READ DATA

```
from google.colab import drive
drive.mount('/content/drive',force_remount=True)
```

```
Mounted at /content/drive
```

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import FunctionTransformer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import seaborn as sns
from sklearn.feature_selection import SelectKBest, f_classif
from keras.layers import Dense, Activation, Flatten, Dropout, BatchNormalization

from keras.models import Sequential
from keras.callbacks import History, ModelCheckpoint
```

```
index_names = ['ENGINE', 'CYCLE']
setting_names = ['SET1', 'SET2', 'SET3']
sensor_names=[ "INLET_TEMP",
"LPC_OUT_TEMP",
"HPC_OUT_TEMP",
"LPT_OUT_TEMP",
"FAN_IN_PR",
"BYPASS-PR",
"HPC_OUT_PR",
"FAN_RPM",
"CORE_RPM",
"ENGINE_PR",
"HPC_OUT",
"FUEL_RATIO",
"FAN_RPM_CORR",
"CORE_RPM_CORR",
"BYPASS RATIO",
"FUEL_RATIO_BURNER",
"ENTHALPY_BLEED",
"FAN_SPEED_REQ",
"CONV_FAN_SPEED",
"HP_AIRFLOW",
"LPC_AIRFLOW" ]
col_names = index_names + setting_names + sensor_names
```

```
data_train1=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/CMaps/train_FD001.txt',sep=" ")
data_train2=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/CMaps/train_FD002.txt',sep=" ")
data_train3=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/CMaps/train_FD003.txt',sep=" ")
data_train4=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/CMaps/train_FD004.txt',sep=" ")
data_train = pd.concat([data_train1, data_train2, data_train3, data_train4], axis=0)
# Reset index
data_train1.drop(columns=['Unnamed: 26','Unnamed: 27'],inplace=True)
# data_train1.drop(columns=[26,27],inplace=True)
data_train1.columns=col_names

# data_train2.drop(columns=[26,27],inplace=True)
# data_train2.columns=col_names

# data_train3.drop(columns=[26,27],inplace=True)
# data_train3.columns=col_names

# data_train4.drop(columns=[26,27],inplace=True)
# data_train4.columns=col_names


# data_train.reset_index(drop=True, inplace=True)
# data_train.drop(columns=[26,27],inplace=True)
```

```
# data_train.columns=col_names
data_train1.head()
```

	ENGINE	CYCLE	SET1	SET2	SET3	INLET_TEMP	LPC_OUT_TEMP	HPC_OUT_TEMP	LPT_OUT_
0	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	14
1	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	14
2	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	14
3	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	14
4	1	6	-0.0043	-0.0001	100.0	518.67	642.10	1584.47	13

```
data_train1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20630 entries, 0 to 20629
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   ENGINE             20630 non-null  int64
 1   CYCLE              20630 non-null  int64
 2   SET1               20630 non-null  float64
 3   SET2               20630 non-null  float64
 4   SET3               20630 non-null  float64
 5   INLET_TEMP         20630 non-null  float64
 6   LPC_OUT_TEMP       20630 non-null  float64
 7   HPC_OUT_TEMP       20630 non-null  float64
 8   LPT_OUT_TEMP       20630 non-null  float64
 9   FAN_IN_PR          20630 non-null  float64
 10  BYPASS-PR          20630 non-null  float64
 11  HPC_OUT_PR         20630 non-null  float64
 12  FAN_RPM            20630 non-null  float64
 13  CORE_RPM           20630 non-null  float64
 14  ENGINE_PR          20630 non-null  float64
 15  HPC_OUT            20630 non-null  float64
 16  FUEL_RATIO         20630 non-null  float64
 17  FAN_RPM_CORR       20630 non-null  float64
 18  CORE_RPM_CORR      20630 non-null  float64
 19  BYPASS RATIO       20630 non-null  float64
 20  FUEL_RATIO_BURNER  20630 non-null  float64
 21  ENTHALPY_BLEED     20630 non-null  int64
 22  FAN_SPEED_REQ      20630 non-null  int64
 23  CONV_FAN_SPEED     20630 non-null  float64
 24  HP_AIRFLOW         20630 non-null  float64
 25  LPC_AIRFLOW        20630 non-null  float64
dtypes: float64(22), int64(4)
memory usage: 4.1 MB
```

Note: There exists no NAN/NULL data. Moreover, there is no object or non-value data

```
data_train1.describe()
```

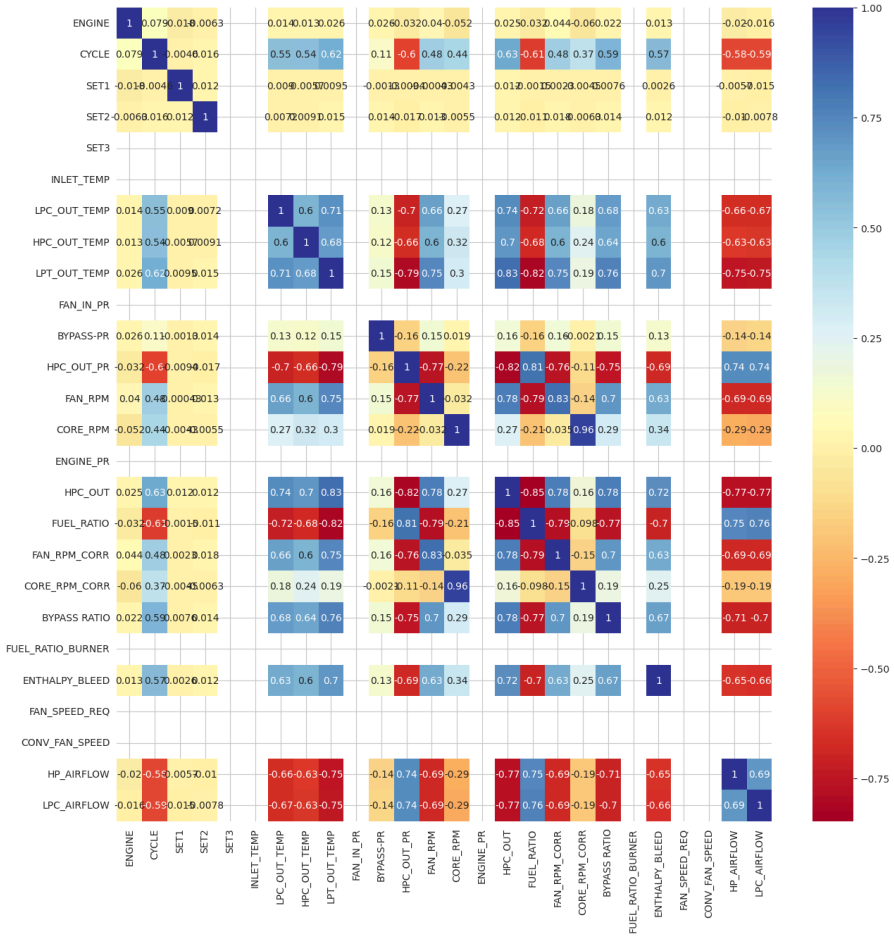
	ENGINE	CYCLE	SET1	SET2	SET3	INLET_TEMP	LPC_OU
count	20630.000000	20630.000000	20630.000000	20630.000000	20630.0	20630.00	20630.
mean	51.509016	108.813088	-0.000009	0.000002	100.0	518.67	642.
std	29.226226	68.878570	0.002187	0.000293	0.0	0.00	0.
min	1.000000	1.000000	-0.008700	-0.000600	100.0	518.67	641.
25%	26.000000	52.000000	-0.001500	-0.000200	100.0	518.67	642.
50%	52.000000	104.000000	-0.000000	0.000000	100.0	518.67	642.
75%	77.000000	156.000000	0.001500	0.000300	100.0	518.67	643.
max	100.000000	362.000000	0.008700	0.000600	100.0	518.67	644.

8 rows × 26 columns

```
data_train1.nunique()
```

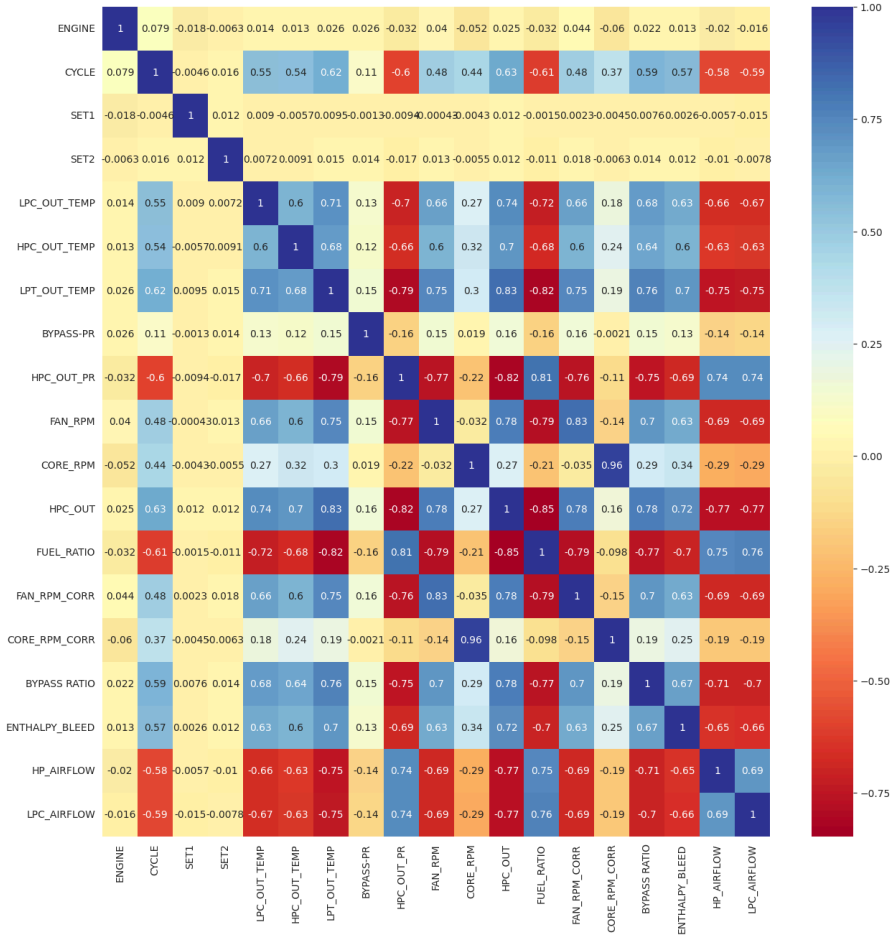
```
ENGINE                100
CYCLE                 362
SET1                  158
SET2                   13
SET3                    1
INLET_TEMP              1
LPC_OUT_TEMP          310
HPC_OUT_TEMP         3012
LPT_OUT_TEMP         4051
FAN_IN_PR               1
BYPASS-PR               2
HPC_OUT_PR            513
FAN_RPM                53
CORE_RPM             6403
ENGINE_PR               1
HPC_OUT               159
FUEL_RATIO            427
FAN_RPM_CORR           56
CORE_RPM_CORR        6078
BYPASS RATIO         1918
FUEL_RATIO_BURNER       1
ENTHALPY_BLEED         13
FAN_SPEED_REQ           1
CONV_FAN_SPEED          1
HP_AIRFLOW            120
LPC_AIRFLOW          4745
dtype: int64
```

```
plt.figure(figsize=(15,15))
sns.set_style("whitegrid", {"axes.facecolor": ".0"})
df_cluster2 = data_train1.corr()
plot_kws={"s": 1}
sns.heatmap(data_train1.corr(),
            cmap='RdYlBu',
            annot=True,
            linecolor='lightgrey').set_facecolor('white')
```


ENGINE
CYCLE
SET1
SET2
SET3
INLET_TEMP
LPC_OUT_TEMP
HPC_OUT_TEMP
LPT_OUT_TEMP
FAN_IN_PR
BYPASS-PR
HPC_OUT_PR
FAN_RPM
CORE_RPM
ENGINE_PR
HPC_OUT
FUEL_RATIO
FAN_RPM_CORR
CORE_RPM_CORR
BYPASS RATIO
FUEL_RATIO_BURNER
ENTHALPY_BLEED
FAN_SPEED_REQ
CONV_FAN_SPEED
HP_AIRFLOW
LPC_AIRFLOW
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

```
unwanted=[]
for i in data_train1.select_dtypes(include=np.number):
    if data_train1[i].nunique()==1:
        unwanted.append(i)
data_train1.drop(columns=unwanted, inplace=True)
```

```
plt.figure(figsize=(15,15))
sns.set_style("whitegrid", {"axes.facecolor": ".0"})
df_cluster2 = data_train1.corr()
plot_kws={"s": 1}
sns.heatmap(data_train1.corr(),
            cmap='RdYlBu',
            annot=True,
            linecolor='lightgrey').set_facecolor('white')
```




```
data_train1['BYPASS-PR'].unique()
```

```
array([21.61, 21.6 ])
```

```
(((data_train1['BYPASS-PR']==21.61).sum())/data_train1.shape[0])*100
```

```
98.0319922443044
```

```
(((data_train1['BYPASS-PR']==21.6).sum())/data_train1.shape[0])*100
```

```
1.9680077556955888
```

```
data_train1.drop(['BYPASS-PR'],axis=1,inplace=True)
```

```
#OUTLIER TREATMENT FOR TRAIN DATA
# calculate the z-scores for each column
z_scores = data_train1.apply(lambda x: np.abs((x - x.mean()) / x.std()))

# set a threshold for the z-score
threshold = 3

# identify the outliers
outliers = z_scores > threshold
```

```
z_scores = (data_train1 - data_train1.mean()) / data_train1.std()

# Replace values that exceed a certain threshold with the mode
threshold = 2.5
for col in data_train1.columns:
    outlier_mask = z_scores[col].abs() > threshold
    data_train1.loc[outlier_mask, col] = data_train1[col].mask(outlier_mask).mode()[0]
```

```
data_train1['CYCLE'].max()
data_train_RUL = data_train1.groupby(['ENGINE']).agg({'CYCLE':'max'})
data_train_RUL.rename(columns={'CYCLE':'LIFE'},inplace=True)
data_train_RUL.head()
```

ENGINE	LIFE
1	192
2	281
3	179
4	189
5	269

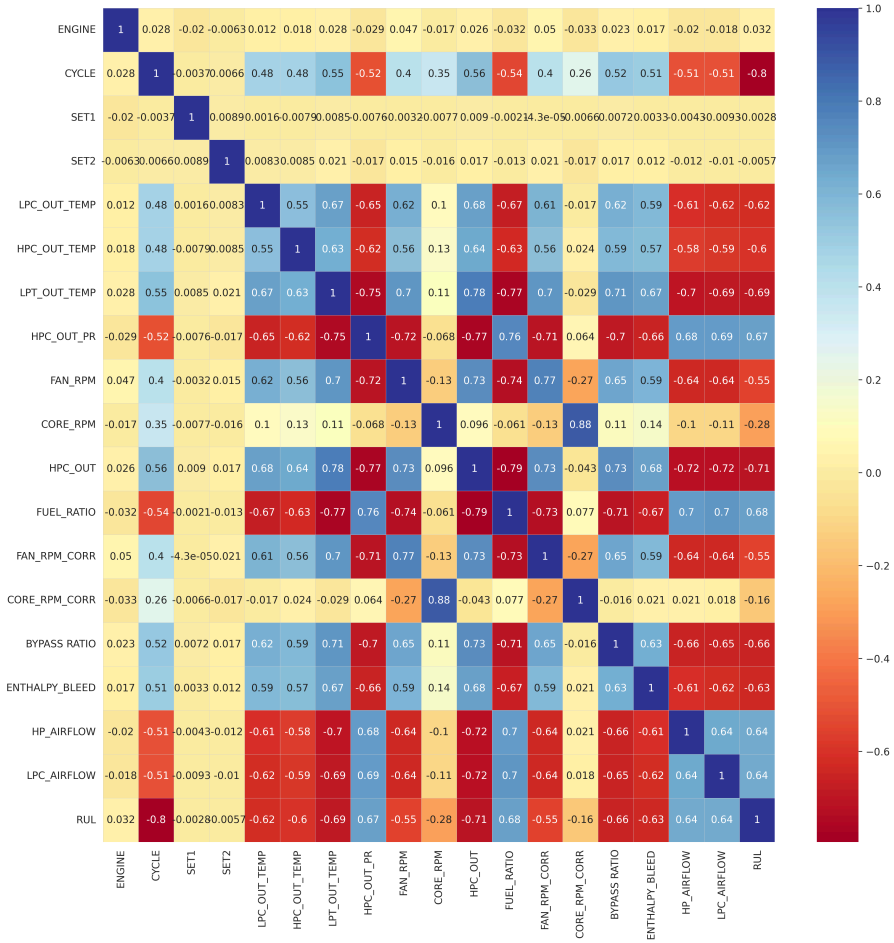
```
data_train1=data_train1.merge(data_train_RUL,how='left',on=['ENGINE'])
```

```
data_train1['RUL']=data_train1['LIFE']-data_train1['CYCLE']
data_train1.drop(['LIFE'],axis=1,inplace=True)

# the RUL prediction is only useful nearer to the end of the engine's life, therefore we put an upper limit on the RUL
# this is a bit sneaky, since it supposes that the test set has RULs of less than this value, the closer you are
# to the true value, the more accurate the model will be
data_train1['RUL'][data_train1['RUL']>125]=125
data_train1.head()
```

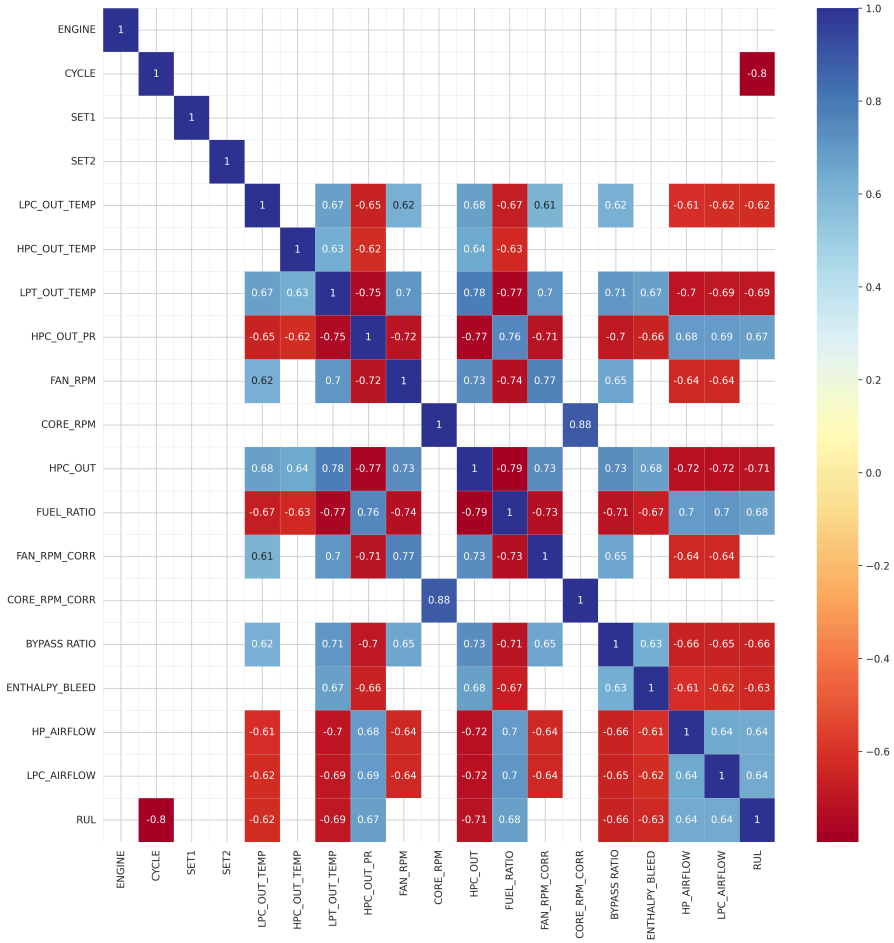
```
<ipython-input-19-abe0d968928b>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  data_train1['RUL'][data_train1['RUL']>125]=125
```

	ENGINE	CYCLE	SET1	SET2	LPC_OUT_TEMP	HPC_OUT_TEMP	LPT_OUT_TEMP	HPC_OUT_PR
0	1	2	0.0019	-0.0003	642.15	1591.82	1403.14	553.75
1	1	3	-0.0043	0.0003	642.35	1587.99	1404.20	554.26
2	1	4	0.0007	0.0000	642.35	1582.79	1401.87	554.45
3	1	5	-0.0019	-0.0002	642.37	1582.85	1406.22	554.00

```
plt.figure(figsize=(15,15),dpi=300)
sns.heatmap(data_train1.corr(),
            cmap='RdYlBu',
            annot=True,
            linewidths=0.2,
            linecolor='lightgrey').set_facecolor('white')
plt.show()
```


ENGINE 1 0.028 -0.02 -0.0063 0.012 0.018 0.028 -0.029 0.047 -0.017 0.026 -0.032 0.05 -0.033 0.023 0.017 -0.02 -0.018 0.032
CYCLE 0.028 1 -0.0037 0.0066 0.48 0.48 0.55 -0.52 0.4 0.35 0.56 -0.54 0.4 0.26 0.52 0.51 -0.51 -0.51 -0.8
SET1 -0.02 -0.0037 1 0.0089 0.0016 -0.0079 0.0085 -0.0076 0.0032 0.0077 0.009 -0.0021 4.3e-05 0.0066 0.0072 0.0033 -0.0043 0.0093 0.0028
SET2 -0.0063 0.0066 0.0089 1 0.0083 0.0085 0.021 -0.017 0.015 -0.016 0.017 -0.013 0.021 -0.017 0.017 0.012 -0.012 -0.01 -0.0057
LPC_OUT_TEMP 0.012 0.48 0.0016 0.0083 1 0.55 0.67 -0.65 0.62 0.1 0.68 -0.67 0.61 -0.017 0.62 0.59 -0.61 -0.62 -0.62
HPC_OUT_TEMP 0.018 0.48 -0.0079 0.0085 0.55 1 0.63 -0.62 0.56 0.13 0.64 -0.63 0.56 0.024 0.59 0.57 -0.58 -0.59 -0.6
LPT_OUT_TEMP 0.028 0.55 0.0085 0.021 0.67 0.63 1 -0.75 0.7 0.11 0.78 -0.77 0.7 -0.029 0.71 0.67 -0.7 -0.69 -0.69
HPC_OUT_PR -0.029 -0.52 -0.0076 -0.017 -0.65 -0.62 -0.75 1 -0.72 -0.068 -0.77 0.76 -0.71 0.064 -0.7 -0.66 0.68 0.69 0.67
FAN_RPM 0.047 0.4 -0.0032 0.015 0.62 0.56 0.7 -0.72 1 -0.13 0.73 -0.74 0.77 -0.27 0.65 0.59 -0.64 -0.64 -0.55
CORE_RPM -0.017 0.35 -0.0077 -0.016 0.1 0.13 0.11 -0.068 -0.13 1 0.096 -0.061 -0.13 0.88 0.11 0.14 -0.1 -0.11 -0.28
HPC_OUT 0.026 0.56 0.009 0.017 0.68 0.64 0.78 -0.77 0.73 0.096 1 -0.79 0.73 -0.043 0.73 0.68 -0.72 -0.72 -0.71
FUEL_RATIO -0.032 -0.54 -0.0021 -0.013 -0.67 -0.63 -0.77 0.76 -0.74 -0.061 -0.79 1 -0.73 0.077 -0.71 -0.67 0.7 0.7 0.68
FAN_RPM_CORR 0.05 0.4 -4.3e-05 0.021 0.61 0.56 0.7 -0.71 0.77 -0.13 0.73 -0.73 1 -0.27 0.65 0.59 -0.64 -0.64 -0.55
CORE_RPM_CORR -0.033 0.26 -0.0066 -0.017 -0.017 0.024 -0.029 0.064 -0.27 0.88 -0.043 0.077 -0.27 1 -0.016 0.021 0.021 0.018 -0.16
BYPASS RATIO 0.023 0.52 0.0072 0.017 0.62 0.59 0.71 -0.7 0.65 0.11 0.73 -0.71 0.65 -0.016 1 0.63 -0.66 -0.65 -0.66
ENTHALPY_BLEED 0.017 0.51 0.0033 0.012 0.59 0.57 0.67 -0.66 0.59 0.14 0.68 -0.67 0.59 0.021 0.63 1 -0.61 -0.62 -0.63
HP_AIRFLOW -0.02 -0.51 -0.0043 -0.012 -0.61 -0.58 -0.7 0.68 -0.64 -0.1 -0.72 0.7 -0.64 0.021 -0.66 -0.61 1 0.64 0.64
LPC_AIRFLOW -0.018 -0.51 -0.0093 -0.01 -0.62 -0.59 -0.69 0.69 -0.64 -0.11 -0.72 0.7 -0.64 0.018 -0.65 -0.62 0.64 1 0.64
RUL 0.032 -0.8 -0.0028 0.0057 -0.62 -0.6 -0.69 0.67 -0.55 -0.28 -0.71 0.68 -0.55 -0.16 -0.66 -0.63 0.64 0.64 1
ENGINE CYCLE SET1 SET2 LPC_OUT_TEMP HPC_OUT_TEMP LPT_OUT_TEMP HPC_OUT_PR FAN_RPM CORE_RPM HPC_OUT FUEL_RATIO FAN_RPM_CORR CORE_RPM_CORR BYPASS RATIO ENTHALPY_BLEED HP_AIRFLOW LPC_AIRFLOW RUL
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

```
plt.figure(figsize=(15,15),dpi=300)
threshold = 0.6
sns.set_style("whitegrid", {"axes.facecolor": ".0"})
df_cluster2 = data_train1.corr()
mask = df_cluster2.where((abs(df_cluster2) >= threshold)).isna()
plot_kws={"s": 1}
sns.heatmap(df_cluster2,
            cmap='RdYlBu',
            annot=True,
            mask=mask,
            linewidths=0.2,
            linecolor='lightgrey').set_facecolor('white')
```


ENGINE
CYCLE
SET1
SET2
LPC_OUT_TEMP
HPC_OUT_TEMP
LPT_OUT_TEMP
HPC_OUT_PR
FAN_RPM
CORE_RPM
HPC_OUT
FUEL_RATIO
FAN_RPM_CORR
CORE_RPM_CORR
BYPASS RATIO
ENTHALPY_BLEED
HP_AIRFLOW
LPC_AIRFLOW
RUL
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

```
data_train1.drop(columns=['ENGINE','SET1','SET2','CORE_RPM',"CORE_RPM_CORR"], inplace=True)
```

```
list(data_train1)
```

```
['CYCLE',
 'LPC_OUT_TEMP',
 'HPC_OUT_TEMP',
 'LPT_OUT_TEMP',
 'HPC_OUT_PR',
 'FAN_RPM',
 'HPC_OUT',
 'FUEL_RATIO',
 'FAN_RPM_CORR',
 'BYPASS RATIO',
 'ENTHALPY_BLEED',
 'HP_AIRFLOW',
 'LPC_AIRFLOW',
 'RUL']
```

```
plt.figure(figsize=(15,15),dpi=300)
threshold = 0.6
sns.set_style("whitegrid", {"axes.facecolor": ".0"})
df_cluster2 = data_train1.corr()
mask = df_cluster2.where((abs(df_cluster2) >= threshold)).isna()
plot_kws={"s": 1}
sns.heatmap(df_cluster2,
            cmap='RdYlBu',
            annot=True,
            mask=mask,
            linewidths=0.2,
            linecolor='lightgrey').set_facecolor('white')
```