A study of micromanufacturing process fingerprints in micro-injection moulding for machine learning and Industry 4.0 applications

International Journal of Advanced Manufacturing Technology

Mert Gülçür (1, *), Ben Whiteside (1)

(1) Centre for Polymer Micro & Nano Technology, Faculty of Engineering and Informatics
University of Bradford, Bradford BD7 1DP, UK

*Corresponding author: Dr. Mert Gülçür, m.gulcur@bradford.ac.uk

In [25]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
#Here the necessary libraries and functions are  imported
```

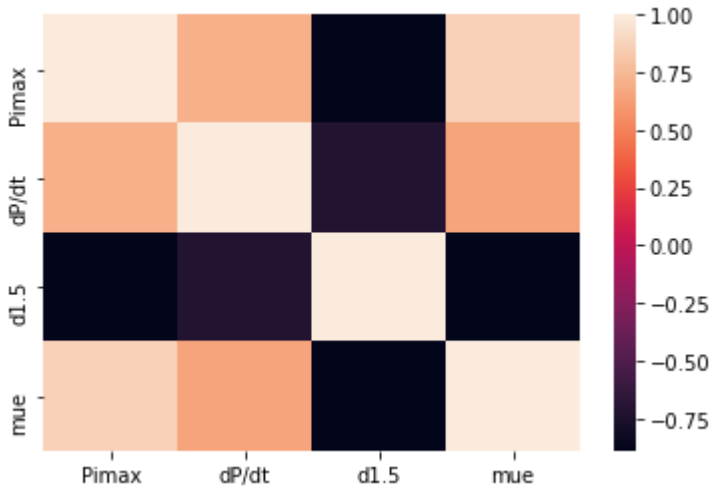
In [26]:
```
data = pd.read_csv('C:/Users/mertg/OneDrive/Desktop/4th_manuscript/data_ana
lysis/python/data/data.csv')
x = data.iloc[:, :-1].values
y = data.iloc[:, 3].values
data.head(10)
#This part of the code assigns different channels of data to specific colum
ns, including the microreplication efficiency (mue).
```

Out[26]:

	Pimax	dP/dt	d1.5	mue
0	1029.14145	13716.86400	283.70188	0.71144
1	1011.58963	12129.46880	266.31811	0.69195
2	1024.95704	14072.60865	275.59651	0.70282
3	1047.56080	9047.76415	257.55696	0.70197
4	1015.72145	10434.27874	263.22405	0.69816
5	1023.01310	12048.50120	264.40052	0.70279
6	1057.52670	9827.04555	253.96898	0.70535
7	1024.95704	14133.32800	270.47897	0.70906
8	963.95106	10216.67752	287.49294	0.68750
9	1045.13050	13966.33946	265.53168	0.69527

In [27]:
```
sns.heatmap(data.corr())
#This is a graphical representation of the correlations between the column
values, as an intermediate step. This gives a rough idea to the user regar
ding the relationships between the columns.
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x27ccb209f70>



In [28]:
```
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, r
andom_state = 0)
#This code splits the process fingerprints and the microreplication efficie
ncy into test and training sets. A test size of 0.2 has been used. Random s
tate = 0 denotes that the selections are  randomised.
```

In [29]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)
#Here, the regression model is formed.
```

Out[29]: LinearRegression()

In [30]:
```
y_pred = regressor.predict(x_test)
print(y_pred)
#This prints the predicted y (microreplication efficiency) values are provi
ded.
```

```
[0.72495623 0.71440293 0.72207568 0.71611538 0.70730036 0.69704735
0.7099012  0.74023935 0.72799826 0.72313149 0.69782908 0.70445839]
```

In [31]:
```
print(regressor.coef_)
#Slope values are calculated for the multiple linear regression equation/mo
del.
```

```
[ 1.35612504e-04 -2.29209446e-07 -9.70099731e-05]
```

In [32]:
```
print(regressor.intercept_)
#Intercept of the  equation.
```

```
0.5880115460043237
```

In [33]:
```
print(y_test)
#Selected test values. Note that these are real  measurements.
```

```
[0.72103 0.71601 0.73739 0.71138 0.70827 0.70282 0.70584 0.75725  0.73407
 0.71738 0.69816 0.70708]
```

In [34]:
```
print(y_pred)
#Predicted values from the model.
```

```
 [0.72495623 0.71440293 0.72207568 0.71611538 0.70730036 0.69704735
 0.7099012  0.74023935 0.72799826 0.72313149 0.69782908 0.70445839]
```