Error (RMSE): 4.319478538121993
R-squared (R²): -6.339314479471753
```

In [52]:
```
input_data = (71.362238,44.513813,156.158610)
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_ = [[f"{num:.2f}" for num in row] for row in prediction]
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[52]:
```
[['473.93', '2.97', '1.65', '97.13', '0.81', '0.30', '48.43', '2.00', '12.61']]
```

In [53]:
```
input_data = (71.362238,44.513813,156.158610)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1 , -1)
prediction = dtmodel.predict(input_data_reshaped)
prediction_ = [[f"{num:.2f}" for num in row] for row in prediction]
prediction_
```

```
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[53]:
```
[['479.17', '1.18', '2.97', '96.47', '0.12', '0.43', '42.31', '2.49', '12.90']]
```

In [54]:
```
input_data = (66.702,43.465,157.58)
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_ = [[f"{num:.2f}" for num in row] for row in prediction]
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]:
```
[['450.09', '2.95', '1.37', '96.80', '0.81', '0.30', '41.78', '2.00', '12.64']]
```

In [55]:
```
Elongation = float(input(""))
Conductivity = float(input(""))
UTS = float(input(""))
```

In [56]:
```
input_data = (Elongation,Conductivity,UTS)
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_ = [[f"{num:.2f}" for num in row] for row in prediction]
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]:
```
[['450.09', '2.95', '1.37', '96.80', '0.81', '0.30', '41.78', '2.00', '12.64']]
```

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

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

In [59]:
```
input_data = (66.702,43.465,157.58)
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_ = [[f"{num:.2f}" for num in row] for row in prediction]
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]:
```
[['450.09', '2.95', '1.37', '96.80', '0.81', '0.30', '41.78', '2.00', '12.64']]
```

In [ ]: