Predictive2,2

January 18, 2025

[1]:
```
import pandas as pd
import torch
```

[2]:
```
torch.cuda.is_available()
```

[2]:
```
True
```

[3]:
```
print(torch.version.cuda)
```

```
11.8
```

[4]:
```
df=pd.read_csv('Battery_Data_Cleaned.csv')
```

[5]:
```
len(df['battery_id'].value_counts())
```

[5]:
```
34
```

[6]:
```
df[df['battery_id']==50]
```

[6]:

	type	ambient_temperature	battery_id	test_id	uid	filename
4316	-1	4	50	0	4319	04319.csv
4317	0	24	50	1	4320	04320.csv
4318	1	4	50	2	4321	04321.csv
4319	0	24	50	3	4322	04322.csv
4320	-1	4	50	4	4323	04323.csv
4321	1	4	50	5	4324	04324.csv
4322	-1	4	50	6	4325	04325.csv
4323	1	4	50	7	4326	04326.csv
4324	-1	4	50	8	4327	04327.csv
4325	1	4	50	9	4328	04328.csv
4326	-1	4	50	10	4329	04329.csv
4327	1	4	50	12	4331	04331.csv
4328	-1	4	50	14	4333	04333.csv
4329	1	4	50	15	4334	04334.csv
4330	-1	4	50	16	4335	04335.csv
4331	1	4	50	17	4336	04336.csv
4332	-1	4	50	18	4337	04337.csv
4333	1	4	50	19	4338	04338.csv

4334	-1	4	50	20	4339	04339.csv
4335	1	4	50	21	4340	04340.csv
4336	-1	4	50	22	4341	04341.csv
4337	0	24	50	23	4342	04342.csv
4338	1	4	50	24	4343	04343.csv
4339	0	24	50	25	4344	04344.csv
4340	-1	4	50	26	4345	04345.csv
4341	1	4	50	27	4346	04346.csv
4342	-1	4	50	28	4347	04347.csv
4343	1	4	50	29	4348	04348.csv
4344	-1	4	50	30	4349	04349.csv
4345	1	4	50	31	4350	04350.csv
4346	-1	4	50	32	4351	04351.csv
4347	1	4	50	33	4352	04352.csv
4348	-1	4	50	34	4353	04353.csv
4349	1	4	50	36	4355	04355.csv
4350	0	24	50	37	4356	04356.csv
4351	-1	4	50	38	4357	04357.csv
4352	1	4	50	39	4358	04358.csv
4353	-1	4	50	40	4359	04359.csv
4354	1	4	50	41	4360	04360.csv
4355	-1	4	50	42	4361	04361.csv
4356	1	4	50	43	4362	04362.csv
4357	-1	4	50	44	4363	04363.csv
4358	1	4	50	45	4364	04364.csv
4359	-1	4	50	46	4365	04365.csv
4360	0	24	50	47	4366	04366.csv
4361	1	4	50	48	4367	04367.csv
4362	0	24	50	49	4368	04368.csv
4363	-1	4	50	50	4369	04369.csv
4364	1	4	50	51	4370	04370.csv
4365	-1	4	50	52	4371	04371.csv
4366	1	4	50	53	4372	04372.csv
4367	-1	4	50	54	4373	04373.csv
4368	1	4	50	55	4374	04374.csv
4369	-1	4	50	56	4375	04375.csv
4370	1	4	50	57	4376	04376.csv
4371	-1	4	50	58	4377	04377.csv
4372	1	4	50	60	4379	04379.csv

	Capacity	Re	Rct
4316	0.622266	0.073173	0.101493
4317	0.622266	0.081193	0.192637
4318	0.622266	0.081193	0.192637
4319	0.622266	0.073414	0.152767
4320	0.974103	0.073414	0.152767
4321	0.974103	0.073414	0.152767

4322	0.956141	0.073414	0.152767
4323	0.956141	0.073414	0.152767
4324	0.936782	0.073414	0.152767
4325	0.936782	0.073414	0.152767
4326	0.032040	0.073414	0.152767
4327	0.032040	0.073414	0.152767
4328	1.292025	0.073414	0.152767
4329	1.292025	0.073414	0.152767
4330	0.858250	0.073414	0.152767
4331	0.858250	0.073414	0.152767
4332	0.864668	0.073414	0.152767
4333	0.864668	0.073414	0.152767
4334	0.855607	0.073414	0.152767
4335	0.855607	0.073414	0.152767
4336	0.858975	0.073414	0.152767
4337	0.858975	0.101905	0.193887
4338	0.858975	0.101905	0.193887
4339	0.858975	0.097746	0.156168
4340	0.423226	0.097746	0.156168
4341	0.423226	0.097746	0.156168
4342	0.875582	0.097746	0.156168
4343	0.875582	0.097746	0.156168
4344	0.887450	0.097746	0.156168
4345	0.887450	0.097746	0.156168
4346	0.073793	0.097746	0.156168
4347	0.073793	0.097746	0.156168
4348	0.453425	0.097746	0.156168
4349	0.453425	0.097746	0.156168
4350	0.453425	0.102644	0.175076
4351	0.263498	0.102644	0.175076
4352	0.263498	0.102644	0.175076
4353	0.000000	0.102644	0.175076
4354	0.000000	0.102644	0.175076
4355	0.165146	0.102644	0.175076
4356	0.165146	0.102644	0.175076
4357	0.778944	0.102644	0.175076
4358	0.778944	0.102644	0.175076
4359	0.091842	0.102644	0.175076
4360	0.091842	0.138584	0.215797
4361	0.091842	0.138584	0.215797
4362	0.091842	0.105789	0.171832
4363	0.245363	0.105789	0.171832
4364	0.245363	0.105789	0.171832
4365	0.245363	0.105789	0.171832
4366	0.245363	0.105789	0.171832
4367	0.245363	0.105789	0.171832
4368	0.245363	0.105789	0.171832

```
4369  0.245363  0.105789  0.171832
4370  0.245363  0.105789  0.171832
4371  0.245363  0.105789  0.171832
4372  0.245363  0.105789  0.171832
```

[7]:
```
df.describe()
```

[7]:
```
                type  ambient_temperature   battery_id      test_id  \
count  7368.000000          7368.000000  7368.000000  7368.000000
mean      0.002443            19.911238    32.213762   166.309718
std       0.865297            11.210718    16.643714   139.771878
min      -1.000000             4.000000     5.000000     0.000000
25%      -1.000000             4.000000    18.000000    54.000000
50%       0.000000            24.000000    36.000000   125.000000
75%       1.000000            24.000000    45.000000   244.250000
max       1.000000            44.000000    56.000000   555.000000

                uid     Capacity           Re          Rct
count  7368.000000  7368.000000  7368.000000  7368.000000
mean   3735.133415     0.824926     0.077739     0.125128
std    2190.232696     0.250283     0.022584     0.044834
min       1.000000     0.000000     0.026691     0.038781
25%    1842.750000     0.775098     0.060875     0.084685
50%    3686.500000     0.894803     0.074693     0.118383
75%    5603.250000     0.986519     0.095817     0.158926
max    7565.000000     1.292025     0.142128     0.238124
```

1 Creation of RUL target values

[8]:
```
import pandas as pd

df = df.reset_index()

# Assign RUL for each battery
for battery in df['battery_id'].unique():
    # fiter 'battery_id'
    group = df[df['battery_id'] == battery]

    # Calculate RUL for each battery id (assigning values from 1 to 0, linealy)
    group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -␣
 ↪group.index.min())

    # update values in the original dataframe
    df.loc[group.index, 'RUL'] = group['RUL']
```

```
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
/tmp/ipykernel_2230/829729603.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  group['RUL'] = 1 - (group.index - group.index.min()) / (group.index.max() -
group.index.min())
```

[9]:
```
df.head()
```

[9]:

	index	type	ambient_temperature	battery_id	test_id	uid	filename
0	0	-1	4	47	0	1	00001.csv
1	1	0	24	47	1	2	00002.csv
2	2	1	4	47	2	3	00003.csv
3	3	0	24	47	3	4	00004.csv
4	4	-1	4	47	4	5	00005.csv

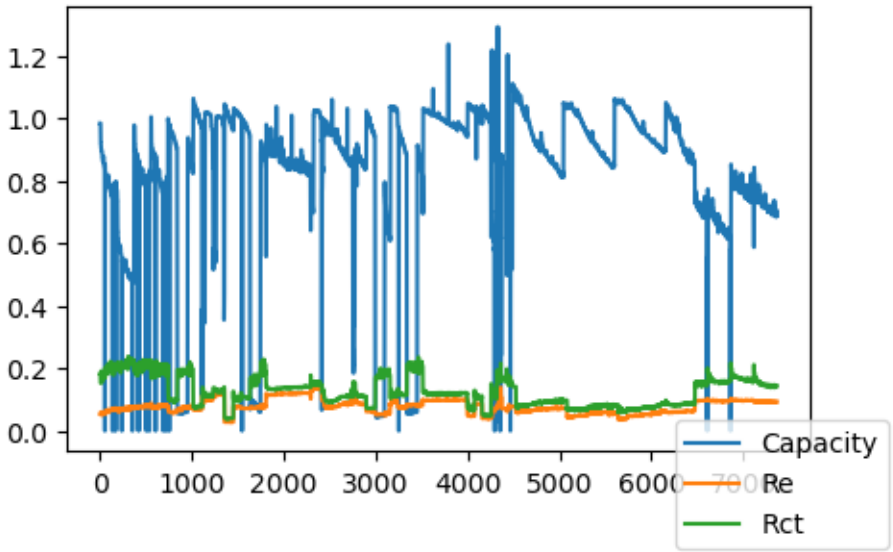
	Capacity	Re	Rct	RUL
0	0.983689	0.054543	0.183130	1.000000
1	0.983689	0.054543	0.183130	0.994536
2	0.983689	0.054543	0.183130	0.989071
3	0.983689	0.051825	0.152493	0.983607
4	0.925990	0.051825	0.152493	0.978142

[10]:
```
df.isnull().sum()
```

[10]:
```
index                  0
type                   0
ambient_temperature    0
battery_id             0
test_id                0
uid                    0
filename               0
Capacity               0
Re                     0
Rct                    0
RUL                    0
dtype: int64
```

[11]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(5,3))
for each in df.drop(['RUL','index','type','battery_id','test_id','uid',␣
 ↪'filename', 'ambient_temperature'], axis=1):
    plt.plot(df[each], label=each)
plt.legend(loc="best",bbox_to_anchor=(0.8,0.1))
```

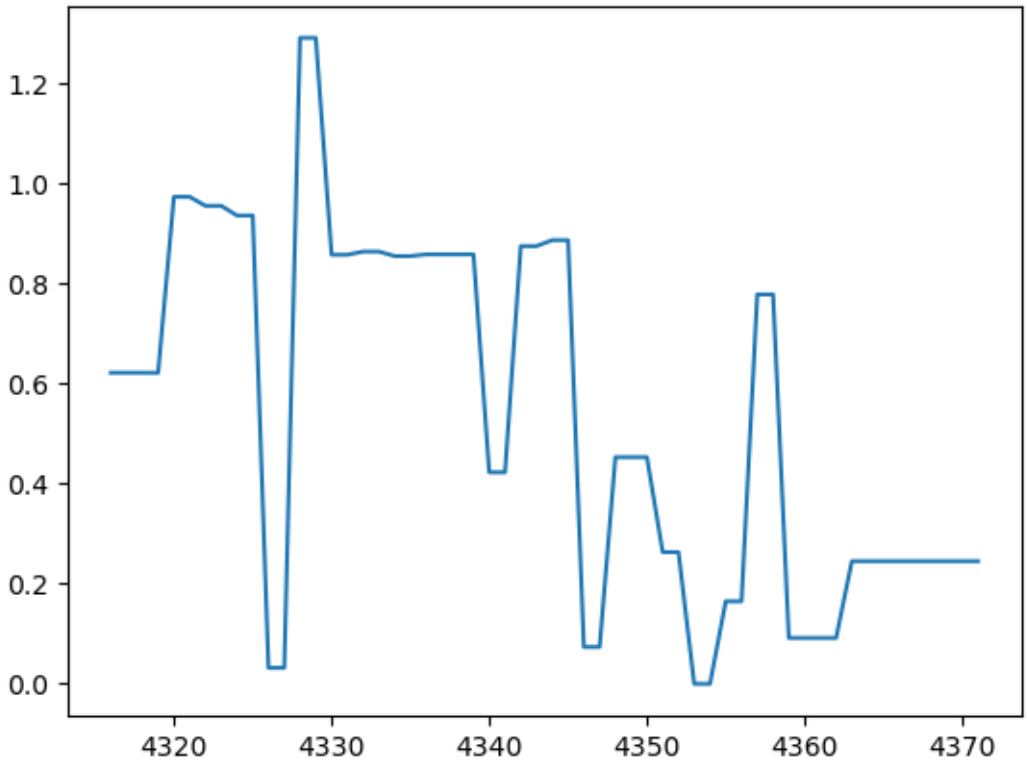
[11]:
```
<matplotlib.legend.Legend at 0x7f3863289d20>
```



1.0.1 Capacity for battery_id = 50

[12]:
```
plt.plot(df['Capacity'][4316:4372])
```

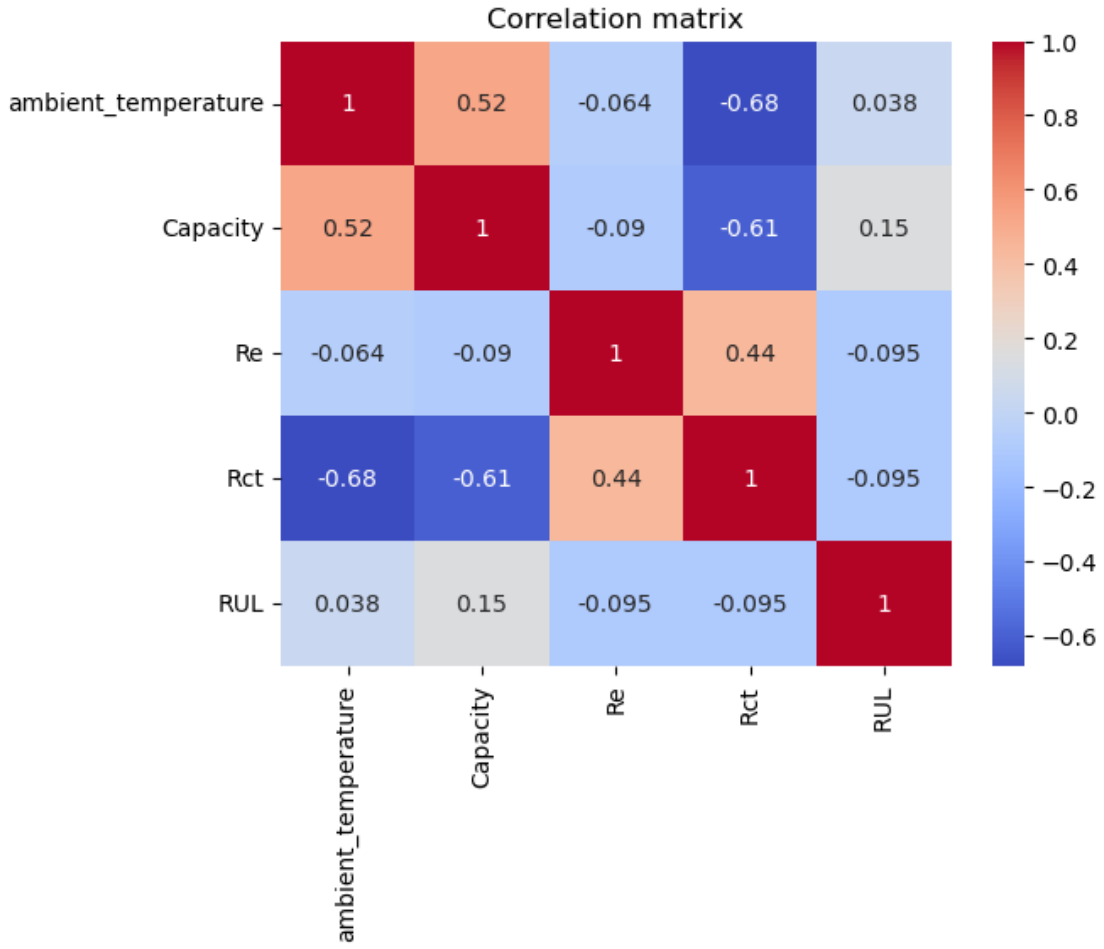
[12]:
```
[<matplotlib.lines.Line2D at 0x7f38631d5d50>]
```



[13]:
```
# plt.figure(figsize=(5,3))
# for each in df:
#      plt.plot(df[each], label=each)
# plt.legend(loc="best",bbox_to_anchor=(0.8,0.1))
```

[14]:
```
# import seaborn as sns
# columns=df.columns

# plt.figure(figsize=(17,15))
# for i, col in enumerate(columns,1):
#     plt.subplot(5,3, i)
#     sns.histplot(df[col], kde=True)
#     plt.title(f'{col} distribution')
# # plt.tight_layout()
# plt.show()
```

[15]:
```
import seaborn as sns
import matplotlib.pyplot as plt
df=df.drop(['index','uid','filename','battery_id','test_id','type'], axis=1)
# Filter only numeric values
df_numeric = df.select_dtypes(include=['float64', 'int64'])

# correlation matrix
correlation = df_numeric.corr()

#print(correlation)

sns.heatmap(correlation, cmap='coolwarm', annot=True)
plt.title('Correlation matrix')
plt.show()
```



2 Removing outliers

[16]:
```
import pandas as pd
# Analize each column


for each in df.columns:
    data = df[each]

    # Calculate IQR
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1

    # limits
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Identify outlier indices
    outliers_index = data[(data < lower_bound) | (data > upper_bound)].index
    #to delete outliers
    # df_cleaned = df_cleaned[(df_cleaned[each] >= lower_bound) &␣
 ↪(df_cleaned[each] <= upper_bound)]

    # Show details
    print(f"Column: {each}")
    print(f"Number of outliers: {len(outliers_index)}")

    # Analyze if the outliers are associated with 'fail'
    outliers_fail = df.loc[outliers_index, 'RUL']
    print("'fail' outliers distribution:")
    print(outliers_fail.value_counts())
    print("-" * 40)
```

```
Column: ambient_temperature
Number of outliers: 0
'fail' outliers distribution:
Series([], Name: count, dtype: int64)
----------------------------------------
Column: Capacity
Number of outliers: 582
'fail' outliers distribution:
RUL
0.726776    4
0.000000    4
0.721311    4
1.000000    4
```

```
0.103825    4
           ..
0.956790    1
0.950617    1
0.944444    1
0.938272    1
0.017857    1
Name: count, Length: 317, dtype: int64
----------------------------------------
Column: Re
Number of outliers: 0
'fail' outliers distribution:
Series([], Name: count, dtype: int64)
----------------------------------------
Column: Rct
Number of outliers: 0
'fail' outliers distribution:
Series([], Name: count, dtype: int64)
----------------------------------------
Column: RUL
Number of outliers: 0
'fail' outliers distribution:
Series([], Name: count, dtype: int64)
----------------------------------------
```

[17]:
```python
df = df.copy()

columns_to_process = ['Capacity']

for column in columns_to_process:
    data = df[column]

    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    df = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

for column in columns_to_process:
    data = df[column]
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
```

```
    outliers_index = data[(data < lower_bound) | (data > upper_bound)].index
    print(f"Column: {column}")
    print(f"Number of outliers remaining: {len(outliers_index)}")
```

```
Column: Capacity
Number of outliers remaining: 205
```

[18]:
```
df
```

[18]:

	ambient_temperature	Capacity	Re	Rct	RUL
0	4	0.983689	0.054543	0.183130	1.000000
1	24	0.983689	0.054543	0.183130	0.994536
2	4	0.983689	0.054543	0.183130	0.989071
3	24	0.983689	0.051825	0.152493	0.983607
4	4	0.925990	0.051825	0.152493	0.978142
…	…	…	…	…	…
7363	24	0.699483	0.092405	0.144011	0.015936
7364	4	0.703166	0.092405	0.144011	0.011952
7365	4	0.703166	0.092405	0.144011	0.007968
7366	4	0.688516	0.092405	0.144011	0.003984
7367	4	0.688516	0.092405	0.144011	0.000000

[6786 rows x 5 columns]

2.0.1 Threshold for Regression

For the regression threshold I selected the Mean strategy. In this case, the model always predicts the average value of the targets. In this case the baseline is 0.084

[19]:
```
df['RUL'].mean()
```

[19]:
```
0.5018118259356504
```

[20]:
```
from sklearn.metrics import mean_squared_error as mse
y_pred = df['RUL']-df['RUL']+df['RUL'].mean()

mse(df['RUL'],y_pred)
```

[20]:
```
0.08689110333886994
```

[21]:
```
df.index=range(0,df.shape[0])
```

3 Spitting the data

[22]:
```
x=df.drop('RUL',axis=1)
y=df['RUL']
```

[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,␣
 ↪random_state=42)#), stratify=y)#x
```

[24]:
```
from sklearn.preprocessing import MinMaxScaler

mms= MinMaxScaler()

x_train_scaled=pd.DataFrame(mms.fit_transform(x_train),columns=x_train.columns)
x_test_scaled=pd.DataFrame(mms.transform(x_test),columns=x_test.columns)
```

[ ]:
```

```

[25]:
```
# from sklearn import decomposition
# import numpy as np
# pca= decomposition.PCA(n_components=x_train_scaled.columns.size)

# pca.fit(x_train_scaled)

# plt.plot(range(1,x_train_scaled.columns.size+1), np.cumsum(pca.
 ↪explained_variance_ratio_))
# plt.grid()
# ###################################
# pca= decomposition.PCA(n_components=0.95) #preserving 95% of the information␣
 ↪#variance = quantity of information that each component can explain
# df_pca = pca.fit_transform(x_train_scaled)

# num_components = pca.n_components_
# print(f"Number of components: {num_components}")
```

[26]:
```
# pca=decomposition.PCA(n_components=4)

# x_train_pca=pca.fit_transform(x_train_scaled)
# x_test_pca=pca.transform(x_test_scaled)
# #x_pca=pd.DataFrame(x_pca,columns=['pca1','pca2','pca3','pca4','pca5','pca6'])
# x_train_pca, y
```

4 Training

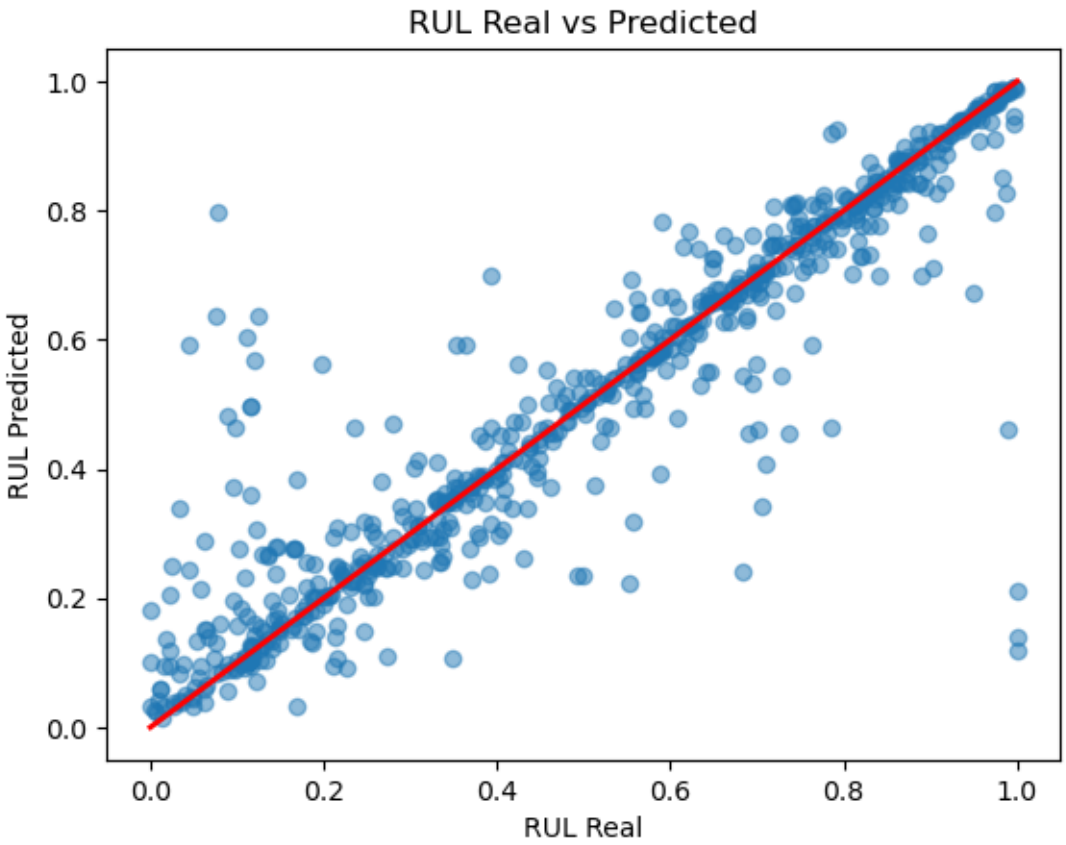
5 KNeighborsRegressor

[27]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

from sklearn.neighbors import KNeighborsRegressor

model = KNeighborsRegressor(n_neighbors=5)
model.fit(x_train_scaled, y_train)
y_pred = model.predict(x_test_scaled)


# predictions
y_pred_scaled = model.predict(x_test_scaled)
```

[28]:
```
#  Metrics
# mae = mean_absolute_error(y_test, y_pred_scaled)
# rmse = mean_squared_error(y_test, y_pred_scaled, squared=False)
# r2 = r2_score(y_test, y_pred_scaled)

mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error (MSE): {mse:.3f}")

mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error (MAE): {mae:.2f}")

r2 = r2_score(y_test, y_pred)
print(f"R² Score: {r2:.2f}")
```

```
Mean Squared Error (MSE): 0.013
Mean Absolute Error (MAE): 0.05
R² Score: 0.85
```

[29]:
```
import matplotlib.pyplot as plt

# Dispersión real vs predicho
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()],
         [y_test.min(), y_test.max()], color='red', linewidth=2)
plt.xlabel("RUL Real")
plt.ylabel("RUL Predicted")
plt.title("RUL Real vs Predicted")
plt.show()
```



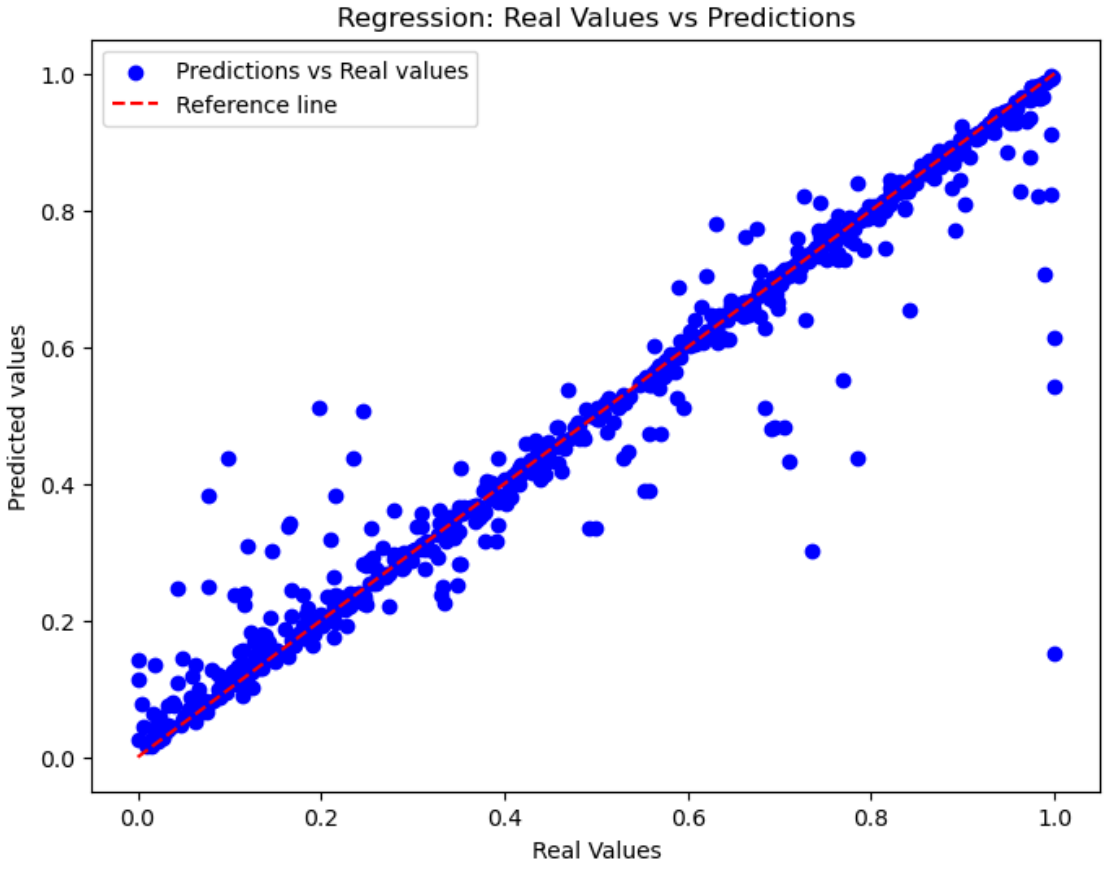
The best model

6 RandomForestRegressor

[30]:
```
import torch
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns


rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(x_train_scaled, y_train)


y_pred = rf_model.predict(x_test_scaled)


mse = mean_squared_error(y_test, y_pred)
```

```python
print(f"Mean Squared Error (MSE): {mse:.3f}")

mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error (MAE): {mae:.2f}")

r2 = r2_score(y_test, y_pred)
print(f"R² Score: {r2:.2f}")


plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, color='blue', label="Predictions vs Real values")
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red',␣
↪linestyle='--', label="Reference line")
plt.xlabel("Real Values")
plt.ylabel("Predicted values")
plt.title("Regression: Real Values vs Predictions")
plt.legend()
plt.show()
```

```
Mean Squared Error (MSE): 0.005
Mean Absolute Error (MAE): 0.03
R² Score: 0.95
```



[31]:
```
import joblib

# Guardar el modelo entrenado
joblib.dump(rf_model, 'random_forest_model.pkl')
```

[31]:
```
['random_forest_model.pkl']
```

[32]:
```
# from sklearn.ensemble import RandomForestRegressor
# from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# # Crear y entrenar el modelo
# rf_model = RandomForestRegressor(
#     n_estimators=300,
#     min_samples_split=10,
#     min_samples_leaf=4,
#     max_depth=10,
#     random_state=42
# )
# # rf_model.fit(x_train_pca, y_train)
# rf_model.fit(x_train_scaled, y_train)
```

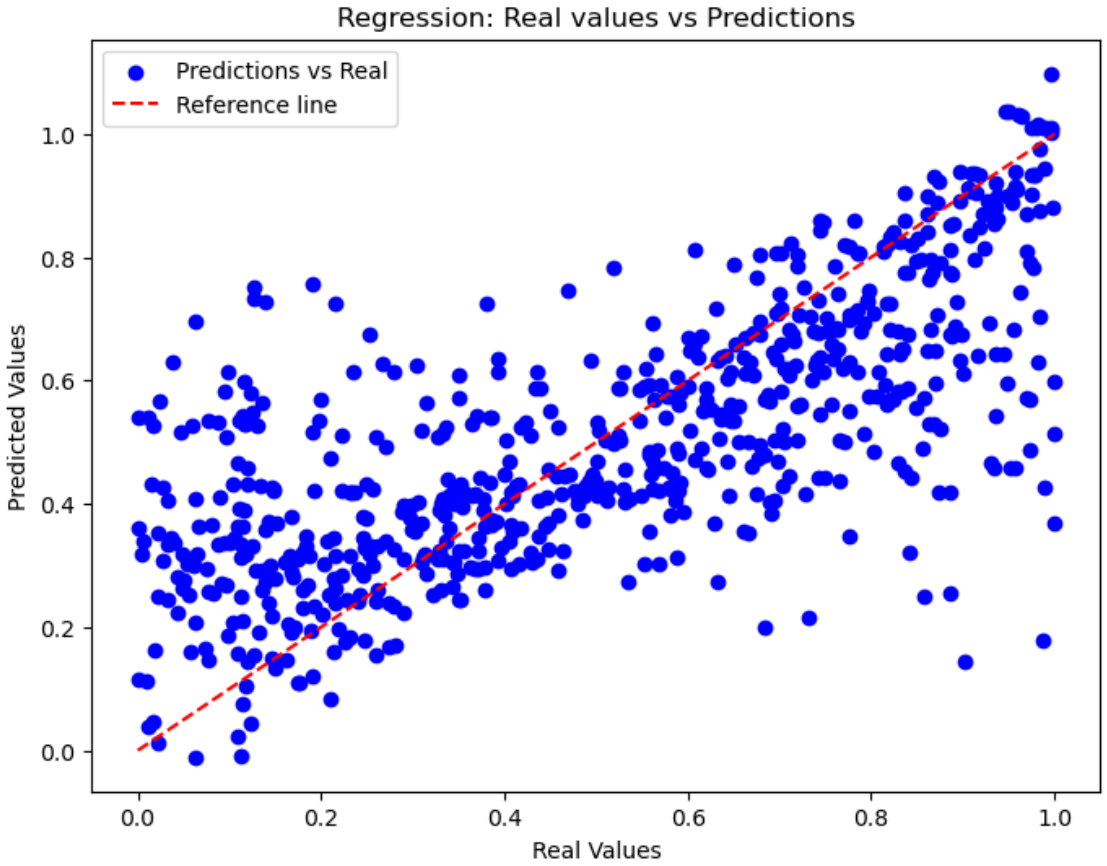
7 SVR

[33]:
```
from sklearn.svm import SVR

# Entrenar el modelo
svr_model = SVR(kernel='rbf')
svr_model.fit(x_train_scaled, y_train)

# Hacer predicciones y evaluar
y_pred_svr = svr_model.predict(x_test_scaled)
```

[34]:
```
# y_pred = rf_model.predict(x_test_pca)
y_pred = svr_model.predict(x_test_scaled)



mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error (MSE): {mse:.2f}")

mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error (MAE): {mae:.2f}")

r2 = r2_score(y_test, y_pred)
```

```
print(f"R² Score: {r2:.2f}")


plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, color='blue', label="Predictions vs Real")
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red',␣
 ↪linestyle='--', label="Reference line")
plt.xlabel("Real Values")
plt.ylabel("Predicted Values")
plt.title("Regression: Real values vs Predictions")
plt.legend()
plt.show()
```

```
Mean Squared Error (MSE): 0.04
Mean Absolute Error (MAE): 0.15
R² Score: 0.53
```



[35]:
```
import torch
import torch.nn as nn
```

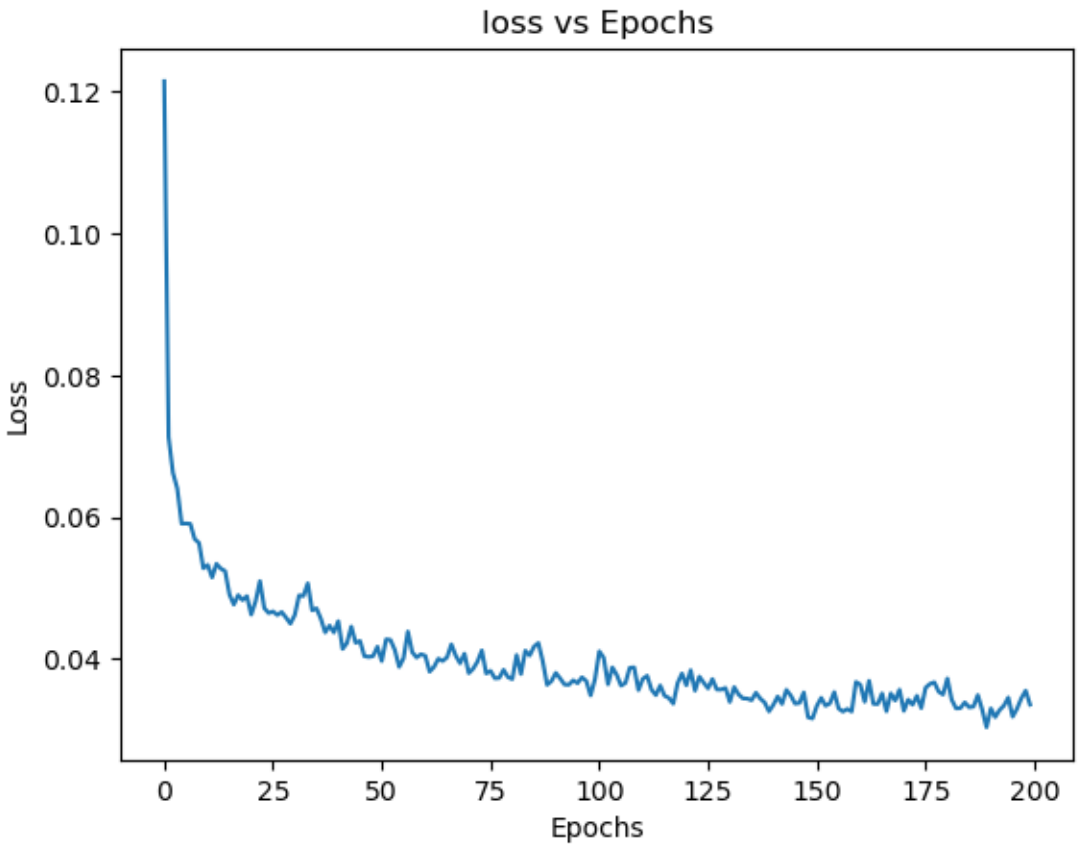
```
import torch.optim as optim
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from torch.utils.data import DataLoader, TensorDataset

# Convert los data to PyTorch tensores

X_train_tensor = torch.tensor(x_train_scaled.values, dtype=torch.float32)
X_test_tensor = torch.tensor(x_test_scaled.values, dtype=torch.float32)

y_train_tensor = torch.tensor(y_train.values, dtype=torch.float32).view(-1, 1)
y_test_tensor = torch.tensor(y_test.values, dtype=torch.float32).view(-1, 1)


# X_train_tensor = torch.tensor(x_train_scaled, dtype=torch.float32)
# X_test_tensor = torch.tensor(x_test_scaled, dtype=torch.float32)
# y_train_tensor = torch.tensor(y_train.values, dtype=torch.float32).view(-1, 1)
# y_test_tensor = torch.tensor(y_test.values, dtype=torch.float32).view(-1, 1)


train_data = TensorDataset(X_train_tensor, y_train_tensor)
train_loader = DataLoader(train_data, batch_size=254, shuffle=True)

test_data = TensorDataset(X_test_tensor, y_test_tensor)
test_loader = DataLoader(test_data, batch_size=254, shuffle=False)

# DNN
class DNNModel(nn.Module):
    def __init__(self, input_size):
        super(DNNModel, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(input_size, 256),
            nn.BatchNorm1d(256),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(256, 128),
            nn.BatchNorm1d(128),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(128, 64),
            nn.BatchNorm1d(64),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(64, 32),
            nn.BatchNorm1d(32),
            nn.ReLU(),
            nn.Linear(32, 1)  # output
        )
```

```
    def forward(self, x):
        return self.model(x)


input_dim = X_train_tensor.shape[1]
model = DNNModel(input_dim)

# D (loss function) and optimizer
criterion = nn.MSELoss()  # Loss (MSE)
optimizer = optim.Adam(model.parameters(), lr=0.001)  # Optimizer Adam

# Función para entrenar el modelo
def train_model(model, train_loader, criterion, optimizer, epochs=250):
    model.train()
    train_losses = []
    for epoch in range(epochs):
        epoch_loss = 0.0
        for data, target in train_loader:
            # Pasar los datos al modelo
            optimizer.zero_grad()
            output = model(data)
            loss = criterion(output, target)
            loss.backward()
            optimizer.step()
            epoch_loss += loss.item()

        avg_loss = epoch_loss / len(train_loader)
        train_losses.append(avg_loss)
        if (epoch+1) % 10 == 0:
            print(f'Epoch {epoch+1}/{epochs}, Loss: {avg_loss:.4f}')

    return train_losses

# trining
train_losses = train_model(model, train_loader, criterion, optimizer,␣
 ↪epochs=200)

# evaluate
def evaluate_model(model, test_loader):
    model.eval()
    y_pred = []
    y_true = []
    with torch.no_grad():
        for data, target in test_loader:
            output = model(data)
            y_pred.extend(output.numpy())
```

```
            y_true.extend(target.numpy())
    return y_true, y_pred

# evaluate
y_true, y_pred = evaluate_model(model, test_loader)


mse_dnn = mean_squared_error(y_true, y_pred)
mae_dnn = mean_absolute_error(y_true, y_pred)
r2_dnn = r2_score(y_true, y_pred)

print(f"MSE: {mse_dnn:.2f}")
print(f"MAE: {mae_dnn:.2f}")
print(f"R²: {r2_dnn:.2f}")
plt.plot(train_losses)
plt.title('loss vs Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
```

```
Epoch 10/200, Loss: 0.0528
Epoch 20/200, Loss: 0.0488
Epoch 30/200, Loss: 0.0450
Epoch 40/200, Loss: 0.0437
Epoch 50/200, Loss: 0.0417
Epoch 60/200, Loss: 0.0406
Epoch 70/200, Loss: 0.0407
Epoch 80/200, Loss: 0.0374
Epoch 90/200, Loss: 0.0368
Epoch 100/200, Loss: 0.0371
Epoch 110/200, Loss: 0.0356
Epoch 120/200, Loss: 0.0379
Epoch 130/200, Loss: 0.0359
Epoch 140/200, Loss: 0.0326
Epoch 150/200, Loss: 0.0316
Epoch 160/200, Loss: 0.0367
Epoch 170/200, Loss: 0.0356
Epoch 180/200, Loss: 0.0350
Epoch 190/200, Loss: 0.0303
Epoch 200/200, Loss: 0.0335
MSE: 0.02
MAE: 0.11
R²: 0.72
```

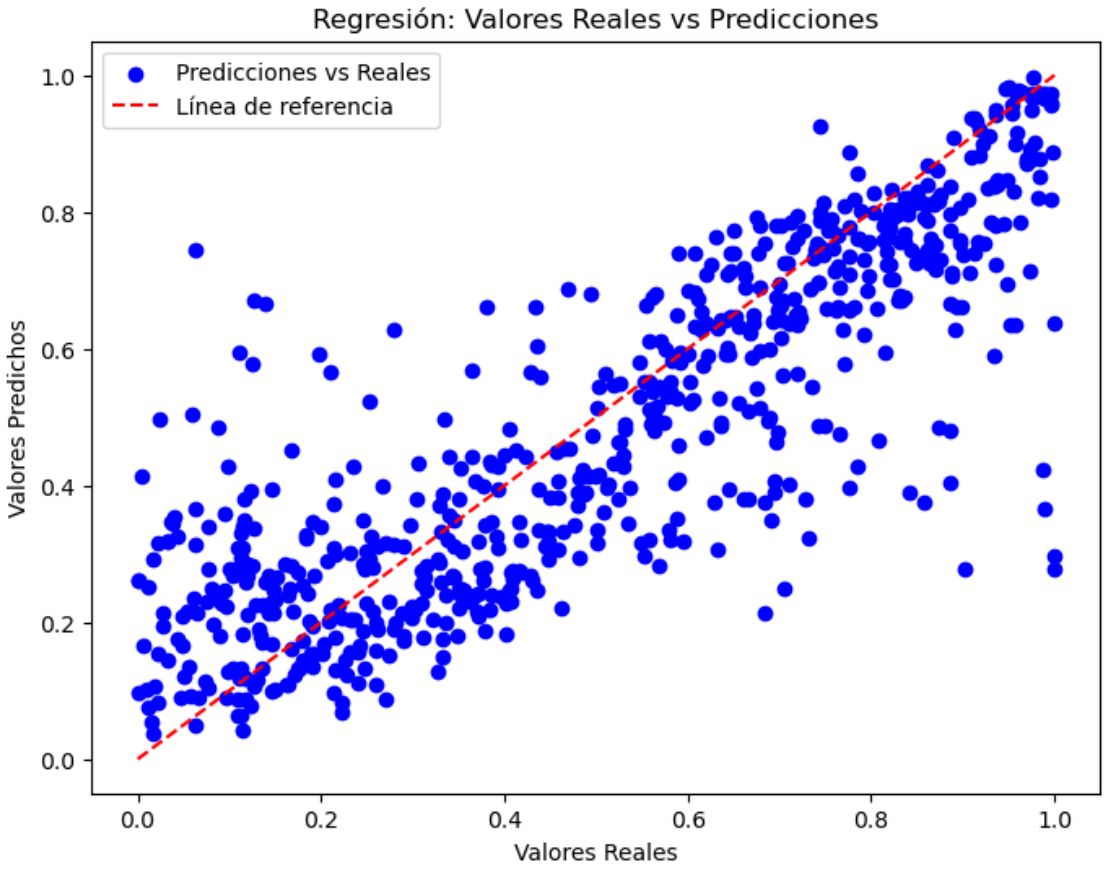
[35]: Text(0, 0.5, 'Loss')



[36]:
```
# y_pred = rf_model.predict(x_test_pca)
#y_pred = svr_model.predict(x_test_pca)


mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error (MSE): {mse:.2f}")

mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error (MAE): {mae:.2f}")

r2 = r2_score(y_test, y_pred)
print(f"R² Score: {r2:.2f}")


plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, color='blue', label="Predicciones vs Reales")
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red',␣
 ↪linestyle='--', label="Línea de referencia")
plt.xlabel("Valores Reales")
plt.ylabel("Valores Predichos")
```

```
plt.title("Regresión: Valores Reales vs Predicciones")
plt.legend()
plt.show()
```

```
Mean Squared Error (MSE): 0.02
Mean Absolute Error (MAE): 0.11
R² Score: 0.72
```



8 implementation

[ ]:

[37]:
```
#x[205:209],

y[800:]
```

[37]:
```
800    0.215328
801    0.211679
```

```
802     0.208029
803     0.204380
804     0.200730
          …
6781    0.015936
6782    0.011952
6783    0.007968
6784    0.003984
6785    0.000000
Name: RUL, Length: 5986, dtype: float64
```

[38]:
```
new_data=x[800:804]
# new_data = {
#     'type': ['-1'],
#     'ambient_temperature': [0.784018],
#     'Capacity': [0.784018],
#     'Re': [0.063757],
#     'Rct': [0.199213],

# }
#new_data = pd.DataFrame(new_data)
```

[39]:
```
mms = MinMaxScaler()
mms.fit(x_train)
real_time_data_scaled = mms.transform(new_data)
```

[40]:
```
# with torch.no_grad():
#     real_time_data_tensor = torch.tensor(real_time_data_scaled, dtype=torch.
 ↪float32)
#     predicted_rul = model(real_time_data_tensor)

# print(f"Predicted RUL: {predicted_rul.item()}")
```

[41]:
```
# Cargar el modelo guardado
rf_model_loaded = joblib.load('random_forest_model.pkl')

# Hacer predicciones con el modelo cargado
y_pred_real_time = rf_model_loaded.predict(real_time_data_scaled)
y_pred_real_time
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/base.py:464: UserWarning: X does
not have valid feature names, but RandomForestRegressor was fitted with feature
names
  warnings.warn(
```

[41]:
```
array([0.38321429, 0.25875637, 0.26233726, 0.22751082])
```

[ ]:

[ ]: