1. Data preparation

In [1]:

```python
#Import libraries and packages
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import random
import time
import joblib

from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import make_scorer
from sklearn.model_selection import LeaveOneGroupOut, cross_val_score, GridSearchCV

# Set random seed for reproducibility
random.seed(123)
```

In [2]:

```python
#Read csv and excel files
cost = pd.read_csv('cost.csv')
suppliers = pd.read_csv('suppliers.csv')
tasks = pd.read_excel('tasks.xlsx')
```

Looking at the dataframes and checking their information (number of columns, rows, data types)

In [3]:

```python
suppliers.head()
```

Out[3]:

	Supplier ID	SF1	SF2	SF3	SF4	SF5	SF6	SF7	SF8	SF9	SF10	SF11	SF12	SF13	SF14	SF15	SF16	SF17	SF18
0	S1	100	1000	1000	50	20	10	2	80	2000	100	1000	5	1000	1000	500	5000	100	96
1	S2	100	1000	1000	50	20	10	0	80	2000	100	1000	5	1000	1000	500	5000	100	96
2	S3	100	1000	1000	50	20	10	2	80	2000	100	1000	5	1000	1000	500	5000	0	96
3	S4	100	1000	1000	50	20	10	2	80	2000	100	1000	5	1000	1000	500	5000	5000	96
4	S5	10	1000	100	500	200	10	2	80	200	100	2000	8	2000	100	2000	5000	15000	90

In [4]:

```python
suppliers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64 entries, 0 to 63
Data columns (total 19 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Supplier ID  64 non-null     object
 1   SF1          64 non-null     int64 
 2   SF2          64 non-null     int64 
 3   SF3          64 non-null     int64 
 4   SF4          64 non-null     int64 
 5   SF5          64 non-null     int64 
 6   SF6          64 non-null     int64 
 7   SF7          64 non-null     int64 
 8   SF8          64 non-null     int64 
 9   SF9          64 non-null     int64 
 10  SF10         64 non-null     int64 
 11  SF11         64 non-null     int64 
 12  SF12         64 non-null     int64 
 13  SF13         64 non-null     int64 
 14  SF14         64 non-null     int64 
 15  SF15         64 non-null     int64 
 16  SF16         64 non-null     int64 
 17  SF17         64 non-null     int64 
 18  SF18         64 non-null     int64 
dtypes: int64(18), object(1)
memory usage: 9.6+ KB
```

In [5]:

```python
tasks.head()
```

Out[5]:

	Task ID	TF1	TF2	TF3	TF4	TF5	TF6	TF7	TF8	TF9	...	TF107	TF108	TF109	TF110	TF111	TF112	TI
0	2019 05 30	706	2539	428536374	367	0.144545	35342375	0.08	829	0.326506	...	125	0	219407.09	903728.98	10174793	0	
1	2019 09 26	697	2199	389831692	431	0.195998	20091114	0.05	460	0.209186	...	149	0	467568.10	1868010.80	19221510	0	
2	2019 11 29	262	4156	500027098	1510	0.363330	89708355	0.18	1010	0.243022	...	394	0	228358.20	1358750.92	38657530	0	
3	2020 01 03	469	4346	547810586	1376	0.316613	90478530	0.17	1097	0.252416	...	394	0	258487.96	1781016.69	39064840	0	
4	2020 01 07	555	3934	521676289	1039	0.264108	69762831	0.13	943	0.239705	...	394	0	285558.89	1869912.30	39064840	0	

5 rows × 117 columns

In [6]:

```python
tasks.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 130 entries, 0 to 129
Data columns (total 117 columns):
 #    Column   Dtype  
---   ------   -----  
 0    Task ID  object 
 1    TF1      int64  
 2    TF2      int64  
 3    TF3      int64  
 4    TF4      int64  
 5    TF5      float64
 6    TF6      int64  
 7    TF7      float64
 8    TF8      int64  
 9    TF9      float64
 10   TF10     int64  
 11   TF11     float64
 12   TF12     int64  
 13   TF13     float64
 14   TF14     int64  
 15   TF15     float64
 16   TF16     float64
 17   TF17     float64
 18   TF18     float64
 19   TF19     int64  
 20   TF20     float64
 21   TF21     float64
 22   TF22     float64
 23   TF23     int64  
 24   TF24     float64
 25   TF25     float64
 26   TF26     int64  
 27   TF27     int64  
 28   TF28     float64
 29   TF29     float64
 30   TF30     int64  
 31   TF31     int64  
 32   TF32     float64
 33   TF33     float64
 34   TF34     int64  
 35   TF35     int64  
 36   TF36     int64  
 37   TF37     int64  
 38   TF38     float64
 39   TF39     float64
 40   TF40     int64  
 41   TF41     int64  
 42   TF42     float64
 43   TF43     float64
 44   TF44     int64  
 45   TF45     int64  
 46   TF46     float64
 47   TF47     float64
 48   TF48     int64  
 49   TF49     int64  
 50   TF50     float64
 51   TF51     float64
 52   TF52     int64  
 53   TF53     float64
 54   TF54     int64  
 55   TF55     float64
 56   TF56     int64  
 57   TF57     float64
 58   TF58     int64  
 59   TF59     float64
 60   TF60     int64  
 61   TF61     float64
 62   TF62     int64  
 63   TF63     float64
 64   TF64     float64
 65   TF65     int64  
 66   TF66     float64
 67   TF67     float64
 68   TF68     float64
 69   TF69     int64  
 70   TF70     int64  
 71   TF71     int64  
 72   TF72     float64
 73   TF73     float64
 74   TF74     int64  
 75   TF75     int64  
 76   TF76     float64
 77   TF77     float64
 78   TF78     int64  
 79   TF79     int64  
 80   TF80     int64  
 81   TF81     float64
 82   TF82     float64
 83   TF83     int64  
 84   TF84     int64  
 85   TF85     float64
 86   TF86     float64
 87   TF87     int64  
 88   TF88     int64  
 89   TF89     float64
 90   TF90     float64
 91   TF91     int64  
 92   TF92     int64  
 93   TF93     float64
 94   TF94     float64
 95   TF95     int64  
 96   TF96     int64  
 97   TF97     float64
 98   TF98     float64
 99   TF99     int64  
 100  TF100    int64  
 101  TF101    float64
 102  TF102    float64
 103  TF103    int64  
 104  TF104    int64  
 105  TF105    float64
 106  TF106    float64
 107  TF107    int64  
 108  TF108    int64  
 109  TF109    float64
 110  TF110    float64
 111  TF111    int64  
 112  TF112    int64  
 113  TF113    int64  
 114  TF114    float64
 115  TF115    int64  
 116  TF116    float64
dtypes: float64(58), int64(58), object(1)
memory usage: 119.0+ KB
```

In [7]:

```python
cost.head()
```

Out[7]:

	Task ID	Supplier ID	Cost
0	30/05/2019	S1	0.478219
1	30/05/2019	S2	0.444543
2	30/05/2019	S3	0.521679
3	30/05/2019	S4	0.307331
4	30/05/2019	S5	0.357689

In [8]:

```python
cost.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7680 entries, 0 to 7679
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Task ID      7680 non-null   object 
 1   Supplier ID  7680 non-null   object 
 2   Cost         7680 non-null   float64
dtypes: float64(1), object(2)
memory usage: 180.1+ KB
```

In [9]:

```python
# Convert 'Task ID' to datetime format
tasks['Task ID'] = pd.to_datetime(tasks['Task ID'], infer_datetime_format=True)
cost['Task ID'] = pd.to_datetime(cost['Task ID'], infer_datetime_format=True)
```

1.1 Data Verification

Missing Values

Calculate the number of null values in each dataset

In [10]:

```python
print('Number of null values in cost:', cost.isnull().any().sum())
print('Number of null values in suppliers:', suppliers.isnull().any().sum())
print('Number of null values in tasks:', tasks.isnull().any().sum())
```

```
Number of null values in cost: 0
Number of null values in suppliers: 0
Number of null values in tasks: 0
```

Number of tasks, suppliers, features and cost values in all data sets

Count number of tasks, suppliers, features and costs values in all datasets

In [11]:

```python
print(f"There are {len(tasks['Task ID'].unique())} unique tasks in Tasks Dataset")
print(f"There are {len(cost['Task ID'].unique())} unique tasks in Costs Dataset")

print(f"\nThere are {len(cost['Supplier ID'].unique())} unique suppliers in Costs Dataset")
print(f"There are {len(suppliers['Supplier ID'].unique())} unique suppliers in Suppliers Dataset")

print(f"\nEach task is specified by {len(tasks.columns)-1} features")
print(f"Each supplier is specified by {len(suppliers.columns)-1} features")
print(f"\nThere is information about {len(cost)} costs associated with different tasks and suppliers")
```

```
There are 130 unique tasks in Tasks Dataset
There are 120 unique tasks in Costs Dataset

There are 64 unique suppliers in Costs Dataset
There are 64 unique suppliers in Suppliers Dataset

Each task is specified by 116 features
Each supplier is specified by 18 features

There is information about 7680 costs associated with different tasks and suppliers
```

ID match: Task IDs

Making sure the Task IDs in Costs and Tasks dataset match.

Cost dataset is the most important for our analysis, since it contains the true value of the predicted variable.

That is why we should check whether other datasets contain features for all tasks presented in the cost dataset

In [12]:

```python
cost_unique = pd.DataFrame(cost['Task ID'].unique())
tasks_unique = tasks['Task ID'].unique()
```

In [13]:

```python
print('Number of unique Task ID in the Cost dataset:', len(cost_unique))
print('Number of unique Task ID in the Tasks dataset:', len(tasks_unique))
print('\nNumber of unique Task ID in the Cost dataset that also appear in the Tasks dataset:',\
      cost_unique.isin(tasks_unique).sum().values[0])
```

```
Number of unique Task ID in the Cost dataset: 120
Number of unique Task ID in the Tasks dataset: 130

Number of unique Task ID in the Cost dataset that also appear in the Tasks dataset: 120
```

Only 120 tasks from Cost dataset are specified in the Task dataset. Consequently, we should remove from both datasets tasks that are not matched

In [14]:

```python
tasks_before = tasks.shape[0]
print('Number of tasks before removal:', tasks_before)

# Eliminating tasks with no costs specified from tasks dataset
tasks = tasks[tasks["Task ID"].isin(cost['Task ID'].unique())]

tasks_after = tasks.shape[0]
print('Number of tasks after removal:', tasks_after)

print(tasks_before - tasks_after, 'tasks were removed from the Tasks dataset')
```

```
Number of tasks before removal: 130
Number of tasks after removal: 120
10 tasks were removed from the Tasks dataset
```

In [15]:

```python
c_tasks_before = cost.shape[0]
print('Number of tasks before removal:', c_tasks_before)

# Eliminating tasks with no costs specified from cost dataset
cost = cost[cost["Task ID"].isin(tasks['Task ID'].unique())]

c_tasks_after = cost.shape[0]
print('Number of tasks after removal:', c_tasks_after)

print(c_tasks_before - c_tasks_after, 'tasks were removed from the Cost dataset')
```

```
Number of tasks before removal: 7680
Number of tasks after removal: 7680
0 tasks were removed from the Cost dataset
```

ID match: Supplier IDs

In [16]:

```python
cost_unique = pd.DataFrame(cost['Supplier ID'].unique())
suppliers_unique = suppliers['Supplier ID'].unique()

print('Number of unique Supplier ID in the Cost dataset:', len(cost_unique))
print('Number of unique Supplier ID in the Suppliers dataset:', len(suppliers_unique))

print('\nNumber of unique Supplier ID in the Cost dataset that also appear in the Suppliers dataset:',\
      cost_unique.isin(suppliers_unique).sum().values[0])
# All suppliers in the cost dataset are also present in the suppliers dataset, no adjustments required
```

```
Number of unique Supplier ID in the Cost dataset: 64
Number of unique Supplier ID in the Suppliers dataset: 64

Number of unique Supplier ID in the Cost dataset that also appear in the Suppliers dataset: 64
```

Nothing was removed from the Costs and the Suppliers dataset

Saving modified DataFrames

In [17]:

```python
cost_df = pd.DataFrame(cost)
suppliers_df = pd.DataFrame(suppliers).set_index('Supplier ID')
tasks_df = pd.DataFrame(tasks).set_index('Task ID')
```

1.2 Calculate the max, min, mean and var of each feature

1.2.1 Tasks Dataset

In [18]:

```python
tasks_df.head()
```

Out[18]:

	TF1	TF2	TF3	TF4	TF5	TF6	TF7	TF8	TF9	TF10	...	TF107	TF108	TF109	TF110	TF111
Task ID																
2019-05-30	706	2539	428536374	367	0.144545	35342375	0.08	829	0.326506	109388318	...	125	0	219407.09	903728.98	10174793
2019-09-26	697	2199	389831692	431	0.195998	20091114	0.05	460	0.209186	72475671	...	149	0	467568.10	1868010.80	19221510
2019-11-29	262	4156	500027098	1510	0.363330	89708355	0.18	1010	0.243022	112404944	...	394	0	228358.20	1358750.92	38657530
2020-01-03	469	4346	547810586	1376	0.316613	90478530	0.17	1097	0.252416	122674168	...	394	0	258487.96	1781016.69	39064840
2020-01-07	555	3934	521676289	1039	0.264108	69762831	0.13	943	0.239705	112768389	...	394	0	285558.89	1869912.30	39064840

5 rows × 116 columns

In [19]:

```python
# Calculating the variance of each task feature
var_tasks = pd.DataFrame(tasks_df.var()).T.rename(index={0:'var'})

# Getting summary statistics of each task feature
tasks_desc = tasks_df.describe()

# Concatenating both result for an easier read
tasks_summary = pd.concat([tasks_desc, var_tasks], axis=0)
tasks_summary
```

Out[19]:

	TF1	TF2	TF3	TF4	TF5	TF6	TF7	TF8	TF9	TF1
count	120.000000	1.200000e+02	1.200000e+02	120.000000	120.000000	1.200000e+02	120.000000	120.000000	120.000000	1.200000e+0
mean	83.100000	4.436525e+03	5.045244e+08	1516.933333	0.336457	9.266722e+07	0.183750	1348.375000	0.304799	1.208782e+0
std	190.002893	1.515095e+03	1.120227e+08	684.999758	0.079158	3.950462e+07	0.071185	511.151116	0.047781	3.792731e+0
min	0.000000	1.127000e+03	7.149093e+07	367.000000	0.144471	2.009111e+07	0.050000	223.000000	0.189399	9.828626e+0
25%	1.000000	3.196250e+03	4.434996e+08	997.750000	0.280037	6.051022e+07	0.137500	954.750000	0.268808	9.365975e+0
50%	1.000000	4.145000e+03	5.057017e+08	1386.500000	0.331774	8.541451e+07	0.180000	1203.000000	0.307448	1.121406e+0
75%	3.000000	5.839500e+03	5.690280e+08	2036.000000	0.376324	1.210465e+08	0.220000	1578.750000	0.344739	1.508170e+0
max	706.000000	7.908000e+03	7.699058e+08	3344.000000	0.642413	1.906866e+08	0.660000	2444.000000	0.388069	1.976945e+0
var	36101.099160	2.295513e+06	1.254908e+16	469224.667787	0.006266	1.560615e+15	0.005067	261275.463235	0.002283	1.438481e+1

9 rows × 116 columns

Remove features with variance lower than 0.01

In [20]:

```python
tf_before = tasks_df.shape[1]
print('Number of features before feature removal:', tf_before)

cols_to_del = []
for i in tasks_df.columns:
    if tasks_df[i].var()<= 0.01:
        cols_to_del.append(i)
tasks_df = tasks_df.drop(cols_to_del, axis=1)

tf_after = tasks_df.shape[1]
print('Number of features after feature removal:', tf_after)
print(tf_before - tf_after, 'features were removed from the Tasks dataset')

tasks_df.head()
```

```
Number of features before feature removal: 116
Number of features after feature removal: 83
33 features were removed from the Tasks dataset
```

Out[20]:

	TF1	TF2	TF3	TF4	TF6	TF8	TF10	TF12	TF14	TF16	...	TF102	TF103	TF105	TF106	TF107	
Task ID																	
2019-05-30	706	2539	428536374	367	35342375	829	109388318	452	65396430	1330.10	...	9836131.71	52077023	6.15	25.10	125	2
2019-09-26	697	2199	389831692	431	20091114	460	72475671	287	47355481	1403.06	...	7342792.32	44103725	9.13	30.13	149	4
2019-11-29	262	4156	500027098	1510	89708355	1010	112404944	812	83864672	1347.16	...	10064509.99	50354895	5.06	23.75	394	2
2020-01-03	469	4346	547810586	1376	90478530	1097	122674168	834	95043161	1345.70	...	9122217.84	49212552	5.25	26.07	394	2
2020-01-07	555	3934	521676289	1039	69762831	943	112768389	682	85212722	1356.06	...	8664591.29	48980163	5.80	27.35	394	2

5 rows × 83 columns

1.2.2 Suppliers Dataset

In [21]:
```python
suppliers_df.head()
```

Out[21]:

	SF1	SF2	SF3	SF4	SF5	SF6	SF7	SF8	SF9	SF10	SF11	SF12	SF13	SF14	SF15	SF16	SF17	SF18
Supplier ID																		
S1	100	1000	1000	50	20	10	2	80	2000	100	1000	5	1000	1000	500	5000	100	96
S2	100	1000	1000	50	20	10	0	80	2000	100	1000	5	1000	1000	500	5000	100	96
S3	100	1000	1000	50	20	10	2	80	2000	100	1000	5	1000	1000	500	5000	0	96
S4	100	1000	1000	50	20	10	2	80	2000	100	1000	5	1000	1000	500	5000	5000	96
S5	10	1000	100	500	200	10	2	80	200	100	2000	8	2000	100	2000	5000	15000	90

In [22]:
```python
# Calculating the variance of each task feature
var_suppliers = pd.DataFrame(suppliers_df.var()).T.rename(index={0:'var'})

# Getting summary statistics of each task feature
suppliers_desc = suppliers_df.describe()

# Concatenating both result for an easier read
suppliers_summary = pd.concat([suppliers_desc, var_suppliers], axis=0)
suppliers_summary
```

Out[22]:

	SF1	SF2	SF3	SF4	SF5	SF6	SF7	SF8	SF9	SI
count	64.000000	64.000000	64.000000	64.000000	64.000000	64.000000	64.000000	64.000000	6.400000e+01	64.0000
mean	353.125000	1031.250000	1031.250000	176.562500	70.625000	35.312500	1.968750	188.750000	2.062500e+03	196.8750
std	441.163235	757.371185	757.371185	220.581618	88.232647	44.116324	0.835117	213.615141	1.514742e+03	209.359
min	10.000000	100.000000	100.000000	5.000000	2.000000	1.000000	0.000000	8.000000	2.000000e+02	10.0000
25%	10.000000	100.000000	100.000000	5.000000	2.000000	1.000000	1.000000	8.000000	2.000000e+02	10.0000
50%	100.000000	1000.000000	1000.000000	50.000000	20.000000	10.000000	2.000000	80.000000	2.000000e+03	100.0000
75%	1000.000000	2000.000000	2000.000000	500.000000	200.000000	100.000000	3.000000	500.000000	4.000000e+03	500.0000
max	1000.000000	2000.000000	2000.000000	500.000000	200.000000	100.000000	3.000000	500.000000	4.000000e+03	500.0000
var	194625.000000	573611.111111	573611.111111	48656.250000	7785.000000	1946.250000	0.697421	45631.428571	2.294444e+06	43831.349

In [23]:
```python
print('Number of features that have variance less than 0.01:', (suppliers_summary.loc['var',:] < 0.01).sum())
```

```
Number of features that have variance less than 0.01: 0
```

There are no features that have variance less than 0.01 in the Suppliers dataset, so no features were removed

1.3 Feature Scaling

In [24]:
```python
# Feature scaling using MinMaxScaler
scaler = MinMaxScaler(feature_range =(-1,1))
tasks_df[:] = scaler.fit_transform(tasks_df)
suppliers_df[:] = scaler.fit_transform(suppliers_df)
```

Tasks and Suppliers dataframe after scaling

In [25]:
```python
tasks_df.head()
```

Out[25]:

	TF1	TF2	TF3	TF4	TF6	TF8	TF10	TF12	TF14	TF16	...	TF102	TF103	
Task ID														
2019-05-30	1.000000	-0.583542	0.022445	-1.000000	-0.821200	-0.454300	0.059902	-0.697013	-0.090516	-0.559193	...	-0.712664	-0.401560	-0.9
2019-09-26	0.974504	-0.683822	-0.088391	-0.957004	-1.000000	-0.786583	-0.333066	-0.931721	-0.394759	0.017794	...	-0.903937	-0.553691	-0.8
2019-11-29	-0.257790	-0.106621	0.227168	-0.232113	-0.183833	-0.291310	0.092017	-0.184922	0.220932	-0.424278	...	-0.695144	-0.434418	-0.9
2020-01-03	0.328612	-0.050583	0.364002	-0.322136	-0.174803	-0.212967	0.201342	-0.153627	0.409446	-0.435824	...	-0.767431	-0.456214	-0.9
2020-01-07	0.572238	-0.172099	0.289163	-0.548539	-0.417667	-0.351643	0.095886	-0.369844	0.243666	-0.353895	...	-0.802537	-0.460648	-0.9

5 rows × 83 columns

In [26]:
```python
suppliers_df.head()
```

Out[26]:

	SF1	SF2	SF3	SF4	SF5	SF6	SF7	SF8	SF9	SF10	SF11	SF12	SI
Supplier ID													
S1	-0.818182	-0.052632	-0.052632	-0.818182	-0.818182	-0.818182	0.333333	-0.707317	-0.052632	-0.632653	-0.052632	0.142857	-0.052
S2	-0.818182	-0.052632	-0.052632	-0.818182	-0.818182	-0.818182	-1.000000	-0.707317	-0.052632	-0.632653	-0.052632	0.142857	-0.052
S3	-0.818182	-0.052632	-0.052632	-0.818182	-0.818182	-0.818182	0.333333	-0.707317	-0.052632	-0.632653	-0.052632	0.142857	-0.052
S4	-0.818182	-0.052632	-0.052632	-0.818182	-0.818182	-0.818182	0.333333	-0.707317	-0.052632	-0.632653	-0.052632	0.142857	-0.052
S5	-1.000000	-0.052632	-1.000000	1.000000	1.000000	-0.818182	0.333333	-0.707317	-1.000000	-0.632653	1.000000	1.000000	1.000

1.4 Absolute correlation of all pairs of features

In [27]:
```python
# Create the correlation matrix (absolute values)
corr_matrix = tasks_df.corr().abs()

# Generate a mask for the upper triangle; True = do NOT show
mask = np.zeros_like(corr_matrix, dtype=bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(30, 30))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(
    corr_matrix,          # The data to plot
    mask=mask,      # Mask some cells
    cmap=cmap,      # What colors to plot the heatmap as
    annot=False,     # Should the values be plotted in the cells?
    vmax=1,        # The maximum value of the legend. All higher vals will be same color
    vmin=0,       # The minimum value of the legend. All lower vals will be same color
    center=0.5,       # The center value of the legend. With divergent cmap, where white is
    square=True,    # Force cells to be square
    linewidths=.5, # Width of lines that divide cells
    cbar_kws={"shrink": .1}  # Extra kwargs for the legend; in this case, shrink by 50%
)
plt.title('Correlation Heatmap')
plt.xlabel('Task Features')
plt.ylabel('Task Features')


# Save plot
f.savefig('Full_Corr_Matrix.png')
```

Removing features with the correlation >= 0.8

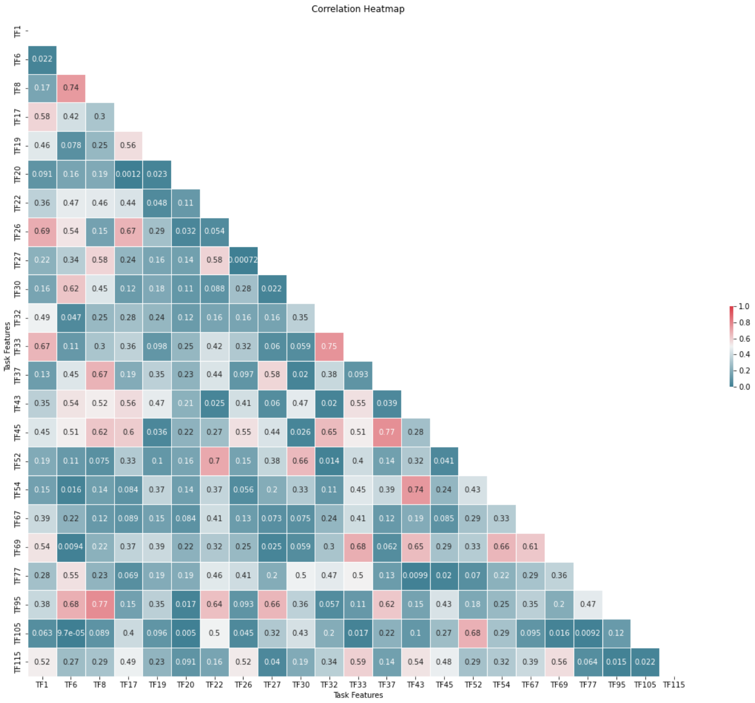
In [28]:
```python
# Define function to get the feature pair with the highest correlation
def get_pair(df):
    a = np.ones(df.shape, int)
    np.fill_diagonal(a, 0)
    df = df*a
    max_corr = np.max(df.max())

    pair = (np.where(df==max_corr)[0][0], np.where(df==max_corr)[1][0])
    return pair

# Define function to get a dataframe in the shape of given dataframe
# Filled with all 1s except for the diagonal which are filled by 0s
def get_mask(df):
    a = np.ones(df.shape, int)
    np.fill_diagonal(a, 0)
    return a

# Loop will run as long as there is a feature pair with correlation >= 0.8
while np.any((corr_matrix * get_mask(corr_matrix)) >= 0.8):
    # Get feature pair with highest correlation
    pair = get_pair(corr_matrix)

    # Calculating number of cases when correlation with first feature is higher than with the second one
    number_of_corr_greater_for_first_var = sum(corr_matrix.iloc[:, pair[0]] > corr_matrix.iloc[:, pair[1]])
    number_of_corr_greater_for_second_var = sum(corr_matrix.iloc[:, pair[1]] > corr_matrix.iloc[:, pair[0]])

    # Choose which feature to drop
    if number_of_corr_greater_for_first_var > number_of_corr_greater_for_second_var:
        corr_matrix = corr_matrix.drop(corr_matrix.columns[pair[0]], axis=1)
        corr_matrix = corr_matrix.drop(corr_matrix.index[pair[0]], axis=0)
    else:
        corr_matrix = corr_matrix.drop(corr_matrix.columns[pair[1]], axis=1)
        corr_matrix = corr_matrix.drop(corr_matrix.index[pair[1]], axis=0)

feat_before = tasks_df.shape[1]
print('Number of features before dropping: ', feat_before)

# Dropping the highly correlated features
tasks_df = tasks_df[corr_matrix.columns]

feat_after = tasks_df.shape[1]
print('Number of features before dropping: ', feat_after)
```

```
print(feat_before - feat_after, 'features were dropped')
```

```
Number of features before dropping:  83
Number of features before dropping:  23
60 features were dropped
```

In [29]:

```
# Generate a mask for the upper triangle; True = do NOT show
mask = np.zeros_like(corr_matrix, dtype=bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(20, 20))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(
    corr_matrix,          # The data to plot
    mask=mask,     # Mask some cells
    cmap=cmap,     # What colors to plot the heatmap as
    annot=True,    # Should the values be plotted in the cells?
    vmax=1,       # The maximum value of the legend. All higher vals will be same color
    vmin=0,      # The minimum value of the legend. All lower vals will be same color
    center=0.5,      # The center value of the legend. With divergent cmap, where white is
    square=True,   # Force cells to be square
    linewidths=.5, # Width of lines that divide cells
    cbar_kws={"shrink": .1}  # Extra kwargs for the legend; in this case, shrink by 50%
)
plt.title('Correlation Heatmap')
plt.xlabel('Task Features')
plt.ylabel('Task Features')

# Save plot
f.savefig('heatmapcorr_new.png')
```



In [30]:

```
tasks_df.to_csv(r'tasks_cleaned.csv')
```

1.5 Top 20 Suppliers

In [31]:

```
# Sort cost dataset by Task ID and Cost, and get the top 20 for each Task
sorted_cost = cost_df.sort_values(['Task ID','Cost'],ascending=True).groupby('Task ID').head(20)
sorted_cost.set_index('Task ID')
```

Out[31]:

	Supplier ID	Cost
Task ID		
2019-05-30	S57	0.290967
2019-05-30	S40	0.293925
2019-05-30	S24	0.296388
2019-05-30	S56	0.302213
2019-05-30	S32	0.302394
...	...	...
2021-12-22	S53	0.399636
2021-12-22	S14	0.401945
2021-12-22	S5	0.402187
2021-12-22	S25	0.403408
2021-12-22	S9	0.405070

2400 rows × 2 columns

In [32]:

```
# Find out the supplier that never appear in the top 20 of any tasks
supp_to_del = suppliers_df[~suppliers_df.index.isin(sorted_cost['Supplier ID'])].index[0]
supp_to_del
# Only one supplier ID (S2) never appears in the top 20 cost per task ID
```

Out[32]:

```
'S2'
```

In [33]:

```
# Removing the flagged supplier from Cost dataframe
final_cost = cost_df[~(cost_df['Supplier ID']==supp_to_del)]
final_cost
```

Out[33]:

	Task ID	Supplier ID	Cost
0	2019-05-30	S1	0.478219
2	2019-05-30	S3	0.521679
3	2019-05-30	S4	0.307331
4	2019-05-30	S5	0.357689
5	2019-05-30	S6	0.351982
...	...	...	...
7675	2021-12-22	S60	0.410605
7676	2021-12-22	S61	0.410376
7677	2021-12-22	S62	0.407884
7678	2021-12-22	S63	0.420536
7679	2021-12-22	S64	0.423008

7560 rows × 3 columns

In [34]:

```
# Checking that the flagged supplier ID was deleted
final_cost[final_cost['Supplier ID']==supp_to_del].count()
```

```
Out[34]: Task ID        0
         Supplier ID    0
         Cost           0
         dtype: int64
```

```
In [35]: # Removing the flagged supplier from Suppliers dataframe
final_supplier = suppliers_df[~(suppliers_df.index == supp_to_del)]
final_supplier
```

Out[35]:

Supplier ID	SF1	SF2	SF3	SF4	SF5	SF6	SF7	SF8	SF9	SF10	SF11	SF12
S1	-0.818182	-0.052632	-0.052632	-0.818182	-0.818182	-0.818182	0.333333	-0.707317	-0.052632	-0.632653	-0.052632	0.142857
S3	-0.818182	-0.052632	-0.052632	-0.818182	-0.818182	-0.818182	0.333333	-0.707317	-0.052632	-0.632653	-0.052632	0.142857
S4	-0.818182	-0.052632	-0.052632	-0.818182	-0.818182	-0.818182	0.333333	-0.707317	-0.052632	-0.632653	-0.052632	0.142857
S5	-1.000000	-0.052632	-1.000000	1.000000	1.000000	-0.818182	0.333333	-0.707317	-1.000000	-0.632653	1.000000	1.000000
S6	-1.000000	-1.000000	1.000000	-1.000000	1.000000	1.000000	-0.333333	-1.000000	1.000000	-1.000000	1.000000	-1.000000
...	...	...	...	...	...	...	...	...	...	...	...	...
S60	-1.000000	-1.000000	-1.000000	-1.000000	1.000000	-0.818182	0.333333	-1.000000	-1.000000	-0.632653	-1.000000	1.000000
S61	-0.818182	1.000000	-0.052632	-1.000000	-1.000000	-1.000000	-0.333333	-1.000000	1.000000	1.000000	-0.052632	-1.000000
S62	1.000000	-0.052632	1.000000	-1.000000	1.000000	-0.818182	0.333333	1.000000	1.000000	1.000000	-1.000000	1.000000
S63	-0.818182	-0.052632	-0.052632	-0.818182	1.000000	-1.000000	0.333333	-1.000000	1.000000	-1.000000	1.000000	-1.000000
S64	-0.818182	1.000000	-0.052632	1.000000	-1.000000	-1.000000	1.000000	-1.000000	1.000000	1.000000	-0.052632	0.142857

63 rows × 18 columns

```
In [36]: # Checking that the flagged supplier ID was deleted
final_supplier[final_supplier.index==supp_to_del].count()
```

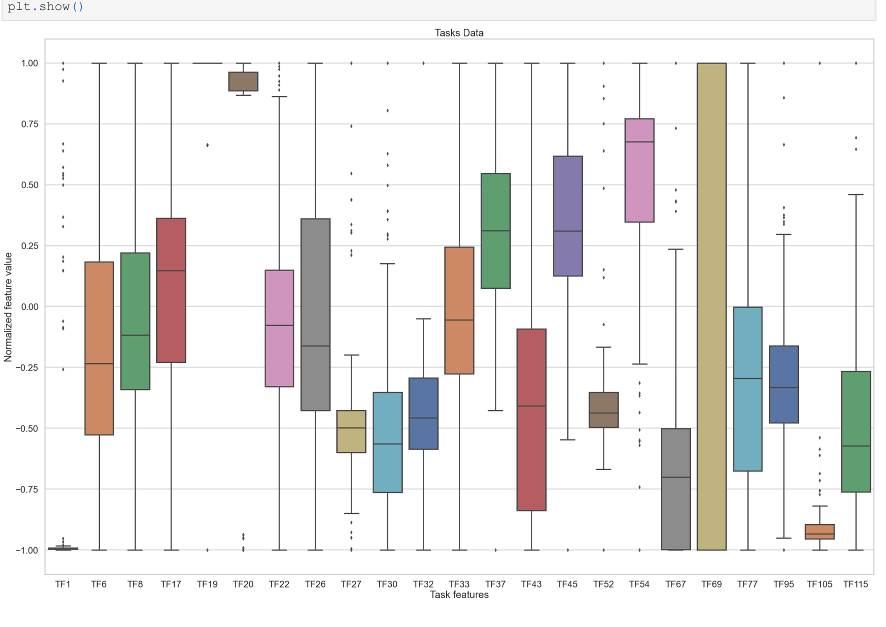
```
Out[36]: SF1     0
SF2     0
SF3     0
SF4     0
SF5     0
SF6     0
SF7     0
SF8     0
SF9     0
SF10    0
SF11    0
SF12    0
SF13    0
SF14    0
SF15    0
SF16    0
SF17    0
SF18    0
dtype: int64
```

```
In [37]: final_cost.to_csv(r'cost_final.csv')
final_supplier.to_csv(r'supplier_final.csv')
```

Exploratory Data Analysis

2.1 Box Plot for Task Dataset

```
In [38]: # Set the figure size
plt.rcParams['figure.figsize'] = [30,20]
plt.rcParams['figure.autolayout'] = True

# Set plot style
sns.set_style('whitegrid')
sns.set_context('poster')

# Load the cleaned Tasks dataframe
tasks_cleaned = pd.read_csv(r'tasks_cleaned.csv').set_index('Task ID')

# Make a boxplot of the Tasks dataframe
sns.boxplot(data=tasks_cleaned, palette='deep')
plt.title('Tasks Data')
plt.xlabel('Task features')
plt.ylabel('Normalized feature value')

# Save plot
plt.savefig(r'boxplot21.png')

# Display the plot
plt.show()
```



2.2 Distribution of Errors

```
In [39]: # Define RMSE function
def rmse(error):
    return np.sqrt(np.sum(error**2)/len(error))
```

```
In [40]: # Load the cleaned Cost dataframe
cost = pd.read_csv(r'cost_final.csv').drop("Unnamed: 0", axis=1)

# Convert the Cost to a wide format
cost_error = cost.pivot(index="Task ID", values="Cost", columns="Supplier ID")

# Get the minimum cost for each task
cost_error['min_costs'] = cost_error.min(axis=1)

# Calculating error distribution for all suppliers
for i in cost_error.columns[:-1]:
    cost_error[i] = cost_error["min_costs"] - cost_error[i]
```

```
In [41]: cost_error.head()
```

Out[41]:

Task ID	S1	S10	S11	S12	S13	S14	S15	S16	S17	S18	...	S6	S60
2019-05-30	-0.187252	-0.040986	-0.043396	-0.022542	-0.018435	-0.029696	-0.016725	-0.135365	-0.058547	-0.035238	...	-0.061014	-0.170021
2019-09-26	-0.124033	-0.043402	-0.048565	-0.041030	-0.026667	-0.005421	-0.040261	-0.037321	-0.031454	-0.023414	...	-0.020424	-0.048707
2019-11-29	-0.153309	-0.094888	-0.024619	-0.018330	-0.020813	-0.069651	-0.028824	-0.043895	-0.074065	-0.074915	...	-0.020996	-0.018257
2020-01-03	-0.163084	-0.037584	-0.025479	-0.035677	-0.028765	-0.046851	-0.032314	-0.074921	-0.056704	-0.067153	...	-0.021011	-0.023527
2020-01-07	-0.149066	-0.016547	-0.023344	-0.013520	-0.018434	-0.031524	-0.009770	-0.015512	-0.054461	-0.017052	...	-0.023803	-0.028648

5 rows × 64 columns

```
In [42]: rmse_df = pd.DataFrame(
    [rmse(cost_error.iloc[:,:-1])]
    , columns=cost_error.columns[:-1]).round(2)
rmse_df
```

Out[42]:

Supplier ID	S1	S10	S11	S12	S13	S14	S15	S16	S17	S18	...	S59	S6	S60	S61	S62	S63	S64	S7	S8	S9
0	0.08	0.04	0.03	0.05	0.04	0.04	0.03	0.06	0.06	0.05	...	0.06	0.04	0.05	0.07	0.06	0.05	0.04	0.05	0.05	0.05

1 rows × 63 columns

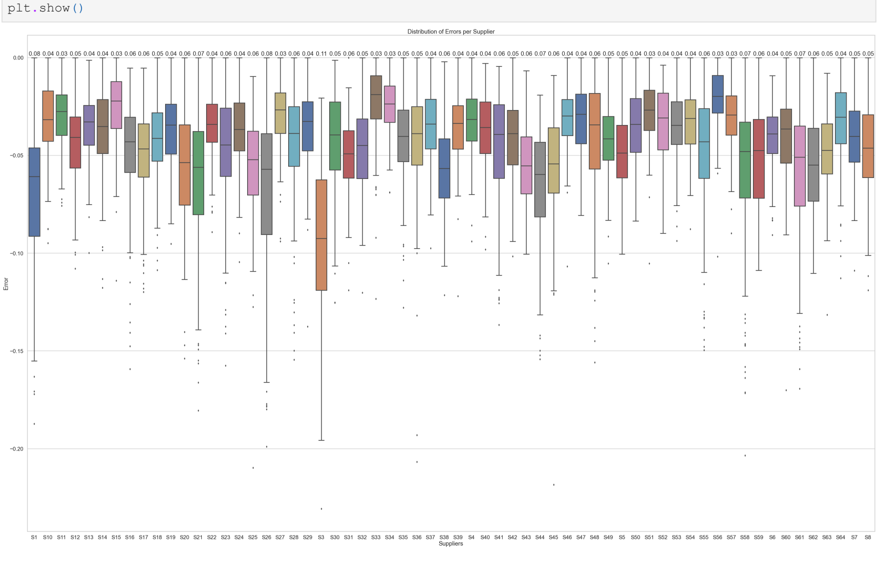
```
In [43]: cost_error
```

Out[43]:

Task ID	S1	S10	S11	S12	S13	S14	S15	S16	S17	S18	...	S6	S60
2019-05-30	-0.187252	-0.040986	-0.043396	-0.022542	-0.018435	-0.029696	-0.016725	-0.135365	-0.058547	-0.035238	...	-0.061014	-0.170021
2019-09-26	-0.124033	-0.043402	-0.048565	-0.041030	-0.026667	-0.005421	-0.040261	-0.037321	-0.031454	-0.023414	...	-0.020424	-0.048707
2019-11-29	-0.153309	-0.094888	-0.024619	-0.018330	-0.020813	-0.069651	-0.028824	-0.043895	-0.074065	-0.074915	...	-0.020996	-0.018257
2020-01-03	-0.163084	-0.037584	-0.025479	-0.035677	-0.028765	-0.046851	-0.032314	-0.074921	-0.056704	-0.067153	...	-0.021011	-0.023527
2020-01-07	-0.149066	-0.016547	-0.023344	-0.013520	-0.018434	-0.031524	-0.009770	-0.015512	-0.054461	-0.017052	...	-0.023803	-0.028648
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2021-12-15	-0.057877	-0.070998	-0.049004	-0.030332	-0.034817	-0.036760	0.000000	-0.054978	-0.040310	-0.049366	...	-0.082069	-0.060973
2021-12-16	-0.084783	-0.073476	-0.022108	-0.071779	-0.075069	-0.032654	-0.034987	-0.016336	-0.046831	-0.042135	...	-0.076117	-0.047087
2021-12-17	-0.048239	-0.036801	-0.021518	-0.011356	-0.068181	-0.015996	-0.041267	-0.018725	-0.010264	-0.056518	...	-0.064575	-0.029553
2021-12-21	-0.070192	-0.016275	-0.032618	-0.007420	-0.015155	-0.038370	-0.028008	-0.018738	-0.031430	-0.065112	...	-0.017190	-0.026539
2021-12-22	-0.108142	-0.033175	-0.041478	-0.009155	-0.044339	-0.024077	-0.036466	-0.017151	-0.046358	-0.044335	...	-0.034129	-0.032737

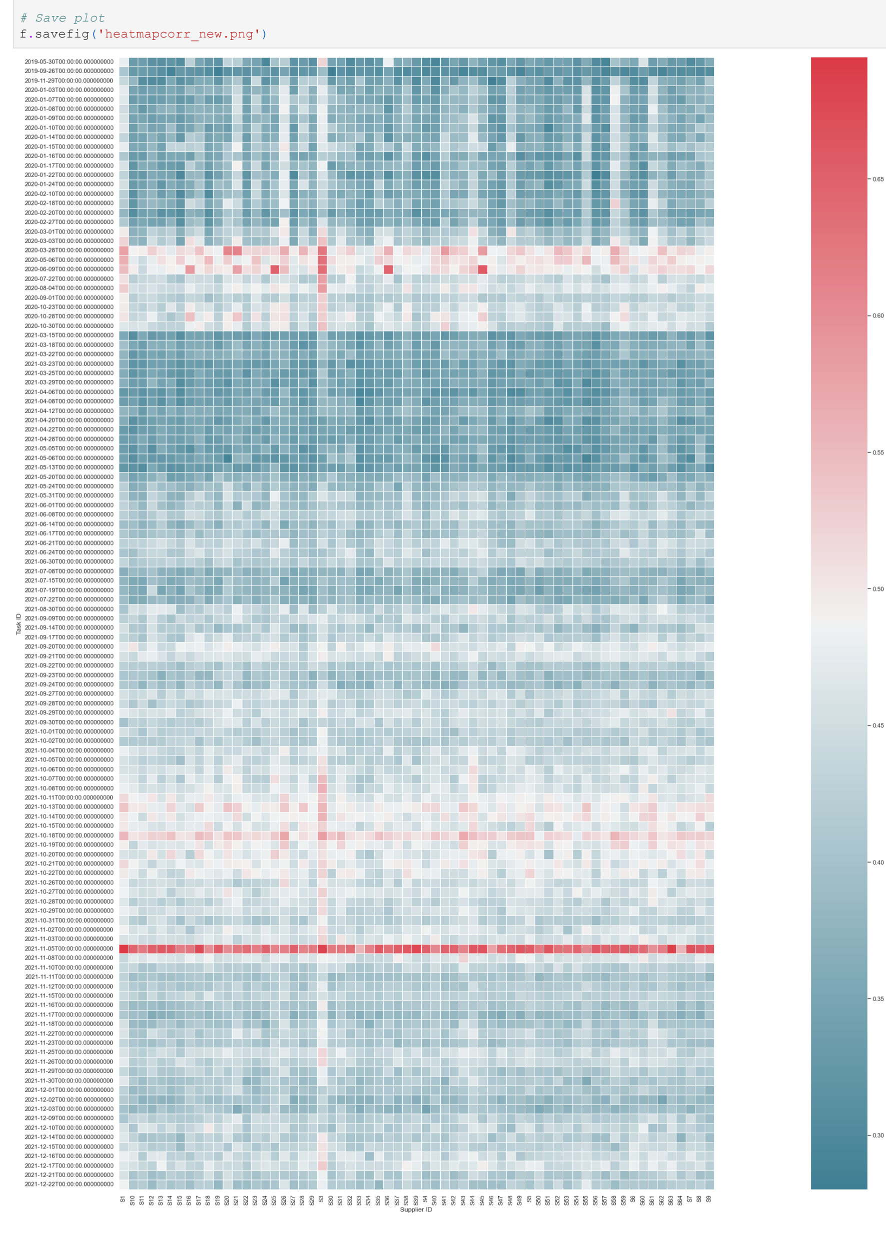
120 rows × 64 columns

[illegible]



2.3 Heatmap

[illegible]



3. ML Model Fitting and Scoring (Random Forest Regressor)

3.1 Combine Data

```
In [48]: # Load the cleaned datasets
cost_df = pd.read_csv(r'cost_final.csv').drop('Unnamed: 0', axis=1)
tasks_df = pd.read_csv(r'tasks_cleaned.csv')
suppliers_df = pd.read_csv(r'supplier_final.csv')
```

```
In [49]: cost_df
```

Out[49]:

	Task ID	Supplier ID	Cost
0	2019-05-30	S1	0.478219
1	2019-05-30	S3	0.521679
2	2019-05-30	S4	0.307331
3	2019-05-30	S5	0.357689
4	2019-05-30	S6	0.351982
...	...	...	...
7555	2021-12-22	S60	0.410605
7556	2021-12-22	S61	0.410376
7557	2021-12-22	S62	0.407884
7558	2021-12-22	S63	0.420536
7559	2021-12-22	S64	0.423008

7560 rows × 3 columns

```
In [50]: tasks_df
```

Out[50]:

	Task ID	TF1	TF6	TF8	TF17	TF19	TF20	TF22	TF26	TF27	...	TF43	TF45	TF52
0	2019-05-30	1.000000	-0.821200	-0.454300	-0.350672	1.000000	0.961792	0.790235	0.547957	-0.599289	...	-0.142857	0.608042	-0.444986
1	2019-09-26	0.974504	-1.000000	-0.786583	-0.560807	0.662732	0.961792	0.386722	0.444928	-0.599289	...	-0.428571	0.214577	-0.380919
2	2019-11-29	-0.257790	-0.183833	-0.291310	-0.642854	0.662732	-0.951252	0.315422	0.557054	-0.599289	...	-0.061224	0.519829	-0.358635
3	2020-01-03	0.328612	-0.174803	-0.212967	-0.627916	0.662732	0.886693	0.259623	0.839891	-0.599289	...	-0.061224	0.581756	-0.288997
4	2020-01-07	0.572238	-0.417667	-0.351643	-0.624852	0.662732	0.886693	0.291139	0.842061	-0.599289	...	-0.142857	0.560783	-0.291783
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
115	2021-12-15	-0.997167	-0.600934	-0.108510	0.458689	1.000000	-0.951252	0.172824	-0.751655	-0.411680	...	-0.918367	0.497116	-0.481894
116	2021-12-16	-0.997167	-0.418980	0.002251	0.445513	1.000000	-0.951252	0.094291	-0.590019	-0.411680	...	-0.877551	0.551804	-0.447772
117	2021-12-17	-0.997167	-0.387297	0.010356	0.439078	1.000000	-0.951252	0.090674	-0.535657	-0.414927	...	-0.877551	0.636270	-0.444986
118	2021-12-21	-0.997167	-0.385009	-0.017560	0.494542	1.000000	-0.951252	-0.058125	-0.525439	0.547029	...	-0.877551	0.596657	-0.465877
119	2021-12-22	-0.997167	-0.253683	0.062584	0.379247	1.000000	-0.951252	-0.103591	-0.541969	-0.198464	...	-0.836735	0.644860	-0.474234

120 rows × 24 columns

```
In [51]: suppliers_df
```

Out[51]:

	Supplier ID	SF1	SF2	SF3	SF4	SF5	SF6	SF7	SF8	SF9	SF10	SF11	SF12
0	S1	-0.818182	-0.052632	-0.052632	-0.818182	-0.818182	-0.818182	0.333333	-0.707317	-0.052632	-0.632653	-0.052632	0.142857
1	S3	-0.818182	-0.052632	-0.052632	-0.818182	-0.818182	-0.818182	0.333333	-0.707317	-0.052632	-0.632653	-0.052632	0.142857
2	S4	-0.818182	-0.052632	-0.052632	-0.818182	-0.818182	-0.818182	0.333333	-0.707317	-0.052632	-0.632653	-0.052632	0.142857
3	S5	-1.000000	-0.052632	-1.000000	1.000000	1.000000	-0.818182	0.333333	-0.707317	-1.000000	-0.632653	1.000000	1.000000
4	S6	-1.000000	-1.000000	1.000000	-1.000000	1.000000	1.000000	-0.333333	-1.000000	1.000000	-1.000000	1.000000	-1.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
58	S60	-1.000000	-1.000000	-1.000000	-1.000000	1.000000	-0.818182	0.333333	-1.000000	-1.000000	-0.632653	-1.000000	1.000000
59	S61	-0.818182	1.000000	-0.052632	-1.000000	-1.000000	-1.000000	-0.333333	-1.000000	1.000000	1.000000	-0.052632	-1.000000
60	S62	1.000000	-0.052632	1.000000	-1.000000	1.000000	-0.818182	0.333333	1.000000	1.000000	1.000000	-1.000000	1.000000
61	S63	-0.818182	-0.052632	-0.052632	-0.818182	1.000000	-1.000000	0.333333	-1.000000	1.000000	-1.000000	1.000000	-1.000000
62	S64	-0.818182	1.000000	-0.052632	1.000000	-1.000000	-1.000000	1.000000	-1.000000	1.000000	1.000000	-0.052632	0.142857

63 rows × 19 columns

```
In [52]: # Convert the Task ID to datetime format in both dataframes
cost_df['Task ID'] = pd.to_datetime(cost_df['Task ID'], infer_datetime_format=True)
tasks_df['Task ID'] = pd.to_datetime(tasks_df['Task ID'], infer_datetime_format=True)

# Merge all dataframes
full_data = pd.merge(cost_df, tasks_df, on='Task ID', how='left')
full_data = pd.merge(full_data, suppliers_df, on='Supplier ID', how='left')

full_data.drop('Supplier ID', axis=1, inplace=True)
full_data
```

Out[52]:

	Task ID	Cost	TF1	TF6	TF8	TF17	TF19	TF20	TF22	TF26	...	SF9	SF10	SF11
0	2019-05-30	0.478219	1.000000	-0.821200	-0.454300	-0.350672	1.0	0.961792	0.790235	0.547957	...	-0.052632	-0.632653	-0.052632
1	2019-05-30	0.521679	1.000000	-0.821200	-0.454300	-0.350672	1.0	0.961792	0.790235	0.547957	...	-0.052632	-0.632653	-0.052632
2	2019-05-30	0.307331	1.000000	-0.821200	-0.454300	-0.350672	1.0	0.961792	0.790235	0.547957	...	-0.052632	-0.632653	-0.052632
3	2019-05-30	0.357689	1.000000	-0.821200	-0.454300	-0.350672	1.0	0.961792	0.790235	0.547957	...	-1.000000	-0.632653	1.000000
4	2019-05-30	0.351982	1.000000	-0.821200	-0.454300	-0.350672	1.0	0.961792	0.790235	0.547957	...	1.000000	-1.000000	1.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7555	2021-12-22	0.410605	-0.997167	-0.253683	0.062584	0.379247	1.0	-0.951252	-0.103591	-0.541969	...	-1.000000	-0.632653	-1.000000
7556	2021-12-22	0.410376	-0.997167	-0.253683	0.062584	0.379247	1.0	-0.951252	-0.103591	-0.541969	...	1.000000	1.000000	-0.052632
7557	2021-12-22	0.407884	-0.997167	-0.253683	0.062584	0.379247	1.0	-0.951252	-0.103591	-0.541969	...	1.000000	1.000000	-1.000000
7558	2021-12-22	0.420536	-0.997167	-0.253683	0.062584	0.379247	1.0	-0.951252	-0.103591	-0.541969	...	1.000000	-1.000000	1.000000
7559	2021-12-22	0.423008	-0.997167	-0.253683	0.062584	0.379247	1.0	-0.951252	-0.103591	-0.541969	...	1.000000	1.000000	-0.052632

7560 rows × 43 columns

```
In [53]: X = full_data.iloc[:, 2:]
y = full_data['Cost']
groups = full_data['Task ID']
```

```
In [54]: X
```

Out[54]:

	TF1	TF6	TF8	TF17	TF19	TF20	TF22	TF26	TF27	TF30	...	SF9	SF10	SF11
0	1.000000	-0.821200	-0.454300	-0.350672	1.0	0.961792	0.790235	0.547957	-0.599289	-0.762431	...	-0.052632	-0.632653	-0.052632
1	1.000000	-0.821200	-0.454300	-0.350672	1.0	0.961792	0.790235	0.547957	-0.599289	-0.762431	...	-0.052632	-0.632653	-0.052632
2	1.000000	-0.821200	-0.454300	-0.350672	1.0	0.961792	0.790235	0.547957	-0.599289	-0.762431	...	-0.052632	-0.632653	-0.052632
3	1.000000	-0.821200	-0.454300	-0.350672	1.0	0.961792	0.790235	0.547957	-0.599289	-0.762431	...	-1.000000	-0.632653	1.000000
4	1.000000	-0.821200	-0.454300	-0.350672	1.0	0.961792	0.790235	0.547957	-0.599289	-0.762431	...	1.000000	-1.000000	1.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7555	-0.997167	-0.253683	0.062584	0.379247	1.0	-0.951252	-0.103591	-0.541969	-0.198464	-0.762431	...	-1.000000	-0.632653	-1.000000
7556	-0.997167	-0.253683	0.062584	0.379247	1.0	-0.951252	-0.103591	-0.541969	-0.198464	-0.762431	...	1.000000	1.000000	-0.052632
7557	-0.997167	-0.253683	0.062584	0.379247	1.0	-0.951252	-0.103591	-0.541969	-0.198464	-0.762431	...	1.000000	1.000000	-1.000000
7558	-0.997167	-0.253683	0.062584	0.379247	1.0	-0.951252	-0.103591	-0.541969	-0.198464	-0.762431	...	1.000000	-1.000000	1.000000
7559	-0.997167	-0.253683	0.062584	0.379247	1.0	-0.951252	-0.103591	-0.541969	-0.198464	-0.762431	...	1.000000	1.000000	-0.052632

7560 rows × 41 columns

```
In [55]: y
```

```
Out[55]: 0       0.478219
1       0.521679
2       0.307331
3       0.357689
4       0.351982
          ...
7555    0.410605
7556    0.410376
7557    0.407884
7558    0.420536
7559    0.423008
Name: Cost, Length: 7560, dtype: float64
```

```
In [56]: groups
```

```
Out[56]: 0      2019-05-30
1      2019-05-30
2      2019-05-30
3      2019-05-30
4      2019-05-30
          ...
7555   2021-12-22
7556   2021-12-22
7557   2021-12-22
7558   2021-12-22
7559   2021-12-22
Name: Task ID, Length: 7560, dtype: datetime64[ns]
```

3.2 Train Test Split

```
In [57]: # Get a random set of 20 unique Task ID
TestGroup = random.sample(list(groups.unique()), 20)

# Define train set based on TestGroup
# Train set is data that is not in TestGroup
X_train = X[~(groups.isin(TestGroup))]
y_train = y[~(groups.isin(TestGroup))]

# Define test set based on TestGroup
# Train set is data that is in TestGroup
X_test = X[groups.isin(TestGroup)]
y_test = y[groups.isin(TestGroup)]

# Get the Task ID of each train and test set
groups_train = pd.DataFrame(groups[~(groups.isin(TestGroup))])
groups_test = pd.DataFrame(groups[groups.isin(TestGroup)])
```

3.3 Train RandomForestRegressor

```
In [58]: start=time.time()

rf = RandomForestRegressor(random_state=42)
rf.fit(X_train, y_train)

print("Time spent on training:",(time.time() - start))

print(f"R2 for train set: {rf.score(X_train, y_train)}")
print(f"R2 for test set: {rf.score(X_test, y_test)}")
```

```
Time spent on training: 6.0119404792785645
R2 for train set: 0.9846925602645882
R2 for test set: 0.7547501392228445
```

3.4 Error Calculation

```
In [59]: # Count number of unique suppliers left for analysis
unique_suppliers = suppliers['Supplier ID'].nunique()

# Define error function
def error_func(y_true, y_pred):
    global unique_suppliers

    error = []
    for i in range(0, len(y_true), unique_suppliers):
        min_cost = np.min(y_true[i:i+unique_suppliers])
        chosen_supplier_index = np.argmin(y_pred[i:i+unique_suppliers])
        ind = i+chosen_supplier_index
        cost_of_chosen_supplier = y_true[ind]
        err = min_cost - cost_of_chosen_supplier
        error.append(err)

    if len(error) == 1:
        error = error[0]

    return error
```

```
In [60]: # RMSE for Random Forest on TRAIN DATA
y_train_pred = np.array(rf.predict(X_train))

errors1 = np.array(error_func(np.array(y_train), y_train_pred))
rmse1 = rmse(errors1)
rmse1
```

```
Out[60]: 0.003120026722821789
```

```
In [61]: # RMSE for Tandom Forest on TEST DATA
y_test_pred = np.array(rf.predict(X_test))

errors2 = np.array(error_func(np.array(y_test), y_test_pred))
rmse2 = rmse(errors2)
rmse2
```

```
Out[61]: 0.026042271933740934
```

4. Cross Validation

```
In [62]: # Make scorer
error_score = make_scorer(error_func)
```

```
In [63]: start = time.time()

# Using LeaveOneGroupOut cross validation
logo = LeaveOneGroupOut()

rf = RandomForestRegressor(random_state=42)

# Converting dataset into np.array
X_train = np.array(X_train)
y_train = np.array(y_train)

# Reshape groups_train into a 2D array
groups_train = np.array(groups_train).reshape(len(groups_train),)
scores = cross_val_score(rf, X_train, y_train, cv=logo, groups=groups_train, scoring=error_score, n_jobs=-1)

print('Mean cross validation score:', np.mean(scores))
print('Time spent on cross validation:', time.time() - start)
print('\nAll cross validation scores:', scores)
```

```
Mean cross validation score: -0.022818062760000007
Time spent on cross validation: 136.7778992652893

All cross validation scores: [-0.01124595 -0.03803286 -0.05562184  0.         -0.02184067 -0.03325926
 -0.00102065 -0.02254393 -0.00157561  0.         -0.01040851 -0.00535117
 -0.01289691 -0.00069682 -0.00463188 -0.01500627 -0.0129736  -0.03559386
 -0.02652747 -0.00645857 -0.02655181 -0.01204386 -0.03649216 -0.00696724
 -0.00638887 -0.02236745  0.          0.         -0.01986231 -0.01062467
 -0.02565542 -0.00858581 -0.01545025 -0.03901925 -0.02589468 -0.01238402
 -0.01249155 -0.0173441  -0.01127459 -0.01883262 -0.00093479 -0.01197867
  0.         -0.02216972 -0.00763493 -0.02306956 -0.01660038 -0.06067611
  0.         -0.05510352 -0.0345263  -0.03406946 -0.03055939 -0.02877947
 -0.03952067 -0.02216265 -0.05199849 -0.05359752 -0.0080816  -0.00029611
 -0.04144735 -0.01027856  0.         -0.04182415 -0.00891985 -0.0188442
 -0.04253061 -0.0299827  -0.02986401 -0.03945351 -0.02848035 -0.00751416
 -0.00392992 -0.02512727 -0.04160327 -0.02758741 -0.03340309 -0.01871617
 -0.01674036 -0.10176827  0.         -0.0122313  -0.02801371 -0.01057142
 -0.01982092 -0.03907324 -0.04037035 -0.02951274 -0.04473348 -0.02505996
 -0.04928178 -0.03419701 -0.03523177 -0.02450455 -0.02955073 -0.04470614
 -0.03346071 -0.03133363 -0.02481582 -0.011646  ]
```

```
In [64]: # RMSE SCORE for Cross Validation - Random Forest
rmse3 = rmse(scores)
rmse3
```

```
Out[64]: 0.028564034293125765
```

5. Hyper-parameter optimization

```
In [263…: # Define the hyperparameter grid that we want to optimize for
param_grid = {"n_estimators":[50,100,200],
             "max_depth":[10,100,None],
             "max_features":["sqrt","auto",None],
             "min_samples_leaf":[1,5,10]}
```

```
In [264…: # Fitting the GridSearchCV
gs = GridSearchCV(rf, param_grid=param_grid, cv=logo, scoring=error_score, n_jobs=12)
gs.fit(X_train, y_train, groups=groups_train)
```

```
Out[264]: GridSearchCV(cv=LeaveOneGroupOut(),
             estimator=RandomForestRegressor(random_state=42), n_jobs=12,
             param_grid={'max_depth': [10, 100, None],
                         'max_features': ['sqrt', 'auto', None],
                         'min_samples_leaf': [1, 5, 10],
                         'n_estimators': [50, 100, 200]},
             scoring=make_scorer(error_func))
```

```
In [146…: # Printing GridSearchCV results
try:
    print('Best parameters:', gs.best_params_)
    print('\nBest score:', gs.best_score_)
    print('\nBest estimator:', gs.best_estimator_)
    print('\nCV results:', gs.cv_results_)
except:
    print("-")
```

```
Best parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'n_estimators': 50}

Best score: -0.019777742560000002

Best estimator: RandomForestRegressor(max_depth=10, max_features='sqrt', n_estimators=50,
                      random_state=42)

CV results: {'mean_fit_time': array([0.25450912, 0.49480661, 0.97300754, 0.22881774, 0.44917853,
       0.90081075, 0.21774773, 0.43503657, 0.85730696, 1.18225791,
       2.35766421, 4.68371303, 1.12014398, 2.24475378, 4.48740628,
       1.06485192, 2.13423187, 4.27811794, 1.17994403, 2.35836241,
       4.68773232, 1.11831903, 2.23769496, 4.45550358, 1.0634556 ,
       2.13843049, 4.24663192, 0.42114352, 0.8548438 , 1.71298854,
       0.28294337, 0.53642525, 1.06323622, 0.23849194, 0.48402542,
       0.93937777, 1.89186004, 3.79168891, 7.6124983 , 1.34143209,
       2.66349949, 5.30550639, 1.17013041, 2.31523767, 4.64062834,
       1.89812304, 3.79381292, 7.55913244, 1.33208688, 2.64969322,
       5.30608842, 1.156806  , 2.31217587, 4.6149469 , 0.42727688,
       0.8594608 , 1.70236677, 0.26951909, 0.53561727, 1.08096864,
       0.24263083, 0.47106027, 0.94517221, 1.90901416, 3.83950098,
       7.57266647, 1.32792829, 2.65062086, 5.30527076, 1.16429601,
       2.31253478, 4.63799544, 1.8968566 , 3.7728391 , 7.63436134,
       1.34454379, 2.67311059, 5.30082273, 1.16863437, 2.31845919,
       4.61593431]), 'std_fit_time': array([0.02572797, 0.03115182, 0.03389966, 0.01284218, 0.01205593,
       0.02226759, 0.00930284, 0.01759742, 0.01851341, 0.02486555,
       0.06029326, 0.06649113, 0.03880651, 0.06926553, 0.10009853,
       0.03879672, 0.06132944, 0.14181557, 0.03853367, 0.06669252,
       0.10081642, 0.03288968, 0.06600782, 0.09903923, 0.02404658,
       0.11423196, 0.07178139, 0.01469698, 0.03598804, 0.07745403,
       0.04220427, 0.03504913, 0.03577874, 0.00880332, 0.03374415,
       0.03521832, 0.02920695, 0.1260253 , 0.18953765, 0.06519232,
       0.06970752, 0.1401635 , 0.03990877, 0.0551607 , 0.1181979 ,
       0.05429663, 0.09413791, 0.12306979, 0.03989906, 0.06946045,
       0.10644832, 0.02193068, 0.06527165, 0.08969332, 0.03691469,
       0.0648614 , 0.04229948, 0.01150624, 0.03506503, 0.07775512,
       0.01478967, 0.01593911, 0.04007607, 0.06809963, 0.19643126,
       0.13312129, 0.04145914, 0.0506964 , 0.12673236, 0.04364988,
       0.06500554, 0.11568269, 0.04373119, 0.07511613, 0.21104653,
       0.05267548, 0.07929168, 0.10255407, 0.04641568, 0.05875381,
       0.0904555 ]), 'mean_score_time': array([0.00318147, 0.00582442, 0.0105817 , 0.00301197, 0.00565499,
       0.01049182, 0.00295221, 0.00548519, 0.01027239, 0.00311168,
       0.00564492, 0.0105818 , 0.00297187, 0.00558512, 0.01063799,
       0.00293207, 0.00564501, 0.01067145, 0.0028723 , 0.00548548,
       0.01056179, 0.00303179, 0.0055154 , 0.01059161, 0.00290215,
       0.00547545, 0.01053192, 0.00329145, 0.00608387, 0.01153907,
       0.00313145, 0.00562492, 0.01093083, 0.00306212, 0.00558504,
       0.01035208, 0.00323135, 0.00597396, 0.01183003, 0.00304178,
       0.00564158, 0.01082093, 0.00300186, 0.00556518, 0.01061172,
       0.00331127, 0.006014  , 0.01175871, 0.00306182, 0.00563497,
       0.01091105, 0.00294194, 0.00553521, 0.01072134, 0.0032314 ,
       0.00619347, 0.01168866, 0.00307176, 0.00558526, 0.01098061,
       0.002982  , 0.00557502, 0.01048192, 0.00329122, 0.0062035 ,
       0.01145952, 0.00308177, 0.00578432, 0.0110804 , 0.00296202,
       0.00557515, 0.01044216, 0.00321146, 0.00614349, 0.01185834,
       0.00316162, 0.00579454, 0.01096054, 0.0030618 , 0.00553517,
       0.01067154]), 'std_score_time': array([0.00057643, 0.00117782, 0.00108249, 0.00061446, 0.00111105,
       0.00137831, 0.00058008, 0.00090863, 0.0007122 , 0.00053292,
       0.00082509, 0.00129964, 0.00058121, 0.0009032 , 0.00139061,
       0.00054302, 0.00099956, 0.00148934, 0.00051399, 0.000726  ,
       0.00124648, 0.00056276, 0.00094043, 0.0013803 , 0.00044818,
       0.00093023, 0.00142656, 0.00060648, 0.0010605 , 0.0012478 ,
       0.00069111, 0.00091968, 0.00173271, 0.00042897, 0.00095649,
       0.00090079, 0.00052995, 0.00069803, 0.00164072, 0.00043186,
       0.00092097, 0.00131842, 0.00055526, 0.00086006, 0.00124973,
       0.00061287, 0.00068303, 0.00157872, 0.0005508 , 0.00087373,
       0.00132919, 0.00038301, 0.00081498, 0.00125665, 0.00054847,
       0.00112212, 0.00155203, 0.00043869, 0.0008808 , 0.00172456,
       0.00051831, 0.00079889, 0.0011416 , 0.00055539, 0.00102275,
       0.0010026 , 0.00059994, 0.00092501, 0.00158887, 0.00041011,
       0.00076082, 0.00105025, 0.00047997, 0.00097429, 0.00163811,
       0.00056519, 0.00085443, 0.00146918, 0.00058582, 0.00078995,
       0.00142791]), 'param_max_depth': masked_array(data=[10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
10,
                   10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
                   100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
                   100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
                   100, 100, 100, 100, 100, None, None, None, None, None,
                   None, None, None, None, None, None, None, None, None,
                   None, None, None, None, None, None, None, None, None,
                   None, None, None, None],
             mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False],
       fill_value='?',
            dtype=object), 'param_max_features': masked_array(data=['sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sq
rt', 'sqrt',
                   'sqrt', 'sqrt', 'auto', 'auto', 'auto', 'auto', 'auto',
                   'auto', 'auto', 'auto', 'auto', None, None, None, None,
                   None, None, None, None, None, 'sqrt', 'sqrt', 'sqrt',
                   'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'auto',
                   'auto', 'auto', 'auto', 'auto', 'auto', 'auto', 'auto',
                   'auto', None, None, None, None, None, None, None, None,
                   None, 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                   'sqrt', 'sqrt', 'sqrt', 'auto', 'auto', 'auto',
                   'auto', 'auto', 'auto', 'auto', 'auto', None, None,
                   None, None, None, None, None, None, None],
             mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False],
[illegible]
       -0.01124595, -0.03314564, -0.01124595, -0.01124595, -0.03314564,
       -0.01124595, -0.01124595, -0.01124595, -0.01124595, -0.01124595,
       -0.03314564, -0.01124595, -0.01124595, -0.03314564, -0.01124595,
       -0.01124595, -0.01124595, -0.01124595, -0.01124595, -0.01124595,
       -0.01124595,  0.        , -0.01124595, -0.01124595, -0.01124595,
        0.        , -0.01124595, -0.01124595, -0.03314564, -0.01124595,
       -0.01124595, -0.03314564, -0.01124595, -0.01124595, -0.01124595,
       -0.01124595, -0.01124595, -0.03314564, -0.01124595, -0.01124595,
       -0.03314564, -0.01124595, -0.01124595, -0.01124595, -0.01124595,
       -0.01124595]), 'split1_test_score': array([-0.0090714 , -0.03026619, -0.0090714 , -0.03026619, -0.030266
19,
       -0.03026619, -0.04715313, -0.04715313, -0.03803286, -0.03803286,
       -0.03803286, -0.03803286, -0.03803286, -0.03803286, -0.03803286,
       -0.03803286, -0.03803286, -0.03803286, -0.03803286, -0.03803286,
       -0.03803286, -0.03803286, -0.03803286, -0.03803286, -0.03803286,
       -0.03803286, -0.03803286, -0.0090714 , -0.03026619, -0.03026619,
       -0.04715313, -0.03026619, -0.03026619, -0.04715313, -0.04715313,
       -0.04715313, -0.03803286, -0.03803286, -0.03803286, -0.03803286,
       -0.03803286, -0.03803286, -0.03803286, -0.03803286, -0.03803286,
       -0.03803286, -0.03803286, -0.03803286, -0.03803286, -0.03803286,
       -0.03803286, -0.03803286, -0.03803286, -0.03803286, -0.0090714 ,
       -0.03026619, -0.03026619, -0.04715313, -0.03026619, -0.03026619,
       -0.04715313, -0.04715313, -0.04715313, -0.03803286, -0.03803286,
       -0.03803286, -0.03803286, -0.03803286, -0.03803286, -0.03803286,
       -0.03803286, -0.03803286, -0.03803286, -0.03803286, -0.03803286,
       -0.03803286, -0.03803286, -0.03803286, -0.03803286, -0.03803286,
       -0.03803286]), 'split2_test_score': array([-0.02419869, -0.02419869, -0.02419869, -0.07491534, -0.024198
69,
       -0.02419869, -0.09488841, -0.02419869, -0.02419869, -0.02882436,
       -0.12338109, -0.12338109, -0.05562184, -0.02882436, -0.02882436,
       -0.03925203, -0.03925203, -0.02419869, -0.02882436, -0.12338109,
       -0.12338109, -0.05562184, -0.02882436, -0.02882436, -0.03925203,
       -0.03925203, -0.02419869, -0.02882436, -0.02419869, -0.02419869,
       -0.05562184, -0.02419869, -0.02419869, -0.02419869, -0.02419869,
       -0.02419869, -0.05562184, -0.05562184, -0.05562184, -0.05562184,
       -0.02882436, -0.02882436, -0.03925203, -0.03925203, -0.05562184,
       -0.05562184, -0.05562184, -0.05562184, -0.05562184, -0.02882436,
       -0.02882436, -0.03925203, -0.03925203, -0.05562184, -0.05562184,
       -0.02419869, -0.02419869, -0.05562184, -0.02419869, -0.02419869,
       -0.02419869, -0.02419869, -0.02419869, -0.05562184, -0.05562184,
       -0.05562184, -0.05562184, -0.02882436, -0.02882436, -0.03925203,
       -0.03925203, -0.05562184, -0.05562184, -0.05562184, -0.05562184,
       -0.05562184, -0.02882436, -0.02882436, -0.03925203, -0.03925203,
       -0.05562184]), 'split3_test_score': array([ 0.        ,  0.        ,  0.        , -0.03231384,  0.
 ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        , -0.01403033,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        , -0.01403033,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ]), 'split4_test_score': array([ 0.        ,  0.        ,  0.        , -0.02184067, -0.021840
67,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
       -0.02686325,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        , -0.02686325,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        , -0.02184067,  0.        ,
        0.        , -0.02184067, -0.02184067,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
       -0.02184067, -0.02184067,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
       -0.02184067,  0.        ,  0.        , -0.02184067, -0.02184067,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        , -0.02184067, -0.02184067,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ]), 'split5_test_score': array([-0.03325926, -0.03325926, -0.03325926, -0.03325926, -0.033259
26,
       -0.03325926, -0.0757475 , -0.03325926, -0.03325926, -0.03325926,
       -0.03325926, -0.03325926, -0.03325926, -0.03325926, -0.03325926,
       -0.03325926, -0.03325926, -0.03325926, -0.03325926, -0.03325926,
       -0.03325926, -0.03325926, -0.03325926, -0.03325926, -0.03325926,
       -0.03325926, -0.03325926, -0.03325926, -0.03325926, -0.03325926,
       -0.03325926, -0.03325926, -0.03325926, -0.03325926, -0.03325926,
       -0.03325926, -0.03325926, -0.03325926, -0.03325926, -0.03325926,
       -0.03325926, -0.03325926, -0.03325926, -0.03325926, -0.03325926,
       -0.03325926, -0.03325926, -0.03325926, -0.03325926, -0.03325926,
       -0.03325926, -0.03325926, -0.03325926, -0.03325926, -0.03325926,
       -0.03325926, -0.03325926, -0.03325926, -0.03325926, -0.03325926,
       -0.03325926, -0.03325926, -0.03325926, -0.03325926, -0.03325926,
       -0.03325926, -0.03325926, -0.03325926, -0.03325926, -0.03325926,
       -0.03325926, -0.03325926, -0.03325926, -0.03325926, -0.03325926,
       -0.03325926, -0.03325926, -0.03325926, -0.03325926, -0.03325926,
       -0.03325926]), 'split6_test_score': array([-0.00102065, -0.00102065, -0.00102065, -0.00102065, -0.001020
65,
       -0.00102065, -0.00102065, -0.00102065, -0.00102065, -0.00102065,
       -0.00102065, -0.00102065, -0.00102065, -0.00102065, -0.00102065,
       -0.00102065, -0.00102065, -0.00102065, -0.00102065, -0.00102065,
       -0.00102065, -0.00102065, -0.00102065, -0.00102065, -0.00102065,
       -0.00102065, -0.00102065, -0.00102065, -0.00102065, -0.00102065,
       -0.00102065, -0.00102065, -0.00102065, -0.00102065, -0.00102065,
       -0.00102065, -0.00102065, -0.00102065, -0.00102065, -0.00102065,
       -0.00102065, -0.00102065, -0.00102065, -0.00102065, -0.00102065,
       -0.00102065, -0.00102065, -0.00102065, -0.00102065, -0.00102065,
       -0.00102065, -0.00102065, -0.00102065, -0.00102065, -0.00102065,
       -0.00102065, -0.00102065, -0.00102065, -0.00102065, -0.00102065,
       -0.00102065, -0.00102065, -0.00102065, -0.00102065, -0.00102065,
       -0.00102065, -0.00102065, -0.00102065, -0.00102065, -0.00102065,
       -0.00102065, -0.00102065, -0.00102065, -0.00102065, -0.00102065,
       -0.00102065, -0.00102065, -0.00102065, -0.00102065, -0.00102065,
       -0.00102065]), 'split7_test_score': array([ 0.        ,  0.        ,  0.        ,  0.        ,  0.
 ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        , -0.02254393, -0.02254393,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        , -0.02254393, -0.02254393, -0.02254393,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
       -0.02254393, -0.02254393, -0.02254393,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
       -0.02254393, -0.02254393,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        , -0.02254393, -0.02254393,
       -0.02254393,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        , -0.02254393, -0.02254393, -0.02254393,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ]), 'split8_test_score': array([-0.00157561, -0.00157561, -0.00157561, -0.00157561, -0.001575
61,
       -0.00157561, -0.02033983, -0.03782271, -0.03782271, -0.00157561,
       -0.00157561, -0.00157561, -0.00157561, -0.00157561, -0.00157561,
       -0.00157561, -0.00157561, -0.00157561, -0.00157561, -0.00157561,
       -0.00157561, -0.00157561, -0.00157561, -0.00157561, -0.00157561,
       -0.00157561, -0.00157561, -0.00157561, -0.00157561, -0.00157561,
       -0.00157561, -0.00157561, -0.00157561, -0.00157561, -0.00157561,
       -0.00157561, -0.00157561, -0.00157561, -0.00157561, -0.00157561,
       -0.00157561, -0.00157561, -0.00157561, -0.00157561, -0.00157561,
       -0.00157561, -0.00157561, -0.00157561, -0.00157561, -0.00157561,
       -0.00157561, -0.00157561, -0.00157561, -0.00157561, -0.00157561,
       -0.00157561, -0.00157561, -0.00157561, -0.00157561, -0.00157561,
       -0.00157561, -0.00157561, -0.00157561, -0.00157561, -0.00157561,
       -0.00157561, -0.00157561, -0.00157561, -0.00157561, -0.00157561,
       -0.00157561, -0.00157561, -0.00157561, -0.00157561, -0.00157561,
       -0.00157561, -0.00157561, -0.00157561, -0.00157561, -0.00157561,
       -0.00157561]), 'split9_test_score': array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]), 'split10_test_score': array([-0.01040851, -0.01040
851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851]), 'split11_test_score': array([-0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00578
242,
       -0.03602054, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117]), 'split12_test_score': array([ 0.        ,  0.        , -0.01289691, -0.01289691, -0.01289
691,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691, -0.01289691,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691, -0.01289691,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691, -0.01289691,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691, -0.01289691,
       -0.01289691, -0.01289691,  0.        , -0.01289691, -0.01289691,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691, -0.01289691,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691, -0.01289691,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691, -0.01289691,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691, -0.01289691,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691,  0.        ,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691, -0.01289691,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691, -0.01289691,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691, -0.01289691,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691, -0.01289691,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691, -0.01289691,
       -0.01289691]), 'split13_test_score': array([-0.01591847, -0.01591847, -0.01591847, -0.00069682, -0.00069
682,
       -0.01591847, -0.00069682, -0.00069682, -0.01591847, -0.01591847,
       -0.01591847, -0.01591847, -0.00069682, -0.00069682, -0.00069682,
       -0.01591847, -0.01591847, -0.01591847, -0.01591847, -0.01591847,
       -0.01591847, -0.00069682, -0.00069682, -0.01591847, -0.01591847,
       -0.01591847, -0.01591847, -0.01591847, -0.01591847, -0.01591847,
       -0.00069682, -0.00069682, -0.00069682, -0.01674642, -0.00069682,
       -0.00069682, -0.00069682, -0.00069682, -0.00069682, -0.00069682,
       -0.00069682, -0.00069682, -0.00069682, -0.00069682, -0.00069682,
       -0.00069682, -0.00069682, -0.00069682, -0.00069682, -0.00069682,
       -0.00069682, -0.00069682, -0.00069682, -0.00069682, -0.01591847,
       -0.01591847, -0.01591847, -0.00069682, -0.00069682, -0.00069682,
       -0.01674642, -0.00069682, -0.00069682, -0.00069682, -0.00069682,
       -0.00069682, -0.00069682, -0.00069682, -0.00069682, -0.00069682,
       -0.00069682, -0.00069682, -0.00069682, -0.00069682, -0.00069682,
       -0.00069682, -0.00069682, -0.00069682, -0.00069682, -0.00069682,
       -0.00069682]), 'split14_test_score': array([ 0.        ,  0.        , -0.02836325, -0.01894686, -0.00463
188,
        0.        , -0.00463188, -0.00463188, -0.00463188, -0.00561272,
       -0.00561272, -0.00561272,  0.        , -0.03453136, -0.00561272,
        0.        ,  0.        ,  0.        , -0.00561272, -0.00561272,
       -0.00561272,  0.        , -0.03453136, -0.00561272,  0.        ,
        0.        ,  0.        , -0.00463188, -0.00463188, -0.00463188,
       -0.00463188, -0.00463188, -0.00463188, -0.00463188, -0.00463188,
       -0.00463188, -0.00561272, -0.00463188, -0.00561272,  0.        ,
       -0.03453136,  0.        , -0.03453136, -0.00463188, -0.00463188,
       -0.00561272, -0.00463188, -0.00561272,  0.        , -0.03453136,
        0.        , -0.03453136, -0.00463188, -0.00463188, -0.00463188,
       -0.00463188, -0.00463188, -0.00463188, -0.00463188, -0.00463188,
       -0.00463188, -0.00463188, -0.00463188, -0.00561272, -0.00463188,
       -0.00561272,  0.        , -0.03453136,  0.        , -0.03453136,
       -0.00463188, -0.00463188, -0.00561272, -0.00463188, -0.00561272,
        0.        , -0.03453136,  0.        , -0.03453136, -0.00463188,
       -0.00463188]), 'split15_test_score': array([-0.01500627, -0.03660663, -0.03660663, -0.01500627, -0.07883
552,
       -0.07883552, -0.01500627, -0.01500627, -0.04596536, -0.01500627,
       -0.01500627, -0.03660663, -0.01500627, -0.01500627, -0.01500627,
       -0.03667882, -0.01500627, -0.04596536, -0.01500627, -0.01500627,
       -0.03660663, -0.01500627, -0.01500627, -0.01500627, -0.03667882,
       -0.01500627, -0.04596536,  0.        , -0.03660663, -0.03660663,
       -0.01500627, -0.03411256, -0.01500627, -0.01500627, -0.03411256,
       -0.01500627, -0.01500627, -0.01500627, -0.03660663, -0.01500627,
       -0.01500627, -0.01500627, -0.01500627, -0.01500627, -0.01500627,
       -0.01500627, -0.01500627, -0.03660663, -0.01500627, -0.01500627,
       -0.01500627, -0.01500627, -0.01500627, -0.01500627,  0.        ,
       -0.03660663, -0.03660663, -0.01500627, -0.03411256, -0.01500627,
       -0.01500627, -0.03411256, -0.01500627, -0.01500627, -0.01500627,
       -0.03660663, -0.01500627, -0.01500627, -0.01500627, -0.01500627,
       -0.01500627, -0.01500627, -0.01500627, -0.01500627, -0.03660663,
       -0.01500627, -0.01500627, -0.01500627, -0.01500627, -0.01500627,
       -0.01500627]), 'split16_test_score': array([-0.0129736 , -0.00130511, -0.0129736 , -0.0129736 , -0.01297
36 ,
       -0.0129736 , -0.04701991, -0.0129736 , -0.03455523, -0.00990789,
       -0.0129736 , -0.04701991, -0.00990789, -0.04701991, -0.04701991,
       -0.00990789, -0.0129736 , -0.04701991, -0.00990789, -0.0129736 ,
       -0.04701991, -0.00990789, -0.04701991, -0.04701991, -0.00990789,
       -0.0129736 , -0.04701991, -0.0129736 , -0.0129736 , -0.0129736 ,
       -0.0129736 , -0.04701991, -0.04701991, -0.0129736 , -0.0129736 ,
       -0.0129736 , -0.0129736 , -0.0129736 , -0.0129736 , -0.0129736 ,
       -0.0129736 , -0.0129736 , -0.00990789, -0.0129736 , -0.04701991,
       -0.0129736 , -0.0129736 , -0.0129736 , -0.0129736 , -0.0129736 ,
       -0.0129736 , -0.00990789, -0.0129736 , -0.04701991, -0.0129736 ,
       -0.0129736 , -0.0129736 , -0.0129736 , -0.04701991, -0.04701991,
       -0.0129736 , -0.0129736 , -0.0129736 , -0.0129736 , -0.0129736 ,
       -0.0129736 , -0.0129736 , -0.0129736 , -0.0129736 , -0.00990789,
       -0.0129736 , -0.04701991, -0.0129736 , -0.0129736 , -0.0129736 ,
       -0.0129736 , -0.0129736 , -0.0129736 , -0.00990789, -0.0129736 ,
       -0.04701991]), 'split17_test_score': array([-0.03559386, -0.03559386, -0.03559386, -0.03559386, -0.03559
386,
       -0.03870327, -0.03559386, -0.03559386, -0.03559386, -0.03559386,
       -0.03559386, -0.03559386, -0.03559386, -0.03559386, -0.03559386,
       -0.02989899, -0.02989899, -0.02989899, -0.03559386, -0.03559386,
       -0.03559386, -0.03559386, -0.03559386, -0.03559386, -0.02989899,
       -0.02989899, -0.02989899, -0.03559386, -0.03559386, -0.03559386,
       -0.03559386, -0.03559386, -0.03870327, -0.03870327, -0.03559386,
       -0.03870327, -0.03559386, -0.03559386, -0.03559386, -0.03559386,
       -0.03559386, -0.03559386, -0.03848251, -0.02989899, -0.03559386,
       -0.03559386, -0.03559386, -0.03559386, -0.03559386, -0.03559386,
       -0.03559386, -0.03848251, -0.02989899, -0.03559386, -0.03559386,
       -0.03559386, -0.03559386, -0.03559386, -0.03559386, -0.03870327,
       -0.03870327, -0.03559386, -0.03870327, -0.03559386, -0.03559386,
       -0.03559386, -0.03559386, -0.03559386, -0.03559386, -0.03848251,
       -0.02989899, -0.03559386, -0.03559386, -0.03559386, -0.03559386,
       -0.03559386, -0.03559386, -0.03559386, -0.03848251, -0.02989899,
       -0.03559386]), 'split18_test_score': array([-0.00628353, -0.00628353, -0.00628353, -0.01485493, -0.01485
493,
       -0.01485493, -0.01485493,  0.        ,  0.        , -0.02652747,
       -0.02652747, -0.02652747, -0.02652747, -0.02652747, -0.02652747,
       -0.02652747, -0.02652747, -0.02652747, -0.02652747, -0.02652747,
       -0.02652747, -0.02652747, -0.02652747, -0.02652747, -0.02652747,
       -0.02652747, -0.02652747,  0.        ,  0.        , -0.02652747,
        0.        ,  0.        ,  0.        ,  0.        , -0.01871144,
        0.        , -0.02652747, -0.02652747, -0.02652747, -0.02652747,
       -0.02652747,  0.        , -0.02652747, -0.02652747, -0.02652747,
       -0.02652747, -0.02652747, -0.02652747, -0.02652747, -0.02652747,
        0.        , -0.02652747, -0.02652747, -0.02652747,  0.        ,
        0.        , -0.02652747,  0.        ,  0.        ,  0.        ,
        0.        , -0.01871144,  0.        , -0.02652747, -0.02652747,
       -0.02652747, -0.02652747, -0.02652747,  0.        , -0.02652747,
       -0.02652747, -0.02652747, -0.02652747, -0.02652747, -0.02652747,
       -0.02652747, -0.02652747,  0.        , -0.02652747, -0.02652747,
       -0.02652747]), 'split19_test_score': array([-0.03020359, -0.01194671, -0.00645857, -0.03020359, -0.03020
359,
       -0.00645857, -0.00645857, -0.00645857, -0.00645857, -0.00645857,
       -0.00645857, -0.01194671, -0.01194671, -0.01194671, -0.01194671,
       -0.01146613, -0.01205083, -0.01194671, -0.00645857, -0.00645857,
       -0.01194671, -0.01194671, -0.01194671, -0.01194671, -0.01146613,
       -0.01205083, -0.01194671, -0.00645857, -0.00645857, -0.00645857,
       -0.02259991, -0.00645857, -0.00645857, -0.00645857, -0.03020359,
       -0.00645857, -0.00645857, -0.00645857, -0.00645857, -0.00645857,
       -0.00645857, -0.00645857, -0.01205083, -0.00645857, -0.01194671,
       -0.00645857, -0.00645857, -0.00645857, -0.00645857, -0.00645857,
       -0.00645857, -0.01205083, -0.00645857, -0.01194671, -0.00645857,
       -0.00645857, -0.00645857, -0.02259991, -0.00645857, -0.00645857,
       -0.00645857, -0.03020359, -0.00645857, -0.00645857, -0.00645857,
       -0.00645857, -0.00645857, -0.00645857, -0.00645857, -0.01205083,
       -0.00645857, -0.01194671, -0.00645857, -0.00645857, -0.00645857,
       -0.00645857, -0.00645857, -0.00645857, -0.01205083, -0.00645857,
       -0.01194671]), 'split20_test_score': array([-0.00918663, -0.03027348, -0.00918663, -0.02068908, -0.00918
663,
       -0.00918663, -0.00918663, -0.01527124, -0.01527124, -0.02068908,
       -0.02068908, -0.02068908, -0.02761438, -0.02761438, -0.02463263,
       -0.00918663, -0.02761438, -0.02463263, -0.02068908, -0.02068908,
       -0.02068908, -0.02761438, -0.02761438, -0.02463263, -0.00918663,
       -0.02761438, -0.02463263, -0.02761438, -0.00918663, -0.00918663,
       -0.03027348, -0.02068908, -0.03027348, -0.02761438, -0.02761438,
       -0.00918663, -0.02655181, -0.02655181, -0.02655181, -0.02463263,
       -0.00918663, -0.00918663, -0.00918663, -0.00918663, -0.00918663,
       -0.02655181, -0.02655181, -0.02655181, -0.02463263, -0.00918663,
       -0.00918663, -0.00918663, -0.00918663, -0.00918663, -0.02761438,
       -0.00918663, -0.00918663, -0.03027348, -0.02068908, -0.03027348,
       -0.02761438, -0.02761438, -0.00918663, -0.02655181, -0.02655181,
       -0.02655181, -0.02463263, -0.00918663, -0.00918663, -0.00918663,
       -0.00918663, -0.00918663, -0.02655181, -0.02655181, -0.02655181,
       -0.02463263, -0.00918663, -0.00918663, -0.00918663, -0.00918663,
       -0.00918663]), 'split21_test_score': array([-0.01204386, -0.01204386, -0.01204386, -0.01204386, -0.01204
386,
       -0.01204386, -0.03961107, -0.01204386, -0.03961107, -0.01512244,
       -0.03689011, -0.01204386, -0.01204386, -0.01204386, -0.01204386,
       -0.01204386, -0.01204386, -0.01204386, -0.01512244, -0.03689011,
       -0.01204386, -0.01204386, -0.01204386, -0.01204386, -0.01204386,
       -0.01204386, -0.01204386, -0.01204386, -0.01204386, -0.01204386,
       -0.01204386, -0.01204386, -0.01204386, -0.01114647, -0.0166982 ,
       -0.01204386, -0.01204386, -0.01204386, -0.01204386, -0.01204386,
       -0.01204386, -0.01204386, -0.01204386, -0.01204386, -0.01204386,
       -0.01204386, -0.01204386, -0.01204386, -0.01204386, -0.01204386,
       -0.01204386, -0.01204386, -0.01204386, -0.01204386, -0.01204386,
       -0.01204386, -0.01204386, -0.01204386, -0.01204386, -0.01204386,
       -0.01114647, -0.0166982 , -0.01204386, -0.01204386, -0.01204386,
       -0.01204386, -0.01204386, -0.01204386, -0.01204386, -0.01204386,
       -0.01204386, -0.01204386, -0.01204386, -0.01204386, -0.01204386,
       -0.01204386, -0.01204386, -0.01204386, -0.01204386, -0.01204386,
       -0.01204386]), 'split22_test_score': array([-0.02903954, -0.04126113, -0.04126113, -0.03649216, -0.03649
216,
       -0.03649216, -0.01450328, -0.03649216, -0.03649216, -0.04126113,
       -0.03649216, -0.03649216, -0.04126113, -0.03649216, -0.03649216,
       -0.03649216, -0.03649216, -0.03649216, -0.04126113, -0.03649216,
       -0.03649216, -0.04126113, -0.03649216, -0.03649216, -0.03649216,
       -0.03649216, -0.03649216, -0.01450328, -0.01450328, -0.01450328,
       -0.040786  , -0.040786  , -0.040786  , -0.01450328, -0.01450328,
       -0.01450328, -0.03649216, -0.03649216, -0.03649216, -0.040786  ,
       -0.040786  , -0.040786  , -0.03649216, -0.03649216, -0.03649216,
       -0.03649216, -0.03649216, -0.03649216, -0.040786  , -0.040786  ,
       -0.040786  , -0.03649216, -0.03649216, -0.03649216, -0.01450328,
       -0.01450328, -0.01450328, -0.040786  , -0.040786  , -0.040786  ,
       -0.01450328, -0.01450328, -0.01450328, -0.03649216, -0.03649216,
       -0.03649216, -0.040786  , -0.040786  , -0.040786  , -0.03649216,
       -0.03649216, -0.03649216, -0.03649216, -0.03649216, -0.03649216,
       -0.040786  , -0.040786  , -0.040786  , -0.03649216, -0.03649216,
       -0.03649216]), 'split23_test_score': array([-0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696
724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724]), 'split24_test_score': array([-0.00638887, -0.00638887, -0.00638887,  0.        ,  0.
 ,
        0.        , -0.00638887,  0.        , -0.00638887,  0.        ,
       -0.00638887, -0.00638887, -0.03107268, -0.03107268, -0.03107268,
        0.        ,  0.        , -0.03107268,  0.        , -0.00638887,
       -0.00638887, -0.03107268, -0.03107268, -0.03107268,  0.        ,
        0.        , -0.03107268,  0.        , -0.00638887, -0.00638887,
        0.        ,  0.        ,  0.        , -0.00638887,  0.        ,
        0.        , -0.00638887, -0.00638887, -0.00638887,  0.        ,
        0.        , -0.03107268,  0.        ,  0.        , -0.03107268,
       -0.00638887, -0.00638887, -0.00638887,  0.        ,  0.        ,
       -0.03107268,  0.        ,  0.        , -0.03107268,  0.        ,
       -0.00638887, -0.00638887,  0.        ,  0.        ,  0.        ,
       -0.00638887,  0.        ,  0.        , -0.00638887, -0.00638887,
       -0.00638887,  0.        ,  0.        , -0.03107268,  0.        ,
        0.        , -0.03107268, -0.00638887, -0.00638887, -0.00638887,
        0.        ,  0.        , -0.03107268,  0.        ,  0.        ,
       -0.03107268]), 'split25_test_score': array([-0.00762855, -0.00762855, -0.00762855, -0.00762855, -0.00762
855,
       -0.00762855, -0.02236745, -0.00762855, -0.00762855, -0.02236745,
       -0.02236745, -0.02236745, -0.00754795, -0.02236745, -0.02236745,
       -0.02236745, -0.02236745, -0.02236745, -0.02236745, -0.02236745,
       -0.02236745, -0.00754795, -0.02236745, -0.02236745, -0.02236745,
       -0.02236745, -0.02236745, -0.02236745, -0.02236745, -0.02236745,
       -0.02236745, -0.02236745, -0.02236745, -0.02236745, -0.02236745,
       -0.02236745, -0.02236745, -0.02236745, -0.02236745, -0.02236745,
       -0.02236745, -0.02236745, -0.02236745, -0.02236745, -0.02236745,
       -0.02236745, -0.02236745, -0.02236745, -0.02236745, -0.02236745,
       -0.02236745, -0.02236745, -0.02236745, -0.02236745, -0.02236745,
       -0.02236745, -0.02236745, -0.02236745, -0.02236745, -0.02236745,
       -0.02236745, -0.02236745, -0.02236745, -0.02236745, -0.02236745,
       -0.02236745, -0.02236745, -0.02236745, -0.02236745, -0.02236745,
       -0.02236745, -0.02236745, -0.02236745, -0.02236745, -0.02236745,
       -0.02236745, -0.02236745, -0.02236745, -0.02236745, -0.02236745,
       -0.02236745]), 'split26_test_score': array([-0.0079833, -0.0079833, -0.0079833,  0.       ,  0.       ,
        0.       ,  0.       ,  0.       ,  0.       , -0.0079833,
       -0.0079833, -0.0079833,  0.       ,  0.       ,  0.       ,
        0.       ,  0.       ,  0.       , -0.0079833, -0.0079833,
       -0.0079833,  0.       ,  0.       ,  0.       ,  0.       ,
        0.       ,  0.       ,  0.       ,  0.       , -0.0079833,
        0.       ,  0.       ,  0.       , -0.0079833,  0.       ,
        0.       ,  0.       ,  0.       ,  0.       ,  0.       ,
        0.       ,  0.       ,  0.       ,  0.       ,  0.       ,
        0.       ,  0.       ,  0.       ,  0.       ,  0.       ,
        0.       ,  0.       ,  0.       ,  0.       ,  0.       ,
        0.       , -0.0079833,  0.       ,  0.       ,  0.       ,
       -0.0079833,  0.       ,  0.       ,  0.       ,  0.       ,
        0.       ,  0.       ,  0.       ,  0.       ,  0.       ,
        0.       ,  0.       ,  0.       ,  0.       ,  0.       ,
        0.       ,  0.       ,  0.       ,  0.       ,  0.       ,
        0.       ]), 'split27_test_score': array([ 0.        ,  0.        ,  0.        , -0.0100213 , -0.010021
3 ,
       -0.0100213 , -0.0100213 , -0.0100213 , -0.0100213 , -0.00983974,
       -0.00983974, -0.00983974, -0.00983974, -0.00983974, -0.00983974,
       -0.0100213 , -0.0100213 , -0.0100213 , -0.0100213 , -0.00983974,
       -0.00983974, -0.00983974, -0.00983974, -0.00983974, -0.0100213 ,
       -0.0100213 , -0.0100213 ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        , -0.0100213 , -0.0100213 , -0.0100213 ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        , -0.0100213 , -0.0100213 ,  0.        , -0.0100213 ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
       -0.0100213 , -0.0100213 , -0.0100213 ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        , -0.0100213 ,
       -0.0100213 , -0.0100213 ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        , -0.0100213 , -0.0100213 ,
       -0.0100213 , -0.0100213 ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        , -0.0100213 , -0.0100213 ,  0.        ,
       -0.0100213 ]), 'split28_test_score': array([-0.01503885, -0.01503885, -0.01503885, -0.01986231, -0.01986
231,
       -0.01986231,  0.        ,  0.        ,  0.        , -0.01986231,
       -0.01986231, -0.01986231, -0.01986231, -0.01986231, -0.01986231,
       -0.01986231, -0.01986231, -0.01986231, -0.01986231, -0.01986231,
       -0.01986231, -0.01986231, -0.01986231, -0.01986231, -0.01986231,
       -0.01986231, -0.01986231,  0.        ,  0.        ,  0.        ,
        0.        , -0.01986231, -0.01986231, -0.01986231, -0.01986231,
       -0.01986231, -0.01986231, -0.01986231, -0.01986231, -0.01986231,
       -0.01986231, -0.01986231, -0.01986231, -0.01986231, -0.01986231,
       -0.01986231, -0.01986231, -0.01986231, -0.01986231, -0.01986231,
       -0.01986231, -0.01986231, -0.01986231, -0.01986231,  0.        ,
        0.        ,  0.        , -0.01986231, -0.01986231, -0.01986231,
       -0.01986231, -0.01986231, -0.01986231, -0.01986231, -0.01986231,
       -0.01986231, -0.01986231, -0.01986231, -0.01986231, -0.01986231,
       -0.01986231, -0.01986231, -0.01986231, -0.01986231, -0.01986231,
       -0.01986231, -0.01986231, -0.01986231, -0.01986231, -0.01986231,
       -0.01986231]), 'split29_test_score': array([-0.01062467, -0.01062467,  0.        ,  0.        ,  0.
 ,
       -0.01062467,  0.        ,  0.        , -0.01062467, -0.01062467,
       -0.01062467, -0.01062467,  0.        ,  0.        , -0.01062467,
       -0.01062467, -0.01062467, -0.01062467, -0.01062467, -0.01062467,
       -0.01062467,  0.        ,  0.        , -0.01062467, -0.01062467,
       -0.01062467, -0.01062467, -0.01062467, -0.01062467, -0.01062467,
       -0.01062467, -0.01062467,  0.        ,  0.        ,  0.        ,
       -0.01062467, -0.01062467, -0.01062467, -0.01062467, -0.01062467,
       -0.01062467, -0.01062467, -0.01062467, -0.01062467, -0.01062467,
       -0.01062467, -0.01062467, -0.01062467, -0.01062467, -0.01062467,
       -0.01062467, -0.01062467, -0.01062467, -0.01062467, -0.01062467,
       -0.01062467, -0.01062467, -0.01062467, -0.01062467, -0.01062467,
        0.        ,  0.        , -0.01062467, -0.01062467, -0.01062467,
       -0.01062467, -0.01062467, -0.01062467, -0.01062467, -0.01062467,
       -0.01062467, -0.01062467, -0.01062467, -0.01062467, -0.01062467,
       -0.01062467, -0.01062467, -0.01062467, -0.01062467, -0.01062467,
       -0.01062467]), 'split30_test_score': array([-0.02565542, -0.02565542, -0.02565542, -0.02565542, -0.01133
016,
       -0.01133016, -0.0099256 , -0.01133016, -0.01133016, -0.0306311 ,
       -0.01133016, -0.01133016, -0.0306311 , -0.01133016, -0.01133016,
       -0.01133016, -0.01133016, -0.01133016, -0.0306311 , -0.01133016,
       -0.01133016, -0.0306311 , -0.01133016, -0.01133016, -0.01133016,
       -0.01133016, -0.01133016, -0.02565542, -0.02565542, -0.02565542,
       -0.01133016, -0.01133016, -0.01133016, -0.02565542, -0.01133016,
       -0.01133016, -0.01133016, -0.02565542, -0.01133016, -0.01133016,
       -0.01133016, -0.01133016, -0.01133016, -0.01133016, -0.01133016,
       -0.01133016, -0.02565542, -0.02565542, -0.01133016, -0.01133016,
       -0.01133016, -0.01133016, -0.01133016, -0.01133016, -0.02565542,
       -0.02565542, -0.02565542, -0.01133016, -0.01133016, -0.01133016,
       -0.02565542, -0.01133016, -0.01133016, -0.01133016, -0.02565542,
       -0.02565542, -0.01133016, -0.01133016, -0.01133016, -0.01133016,
       -0.01133016, -0.01133016, -0.01133016, -0.02565542, -0.02565542,
       -0.01133016, -0.01133016, -0.01133016, -0.01133016, -0.01133016,
       -0.01133016]), 'split31_test_score': array([-0.00858581, -0.00858581, -0.00858581, -0.00858581, -0.00858
581,
       -0.00858581, -0.00858581, -0.00858581, -0.00858581, -0.0239641 ,
       -0.00858581, -0.00858581, -0.00858581, -0.00858581, -0.00858581,
       -0.02259789, -0.00858581, -0.00858581, -0.0239641 , -0.00858581,
       -0.00858581, -0.00858581, -0.00858581, -0.00858581, -0.02259789,
       -0.00858581, -0.00858581, -0.00858581, -0.00858581, -0.00858581,
       -0.00858581, -0.00858581, -0.00858581, -0.00858581, -0.00858581,
       -0.00858581, -0.00858581, -0.00858581, -0.00858581, -0.00858581,
       -0.00858581, -0.00858581, -0.00858581, -0.00858581, -0.00858581,
       -0.00858581, -0.00858581, -0.00858581, -0.00858581, -0.00858581,
       -0.00858581, -0.00858581, -0.00858581, -0.00858581, -0.00858581,
       -0.00858581, -0.00858581, -0.00858581, -0.00858581, -0.00858581,
       -0.00858581, -0.00858581, -0.00858581, -0.00858581, -0.00858581,
       -0.00858581, -0.00858581, -0.00858581, -0.00858581, -0.00858581,
       -0.00858581, -0.00858581, -0.00858581, -0.00858581, -0.00858581,
       -0.00858581, -0.00858581, -0.00858581, -0.00858581, -0.00858581,
       -0.00858581]), 'split32_test_score': array([-0.01545025,  0.        ,  0.        ,  0.        ,  0.
 ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        , -0.01545025,  0.        ,  0.        ,
       -0.02470615,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        , -0.01545025,  0.        ,  0.        , -0.02470615,
        0.        ,  0.        , -0.02024199, -0.02024199, -0.02024199,
        0.        , -0.01545025, -0.01545025, -0.02024199,  0.        ,
        0.        , -0.01545025, -0.01545025, -0.01545025, -0.01545025,
       -0.01545025, -0.01545025, -0.01545025, -0.01545025, -0.01545025,
       -0.01545025, -0.01545025, -0.01545025, -0.01545025, -0.01545025,
       -0.01545025, -0.01545025, -0.01545025, -0.01545025, -0.02024199,
       -0.02024199, -0.02024199,  0.        , -0.01545025, -0.01545025,
       -0.02024199,  0.        ,  0.        , -0.01545025, -0.01545025,
       -0.01545025, -0.01545025, -0.01545025, -0.01545025, -0.01545025,
       -0.01545025, -0.01545025, -0.01545025, -0.01545025, -0.01545025,
       -0.01545025, -0.01545025, -0.01545025, -0.01545025, -0.01545025,
       -0.01545025]), 'split33_test_score': array([-0.009203  , -0.006445  , -0.006445  , -0.01969709, -0.00644
5  ,
       -0.006445  , -0.03338262, -0.006445  , -0.006445  , -0.009203  ,
       -0.03901925, -0.03901925, -0.03901925, -0.03901925, -0.03901925,
       -0.009203  , -0.03430196, -0.03901925, -0.009203  , -0.03901925,
       -0.03901925, -0.03901925, -0.03901925, -0.03901925, -0.009203  ,
       -0.03430196, -0.03901925, -0.006445  , -0.006445  , -0.006445  ,
       -0.006445  , -0.006445  , -0.006445  , -0.02329222, -0.006445  ,
       -0.006445  , -0.03901925, -0.03901925, -0.03901925, -0.03901925,
       -0.03901925, -0.03901925, -0.03901925, -0.03430196, -0.03901925,
       -0.03901925, -0.03901925, -0.03901925, -0.03901925, -0.03901925,
       -0.03901925, -0.03901925, -0.03430196, -0.03901925, -0.006445  ,
       -0.006445  , -0.006445  , -0.006445  , -0.006445  , -0.006445  ,
       -0.02329222, -0.006445  , -0.006445  , -0.03901925, -0.03901925,
       -0.03901925, -0.03901925, -0.03901925, -0.03901925, -0.03901925,
       -0.03430196, -0.03901925, -0.03901925, -0.03901925, -0.03901925,
       -0.03901925, -0.03901925, -0.03901925, -0.03901925, -0.03430196,
       -0.03901925]), 'split34_test_score': array([-0.02399759, -0.02589468, -0.02399759, -0.02399759, -0.02399
759,
       -0.02399759, -0.02563155, -0.02399759, -0.02399759, -0.02589468,
       -0.02589468, -0.02589468, -0.02589468, -0.02589468, -0.02589468,
       -0.02589468, -0.02589468, -0.02589468, -0.02589468, -0.02589468,
       -0.02589468, -0.02589468, -0.02589468, -0.02589468, -0.02589468,
       -0.02589468, -0.02589468, -0.02399759, -0.02589468, -0.02589468,
       -0.02399759, -0.02399759, -0.02399759, -0.01058767, -0.02399759,
       -0.02399759, -0.02589468, -0.02589468, -0.02589468, -0.02589468,
       -0.02589468, -0.02589468, -0.02589468, -0.02589468, -0.02589468,
       -0.02589468, -0.02589468, -0.02589468, -0.02589468, -0.02589468,
       -0.02589468, -0.02589468, -0.02589468, -0.02589468, -0.02589468,
       -0.02589468, -0.02589468, -0.02589468, -0.02589468, -0.02399759,
       -0.02589468, -0.02589468, -0.02399759, -0.02399759, -0.02399759,
       -0.01058767, -0.02399759, -0.02399759, -0.02589468, -0.02589468,
       -0.02589468, -0.02589468, -0.02589468, -0.02589468, -0.02589468,
       -0.02589468, -0.02589468, -0.02589468, -0.02589468, -0.02589468,
       -0.02589468]), 'split35_test_score': array([-0.0172447 , -0.01238402, -0.01238402, -0.0172447 , -0.01724
47 ,
       -0.01238402, -0.0172447 , -0.0172447 , -0.0172447 , -0.0172447 ,
       -0.0172447 , -0.0172447 , -0.0172447 , -0.0172447 , -0.0172447 ,
       -0.0172447 , -0.0172447 , -0.0172447 , -0.0172447 , -0.0172447 ,
       -0.0172447 , -0.0172447 , -0.0172447 , -0.0172447 , -0.0172447 ,
       -0.0172447 , -0.0172447 , -0.01238402, -0.02333597, -0.01238402,
       -0.0172447 , -0.0172447 , -0.01238402, -0.0172447 , -0.02333597,
       -0.0172447 , -0.02834927, -0.01238402, -0.01238402, -0.02834927,
       -0.02834927, -0.02834927, -0.0172447 , -0.0172447 , -0.0172447 ,
       -0.02834927, -0.01238402, -0.01238402, -0.02834927, -0.02834927,
       -0.02834927, -0.0172447 , -0.0172447 , -0.0172447 , -0.01238402,
       -0.02333597, -0.01238402, -0.0172447 , -0.0172447 , -0.01238402,
       -0.0172447 , -0.02333597, -0.02834927, -0.0172447 , -0.01238402,
       -0.01238402, -0.02834927, -0.02834927, -0.02834927, -0.0172447 ,
       -0.0172447 , -0.0172447 , -0.02834927, -0.01238402, -0.01238402,
       -0.02834927, -0.02834927, -0.02834927, -0.0172447 , -0.0172447 ,
       -0.0172447 ]), 'split36_test_score': array([-0.01249155, -0.01249155, -0.01249155, -0.02976107, -0.02956
661,
       -0.02956661, -0.02116892, -0.01249155, -0.01249155, -0.01249155,
       -0.01249155, -0.01249155, -0.01249155, -0.01249155, -0.01249155,
       -0.01249155, -0.01249155, -0.01249155, -0.01249155, -0.01249155,
       -0.01249155, -0.01249155, -0.01249155, -0.01249155, -0.01249155,
       -0.01249155, -0.01249155, -0.01249155, -0.01249155, -0.01249155,
       -0.02956661, -0.02956661, -0.02956661, -0.05947427, -0.01249155,
       -0.01249155, -0.01249155, -0.01249155, -0.01249155, -0.01249155,
       -0.01249155, -0.01249155, -0.01249155, -0.01249155, -0.01249155,
       -0.01249155, -0.01249155, -0.01249155, -0.01249155, -0.01249155,
       -0.01249155, -0.01249155, -0.01249155, -0.01249155, -0.01249155,
       -0.01249155, -0.01249155, -0.01249155, -0.01249155, -0.01249155,
       -0.01249155, -0.01249155, -0.02956661, -0.02956661, -0.02956661,
       -0.05947427, -0.01249155, -0.01249155, -0.01249155, -0.01249155,
       -0.01249155, -0.01249155, -0.01249155, -0.01249155, -0.01249155,
       -0.01249155, -0.01249155, -0.01249155, -0.01249155, -0.01249155,
       -0.01249155]), 'split37_test_score': array([-0.00853785, -0.00853785, -0.01916341, -0.01916341, -0.01916
341,
       -0.01017   , -0.01916341, -0.01916341, -0.00853785, -0.03587706,
       -0.00853785, -0.00853785, -0.00853785, -0.00853785, -0.00853785,
       -0.00853785, -0.00853785, -0.00853785, -0.03587706, -0.00853785,
       -0.00853785, -0.00853785, -0.00853785, -0.00853785, -0.00853785,
       -0.00853785, -0.00853785, -0.03587706, -0.03587706, -0.00853785,
       -0.01886496, -0.00853785, -0.00853785, -0.0173441 , -0.01916341,
       -0.01916341, -0.0173441 , -0.0173441 , -0.0173441 , -0.00853785,
       -0.00853785, -0.00853785, -0.00853785, -0.00853785, -0.00853785,
       -0.0173441 , -0.0173441 , -0.0173441 , -0.00853785, -0.00853785,
       -0.00853785, -0.00853785, -0.00853785, -0.00853785, -0.03587706,
       -0.03587706, -0.00853785, -0.01886496, -0.00853785, -0.00853785,
       -0.0173441 , -0.01916341, -0.01916341, -0.0173441 , -0.0173441 ,
       -0.0173441 , -0.00853785, -0.00853785, -0.00853785, -0.00853785,
       -0.00853785, -0.00853785, -0.0173441 , -0.0173441 , -0.0173441 ,
       -0.00853785, -0.00853785, -0.00853785, -0.00853785, -0.00853785,
       -0.00853785]), 'split38_test_score': array([-0.02285816, -0.02285816, -0.01127459, -0.0112903 , -0.01129
03 ,
       -0.0112903 , -0.00286843, -0.01127459, -0.01127459, -0.00286843,
       -0.01722973, -0.00286843, -0.00286843, -0.0112903 , -0.00286843,
       -0.03297097, -0.01722973, -0.0112903 , -0.00286843, -0.01722973,
       -0.00286843, -0.00286843, -0.0112903 , -0.00286843, -0.03297097,
       -0.01722973, -0.0112903 , -0.01862091, -0.01862091, -0.01862091,
       -0.0112903 , -0.0112903 , -0.0112903 , -0.0112903 , -0.0112903 ,
       -0.0112903 , -0.01127459, -0.01127459, -0.01127459, -0.01127459,
       -0.01127459, -0.01127459, -0.01127459, -0.01127459, -0.01127459,
       -0.01127459, -0.01127459, -0.01127459, -0.01127459, -0.01127459,
       -0.01127459, -0.01127459, -0.01127459, -0.01127459, -0.01862091,
       -0.01862091, -0.0112903 , -0.0112903 , -0.0112903 , -0.0112903 ,
       -0.0112903 , -0.0112903 , -0.0112903 , -0.01127459, -0.01127459,
       -0.01127459, -0.01127459, -0.01127459, -0.01127459, -0.01127459,
       -0.01127459, -0.01127459, -0.01127459, -0.01127459, -0.01127459,
       -0.01127459, -0.01127459, -0.01127459, -0.01127459, -0.01127459,
       -0.01127459]), 'split39_test_score': array([-0.02589068, -0.00885152, -0.00885152, -0.00917298, -0.00917
298,
       -0.00917298, -0.01644333, -0.01644333, -0.01644333, -0.01883262,
       -0.01883262, -0.01883262, -0.01883262, -0.00917298, -0.00917298,
       -0.01883262, -0.01883262, -0.01883262, -0.01883262, -0.01883262,
       -0.01883262, -0.01883262, -0.00917298, -0.00917298, -0.01883262,
       -0.01883262, -0.01883262, -0.00885152, -0.00885152, -0.00885152,
       -0.02589068, -0.00917298, -0.00917298, -0.01644333, -0.00917298,
       -0.01644333, -0.01883262, -0.01883262, -0.00917298, -0.00917298,
       -0.00917298, -0.00917298, -0.01883262, -0.01883262, -0.01883262,
       -0.01883262, -0.01883262, -0.00917298, -0.00917298, -0.00917298,
       -0.00917298, -0.01883262, -0.01883262, -0.01883262, -0.00885152,
       -0.00885152, -0.02589068, -0.00917298, -0.00917298, -0.00917298,
       -0.01644333, -0.00917298, -0.01644333, -0.01883262, -0.01883262,
       -0.00917298, -0.00917298, -0.00917298, -0.00917298, -0.01883262,
       -0.01883262, -0.01883262, -0.01883262, -0.01883262, -0.00917298,
       -0.00917298, -0.00917298, -0.00917298, -0.01883262, -0.01883262,
       -0.01883262]), 'split40_test_score': array([-0.00093479, -0.03889107, -0.00093479, -0.03889107, -0.02480
05 ,
        0.        , -0.02785339, -0.0248005 , -0.0248005 , -0.00093479,
        0.        ,  0.        , -0.00093479, -0.0248005 , -0.00093479,
       -0.0248005 , -0.0248005 , -0.0248005 , -0.00093479,  0.        ,
        0.        , -0.00093479, -0.00093479, -0.0248005 , -0.0248005 ,
       -0.0248005 , -0.0248005 , -0.02785339, -0.00093479, -0.00093479,
       -0.0248005 , -0.0248005 , -0.0248005 , -0.0248005 , -0.0248005 ,
       -0.0248005 , -0.00093479, -0.00093479, -0.00093479, -0.00093479,
       -0.00093479, -0.00093479, -0.0248005 , -0.0248005 , -0.00093479,
       -0.00093479, -0.00093479, -0.00093479, -0.00093479, -0.00093479,
       -0.00093479, -0.0248005 , -0.0248005 , -0.00093479, -0.02785339,
       -0.00093479, -0.00093479, -0.0248005 , -0.0248005 , -0.00093479,
       -0.0248005 , -0.0248005 , -0.0248005 , -0.00093479, -0.00093479,
       -0.00093479, -0.00093479, -0.00093479, -0.00093479, -0.0248005 ,
```

-0.0248005 , -0.00093479, -0.00093479, -0.00093479, -0.00093479,
-0.00093479, -0.00093479, -0.00093479, -0.0248005 , -0.0248005 ,
-0.00093479]), 'split41_test_score': array([-0.0109947 , -0.02378801, -0.02378801, -0.02378801, -0.02378801,
-0.02378801, -0.02431828, -0.02378801, -0.02378801, -0.01197867,
-0.01197867, -0.0109947 , -0.0109947 , -0.02378801, -0.0109947 ,
-0.01197867, -0.01197867, -0.01197867, -0.01197867, -0.01197867,
-0.0109947 , -0.0109947 , -0.02378801, -0.0109947 , -0.01197867,
-0.01197867, -0.01197867, -0.0109947 , -0.01197867, -0.0109947 ,
-0.01885879, -0.01885879, -0.0109947 , -0.02378801, -0.02378801,
-0.02378801, -0.02378801, -0.01197867, -0.0109947 , -0.01197867,
-0.01197867, -0.02378801, -0.01197867, -0.01197867, -0.02378801,
-0.02378801, -0.0109947 , -0.01197867, -0.01197867, -0.01197867,
-0.02378801, -0.01197867, -0.01197867, -0.02378801, -0.0109947 ,
-0.0109947 , -0.0109947 , -0.01885879, -0.01885879, -0.0109947 ,
-0.02378801, -0.02378801, -0.02378801, -0.02378801, -0.01197867,
-0.0109947 , -0.01197867, -0.01197867, -0.02378801, -0.02378801,
-0.01197867, -0.02378801, -0.02378801, -0.01197867, -0.0109947 ,
-0.01197867, -0.01197867, -0.02378801, -0.01197867, -0.01197867,
-0.02378801]), 'split42_test_score': array([-0.01142697, -0.02047689, -0.03273235, 0. , -0.03273235,
-0.02047689, -0.03273235, -0.02047689, -0.02047689, -0.01585338,
-0.02047689, -0.02047689, -0.01585338, -0.02047689, -0.02047689,
-0.01585338, -0.02047689, -0.02047689, -0.01585338, -0.02047689,
-0.02047689, -0.01585338, -0.02047689, -0.02047689, -0.02047689,
-0.02047689, -0.02047689, -0.02047689, -0.02047689, -0.01585338,
-0.01585338, -0.03273235, -0.02047689, -0.03273235, -0.03273235,
-0.02047689, 0. , 0. , 0. , 0. ,
-0.02047689, -0.02047689, 0. , -0.02047689, -0.02047689,
0. , 0. , 0. , 0. , -0.02047689,
-0.02047689, 0. , -0.02047689, -0.02047689, -0.02047689,
-0.02047689, -0.01585338, -0.01585338, -0.03273235, -0.02047689,
-0.03273235, -0.03273235, -0.02047689, 0. , 0. ,
0. , 0. , -0.02047689, -0.02047689, 0. ,
-0.02047689, -0.02047689, 0. , 0. , 0. ,
0. , -0.02047689, -0.02047689, 0. , -0.02047689,
-0.02047689]), 'split43_test_score': array([-0.03411179, -0.01127753, -0.01841869, -0.03269157, -0.03017444,
-0.03017444, -0.03411179, -0.01127753, -0.01127753, -0.01841869,
-0.01841869, -0.01841869, -0.01841869, -0.01841869, -0.01841869,
-0.01841869, -0.01841869, -0.01841869, -0.01841869, -0.01841869,
-0.01841869, -0.01841869, -0.01841869, -0.01841869, -0.01841869,
-0.01841869, -0.01841869, -0.02216972, -0.01841869, -0.01841869,
-0.01127753, -0.01127753, -0.01127753, -0.01127753, -0.01127753,
-0.01127753, -0.01841869, -0.02216972, -0.01841869, -0.03269157,
-0.01841869, -0.01841869, -0.01841869, -0.01841869, -0.01841869,
-0.01841869, -0.02216972, -0.01841869, -0.03269157, -0.01841869,
-0.01841869, -0.01841869, -0.01841869, -0.01841869, -0.02216972,
-0.01841869, -0.01841869, -0.01127753, -0.01127753, -0.01127753,
-0.01127753, -0.01127753, -0.01127753, -0.01841869, -0.02216972,
-0.01841869, -0.03269157, -0.01841869, -0.01841869, -0.01841869,
-0.01841869, -0.01841869, -0.01841869, -0.02216972, -0.01841869,
-0.03269157, -0.01841869, -0.01841869, -0.01841869, -0.01841869,
-0.01841869]), 'split44_test_score': array([-0.0188117 , -0.0188117 , -0.0188117 , -0.00763493, -0.00763493,
-0.00763493, -0.00763493, -0.00763493, -0.00763493, -0.0277639 ,
-0.00763493, -0.00763493, -0.00763493, -0.00763493, -0.00763493,
-0.00763493, -0.00763493, -0.00763493, -0.0277639 , -0.00763493,
-0.00763493, -0.00763493, -0.00763493, -0.00763493, -0.00763493,
-0.00763493, -0.00763493, -0.0147765 , -0.00763493, -0.00763493,
-0.00763493, -0.00763493, -0.00763493, -0.00983405, -0.0188117 ,
-0.00763493, -0.00763493, -0.00763493, -0.00763493, -0.00763493,
-0.00763493, -0.00763493, -0.00763493, -0.00763493, -0.00763493,
-0.00763493, -0.00763493, -0.00763493, -0.00763493, -0.00763493,
-0.00763493, -0.00763493, -0.00763493, -0.00763493, -0.0147765 ,
-0.00763493, -0.00763493, -0.00763493, -0.00763493, -0.00763493,
-0.00983405, -0.0188117 , -0.00763493, -0.00763493, -0.00763493,
-0.00763493, -0.00763493, -0.00763493, -0.00763493, -0.00763493,
-0.00763493, -0.00763493, -0.00763493, -0.00763493, -0.00763493,
-0.00763493, -0.00763493, -0.00763493, -0.00763493, -0.00763493,
-0.00763493]), 'split45_test_score': array([-0.02306956, -0.02306956, -0.01855489, -0.01855489, -0.01855489,
-0.01855489, -0.01855489, -0.01855489, -0.01855489, -0.02306956,
-0.02306956, -0.02306956, -0.02306956, -0.02306956, -0.02306956,
-0.02306956, -0.02306956, -0.02306956, -0.02306956, -0.02306956,
-0.02306956, -0.02306956, -0.02306956, -0.02306956, -0.02306956,
-0.02306956, -0.02306956, -0.01855489, -0.01855489, -0.01855489,
-0.01663637, -0.01855489, -0.01855489, -0.01855489, -0.01855489,
-0.01855489, -0.01509302, -0.02306956, -0.02306956, -0.02306956,
-0.02306956, -0.02306956, -0.02306956, -0.02306956, -0.02306956,
-0.01509302, -0.02306956, -0.02306956, -0.02306956, -0.02306956,
-0.02306956, -0.02306956, -0.02306956, -0.02306956, -0.01855489,
-0.01855489, -0.01855489, -0.01663637, -0.01855489, -0.01855489,
-0.01855489, -0.01855489, -0.01855489, -0.01509302, -0.02306956,
-0.02306956, -0.02306956, -0.02306956, -0.02306956, -0.02306956,
-0.02306956, -0.02306956, -0.01509302, -0.02306956, -0.02306956,
-0.02306956, -0.02306956, -0.02306956, -0.02306956, -0.02306956,
-0.02306956]), 'split46_test_score': array([-0.04496832, -0.0117361 , -0.0117361 , -0.05473565, -0.0117361 ,
-0.0117361 , -0.04976599, -0.0117361 , -0.0117361 , -0.01660038,
-0.01660038, -0.01660038, -0.01660038, -0.01660038, -0.01660038,
-0.05805931, -0.01660038, -0.01660038, -0.01660038, -0.01660038,
-0.01660038, -0.01660038, -0.01660038, -0.01660038, -0.05805931,
-0.01660038, -0.01660038, -0.0117361 , -0.0117361 , -0.0117361 ,
-0.0117361 , -0.0117361 , -0.0117361 , -0.0117361 , -0.05473565,
-0.01660038, -0.01660038, -0.01660038, -0.01660038, -0.01660038,
-0.01660038, -0.01660038, -0.01660038, -0.01660038, -0.01660038,
-0.01660038, -0.01660038, -0.01660038, -0.01660038, -0.01660038,
-0.01660038, -0.01660038, -0.01660038, -0.01660038, -0.0117361 ,
-0.0117361 , -0.0117361 , -0.0117361 , -0.0117361 , -0.0117361 ,
-0.0117361 , -0.05473565, -0.01660038, -0.01660038, -0.01660038,
-0.01660038, -0.01660038, -0.01660038, -0.01660038, -0.01660038,
-0.01660038, -0.01660038, -0.01660038, -0.01660038, -0.01660038,
-0.01660038, -0.01660038, -0.01660038, -0.01660038, -0.01660038,
-0.01660038]), 'split47_test_score': array([-0.00896535, -0.00896535, -0.00896535, -0.00896535, -0.00896535,
-0.00896535, -0.01063181, -0.00896535, -0.00896535, -0.00258154,
-0.06067611, -0.04756948, -0.06067611, -0.06067611, -0.06067611,
-0.06067611, -0.06067611, -0.06067611, -0.00258154, -0.06067611,
-0.04756948, -0.06067611, -0.06067611, -0.06067611, -0.06067611,
-0.06067611, -0.06067611, -0.00896535, -0.00896535, -0.00896535,
-0.0027187 , -0.00896535, -0.00896535, -0.00896535, -0.00896535,
-0.00896535, -0.06067611, -0.06067611, -0.03964681, -0.06067611,
-0.06067611, -0.06067611, -0.06067611, -0.06067611, -0.06067611,
-0.06067611, -0.06067611, -0.03964681, -0.06067611, -0.06067611,
-0.06067611, -0.06067611, -0.06067611, -0.06067611, -0.00896535,
-0.00896535, -0.00896535, -0.0027187 , -0.00896535, -0.00896535,
-0.00896535, -0.00896535, -0.00896535, -0.06067611, -0.06067611,
-0.03964681, -0.06067611, -0.06067611, -0.06067611, -0.06067611,
-0.06067611, -0.06067611, -0.06067611, -0.06067611, -0.03964681,
-0.06067611, -0.06067611, -0.06067611, -0.06067611, -0.06067611,
-0.06067611]), 'split48_test_score': array([-0.03253488, -0.03253488, 0. , -0.03253488, -0.04320603,
-0.04320603, -0.04320603, -0.04320603, 0. , -0.0364566 ,
-0.0364566 , 0. , -0.0364566 , 0. , 0. ,
0. , 0. , 0. , -0.0364566 , -0.0364566 ,
0. , -0.0364566 , 0. , 0. , 0. ,
0. , 0. , -0.03253488, 0. , 0. ,
0. , 0. , 0. , -0.04398129, -0.03253488,
-0.05677733, 0. , 0. , 0. , 0. ,
0. , 0. , 0. , 0. , 0. ,
0. , 0. , 0. , 0. , 0. ,
0. , 0. , 0. , 0. , -0.03253488,
0. , 0. , 0. , 0. , 0. ,
-0.04398129, -0.03253488, -0.05677733, 0. , 0. ,
0. , 0. , 0. , 0. , 0. ,
0. , 0. , 0. , 0. , 0. ,
0. , 0. , 0. , 0. , 0. ,
0.]), 'split49_test_score': array([0. , 0. , 0. , -0.05510352, -0.05510352,
0. , -0.01261898, 0. , 0. , -0.05510352,
-0.05510352, -0.04685339, -0.04643884, -0.05510352, -0.05510352,
-0.05510352, -0.03827498, -0.05510352, -0.05510352, -0.05510352,
-0.04685339, -0.04643884, -0.05510352, -0.05510352, -0.05510352,
-0.03827498, -0.05510352, -0.05454079, -0.0450196 , 0. ,
-0.02506848, 0. , 0. , -0.03279762, -0.03279762,
-0.02506848, -0.05510352, -0.05510352, -0.05510352, -0.05510352,
-0.05510352, -0.05510352, -0.05510352, -0.05510352, -0.05510352,
-0.05510352, -0.05510352, -0.05510352, -0.05510352, -0.05510352,
-0.05510352, -0.05510352, -0.05510352, -0.05510352, -0.05454079,
-0.0450196 , 0. , -0.02506848, 0. , 0. ,
-0.03279762, -0.03279762, -0.02506848, -0.05510352, -0.05510352,
-0.05510352, -0.05510352, -0.05510352, -0.05510352, -0.05510352,
-0.05510352, -0.05510352, -0.05510352, -0.05510352, -0.05510352,
-0.05510352, -0.05510352, -0.05510352, -0.05510352, -0.05510352,
-0.05510352]), 'split50_test_score': array([-0.0345263 , -0.0345263 , -0.0345263 , -0.08773421, -0.08773421,
-0.0345263 , -0.0345263 , -0.0345263 , -0.0345263 , -0.0345263 ,
-0.0345263 , -0.0345263 , -0.0345263 , -0.0345263 , -0.0345263 ,
-0.05040268, -0.05040268, -0.05040268, -0.0345263 , -0.0345263 ,
-0.0345263 , -0.0345263 , -0.0345263 , -0.0345263 , -0.05040268,
-0.05040268, -0.05040268, -0.0345263 , -0.0345263 , -0.0345263 ,
0. , -0.0345263 , -0.0345263 , -0.05927746, -0.0345263 ,
-0.08773421, -0.0345263 , -0.0345263 , -0.0345263 , -0.0345263 ,
-0.0345263 , -0.0345263 , -0.05040268, -0.05040268, -0.05040268,
-0.0345263 , -0.0345263 , -0.0345263 , -0.0345263 , -0.0345263 ,
-0.0345263 , -0.05040268, -0.05040268, -0.05040268, -0.0345263 ,
-0.0345263 , -0.0345263 , 0. , -0.0345263 , -0.0345263 ,
-0.05927746, -0.0345263 , -0.08773421, -0.0345263 , -0.0345263 ,
-0.0345263 , -0.0345263 , -0.0345263 , -0.0345263 , -0.05040268,
-0.05040268, -0.05040268, -0.0345263 , -0.0345263 , -0.0345263 ,
-0.0345263 , -0.0345263 , -0.0345263 , -0.05040268, -0.05040268,
-0.05040268]), 'split51_test_score': array([-0.04645681, -0.0036591 , -0.05311096, -0.03406946, -0.03406946,
-0.03406946, -0.03015965, -0.03015965, -0.03015965, -0.04089422,
-0.04089422, -0.04089422, -0.04230339, -0.03406946, -0.04230339,
-0.04230339, -0.04089422, -0.04089422, -0.04089422, -0.04089422,
-0.04089422, -0.04230339, -0.03406946, -0.04230339, -0.04230339,
-0.04089422, -0.04089422, -0.03612807, -0.03406946, -0.03406946,
-0.04875408, -0.03406946, -0.03406946, -0.03015965, -0.03784148,
-0.03784148, -0.03406946, -0.03406946, -0.03406946, -0.04230339,
-0.03406946, -0.03406946, -0.04230339, -0.04230339, -0.04780508,
-0.03406946, -0.03406946, -0.03406946, -0.04230339, -0.03406946,
-0.03406946, -0.04230339, -0.04230339, -0.04780508, -0.03612807,
-0.03406946, -0.03406946, -0.04875408, -0.03406946, -0.03406946,
-0.03015965, -0.03784148, -0.03784148, -0.03406946, -0.03406946,
-0.03406946, -0.04230339, -0.03406946, -0.03406946, -0.04230339,
-0.04230339, -0.04780508, -0.03406946, -0.03406946, -0.03406946,
-0.04230339, -0.03406946, -0.03406946, -0.04230339, -0.04230339,
-0.04780508]), 'split52_test_score': array([-0.01834451, -0.02124007, -0.0316382 , -0.02432721, -0.0439386 ,
-0.0316382 , -0.04723427, -0.0158236 , -0.06152819, -0.03237523,
-0.03237523, -0.03237523, -0.02124007, -0.01047262, -0.02124007,
-0.04278179, -0.04278179, -0.01047262, -0.03237523, -0.03237523,
-0.03237523, -0.02124007, -0.01047262, -0.02124007, -0.04278179,
-0.04278179, -0.01047262, -0.02124007, -0.03055939, -0.02124007,
-0.0316382 , -0.01947131, -0.0316382 , -0.03229364, -0.0316382 ,
-0.0316382 , -0.03055939, -0.03055939, -0.03055939, -0.03055939,
-0.01047262, -0.03055939, -0.01947131, -0.01047262, -0.01047262,
-0.03055939, -0.03055939, -0.03055939, -0.03055939, -0.01047262,
-0.03055939, -0.01947131, -0.01047262, -0.01047262, -0.02124007,
-0.03055939, -0.02124007, -0.0316382 , -0.01947131, -0.0316382 ,
-0.03229364, -0.0316382 , -0.0316382 , -0.03055939, -0.03055939,
-0.03055939, -0.03055939, -0.01047262, -0.03055939, -0.01947131,
-0.01047262, -0.01047262, -0.03055939, -0.03055939, -0.03055939,
-0.03055939, -0.01047262, -0.03055939, -0.01947131, -0.01047262,
-0.01047262]), 'split53_test_score': array([-0.00332037, -0.02736611, -0.02736611, -0.02553667, -0.02470733,
-0.02470733, -0.04074667, -0.04311203, -0.04311203, -0.04074667,
-0.0175306 , -0.02470733, -0.04074667, -0.02470733, -0.02470733,
-0.02470733, -0.02470733, -0.02470733, -0.04074667, -0.0175306 ,
-0.02470733, -0.04074667, -0.02470733, -0.02470733, -0.02470733,
-0.02470733, -0.02470733, -0.03934008, -0.03230402, -0.02877947,
-0.02965158, -0.02965158, -0.02470733, -0.0175306 , -0.01317344,
-0.01535228, -0.02877947, -0.02877947, -0.02877947, -0.02877947,
-0.02877947, -0.02470733, -0.02470733, -0.02470733, -0.04074667,
-0.02877947, -0.02877947, -0.02877947, -0.02877947, -0.02877947,
-0.02470733, -0.02470733, -0.02470733, -0.04074667, -0.03934008,
-0.03230402, -0.02877947, -0.02965158, -0.02965158, -0.02470733,
-0.0175306 , -0.01317344, -0.01535228, -0.02877947, -0.02877947,
-0.02877947, -0.02877947, -0.02877947, -0.02470733, -0.02470733,
-0.02470733, -0.04074667, -0.02877947, -0.02877947, -0.02877947,
-0.02877947, -0.02877947, -0.02470733, -0.02470733, -0.02470733,
-0.04074667]), 'split54_test_score': array([-0.02615246, -0.01353868, -0.01353868, -0.03570755, -0.0140585 ,
-0.03804492, -0.00225428, -0.03861648, -0.00225428, -0.03804492,
-0.03804492, -0.03804492, -0.03804492, -0.03804492, -0.03804492,
-0.02615246, -0.02615246, -0.01353868, -0.03804492, -0.03804492,
-0.03804492, -0.03804492, -0.03804492, -0.03804492, -0.02615246,
-0.02615246, -0.01353868, -0.0196231 , -0.0196231 , -0.0196231 ,
-0.00225428, -0.0196231 , -0.0196231 , -0.03757732, -0.03016014,
-0.03016014, -0.0140585 , -0.03952067, -0.0140585 , -0.0140585 ,
-0.0140585 , -0.0140585 , -0.03804492, -0.0140585 , -0.0140585 ,
-0.0140585 , -0.03952067, -0.0140585 , -0.0140585 , -0.0140585 ,
-0.0140585 , -0.03804492, -0.0140585 , -0.0140585 , -0.0196231 ,
-0.0196231 , -0.0196231 , -0.00225428, -0.0196231 , -0.0196231 ,
-0.03757732, -0.03016014, -0.03016014, -0.0140585 , -0.03952067,
-0.0140585 , -0.0140585 , -0.0140585 , -0.0140585 , -0.03804492,
-0.0140585 , -0.0140585 , -0.0140585 , -0.03952067, -0.0140585 ,
-0.0140585 , -0.0140585 , -0.0140585 , -0.03804492, -0.0140585 ,
-0.0140585]), 'split55_test_score': array([-0.03562337, -0.03562337, -0.02216265, -0.04239547, -0.04207632,
-0.02715555, -0.03123136, -0.03527387, -0.02505411, -0.03527387,
-0.03527387, -0.0186622 , -0.00989881, -0.02216265, -0.02216265,
-0.0186622 , -0.02216265, -0.03527387, -0.03527387, -0.03527387,
-0.0186622 , -0.00989881, -0.02216265, -0.02216265, -0.0186622 ,
-0.02216265, -0.03527387, -0.00989881, -0.02505411, -0.0186622 ,
-0.02505411, -0.02505411, -0.02216265, -0.03527387, -0.02505411,
-0.02505411, -0.02216265, -0.02216265, -0.02216265, -0.0186622 ,
-0.0186622 , -0.0186622 , -0.0186622 , -0.0186622 , -0.0186622 ,
-0.02216265, -0.02216265, -0.02216265, -0.0186622 , -0.0186622 ,
-0.0186622 , -0.0186622 , -0.0186622 , -0.0186622 , -0.00989881,
-0.02505411, -0.0186622 , -0.02505411, -0.02505411, -0.02216265,
-0.03527387, -0.02505411, -0.02505411, -0.02216265, -0.02216265,
-0.02216265, -0.0186622 , -0.0186622 , -0.0186622 , -0.0186622 ,
-0.0186622 , -0.0186622 , -0.02216265, -0.02216265, -0.02216265,
-0.0186622 , -0.0186622 , -0.0186622 , -0.0186622 , -0.0186622 ,
-0.0186622]), 'split56_test_score': array([-0.015378 , -0.03507121, -0.03507121, -0.05199849, -0.02913495,
-0.02913495, -0.03683786, -0.03683786, -0.05199849, -0.03683786,
-0.03683786, -0.03683786, -0.03683786, -0.05199849, -0.03683786,
-0.05199849, -0.03683786, -0.03683786, -0.03683786, -0.03683786,
-0.03683786, -0.03683786, -0.05199849, -0.03683786, -0.05199849,
-0.03683786, -0.03683786, -0.05199849, -0.00969482, -0.03683786,
-0.015378 , -0.05199849, -0.05199849, -0.00969482, -0.02913495,
-0.02913495, -0.00969482, -0.05199849, -0.05199849, -0.01772468,
-0.03683786, -0.03683786, -0.05199849, -0.03683786, -0.03683786,
-0.00969482, -0.05199849, -0.01772468, -0.05199849, -0.03683786,
-0.03683786, -0.05199849, -0.03683786, -0.03683786, -0.05199849,
-0.00969482, -0.05199849, -0.015378 , -0.05199849, -0.05199849,
-0.00969482, -0.02913495, -0.02913495, -0.00969482, -0.05199849,
-0.05199849, -0.01772468, -0.03683786, -0.03683786, -0.05199849,
-0.03683786, -0.03683786, -0.00969482, -0.05199849, -0.05199849,
-0.01772468, -0.03683786, -0.03683786, -0.05199849, -0.03683786,
-0.03683786]), 'split57_test_score': array([-0.04621437, -0.05413964, -0.03614858, -0.02436981, -0.02436981,

```
       -0.02450455, -0.02450455, -0.00905318, -0.00905318, -0.02450455,
       -0.00905318, -0.00905318, -0.00905318, -0.02450455, -0.02450455,
       -0.02450455, -0.00905318, -0.00905318, -0.02450455, -0.00905318,
       -0.00905318]), 'split94_test_score': array([-0.03423681, -0.03423681, -0.02161818, -0.05232229, -0.01479
132,
       -0.07233037, -0.02576757, -0.01479132, -0.01479132, -0.02161818,
       -0.02161818, -0.02161818, -0.01479132, -0.01479132, -0.01479132,
       -0.01479132, -0.01479132, -0.01479132, -0.02161818, -0.02161818,
       -0.02161818, -0.01479132, -0.01479132, -0.01479132, -0.01479132,
       -0.01479132, -0.01479132, -0.02161818, -0.02161818, -0.03423681,
       -0.01479132, -0.01479132, -0.01479132, -0.01479132, -0.01479132,
       -0.01479132, -0.02161818, -0.02955073, -0.02161818, -0.01479132,
       -0.01479132, -0.01479132, -0.01479132, -0.01479132, -0.01479132,
       -0.02161818, -0.02955073, -0.02161818, -0.01479132, -0.01479132,
       -0.01479132, -0.01479132, -0.01479132, -0.01479132, -0.02161818,
       -0.02161818, -0.03423681, -0.01479132, -0.01479132, -0.01479132,
       -0.01479132, -0.01479132, -0.01479132, -0.02161818, -0.02955073,
       -0.02161818, -0.01479132, -0.01479132, -0.01479132, -0.01479132,
       -0.01479132, -0.01479132, -0.02161818, -0.02955073, -0.02161818,
       -0.01479132, -0.01479132, -0.01479132, -0.01479132, -0.01479132,
       -0.01479132]), 'split95_test_score': array([-0.03685942, -0.0238936 , -0.0238936 , -0.03111607, -0.02389
36 ,
       -0.0238936 , -0.0238936 , -0.03770144, -0.03770144, -0.04470614,
       -0.04470614, -0.04470614, -0.04470614, -0.04470614, -0.0238936 ,
       -0.0238936 , -0.0238936 , -0.0238936 , -0.04470614, -0.04470614,
       -0.04470614, -0.04470614, -0.04470614, -0.0238936 , -0.0238936 ,
       -0.0238936 , -0.0238936 , -0.0238936 , -0.0238936 , -0.0238936 ,
       -0.0238936 , -0.03770144, -0.03770144, -0.0238936 , -0.0238936 ,
       -0.0238936 , -0.04470614, -0.04470614, -0.04470614, -0.0238936 ,
       -0.0238936 , -0.0238936 , -0.0238936 , -0.0238936 , -0.0238936 ,
       -0.04470614, -0.04470614, -0.04470614, -0.0238936 , -0.0238936 ,
       -0.0238936 , -0.0238936 , -0.0238936 , -0.0238936 , -0.0238936 ,
       -0.0238936 , -0.0238936 , -0.0238936 , -0.03770144, -0.03770144,
       -0.0238936 , -0.0238936 , -0.0238936 , -0.04470614, -0.04470614,
       -0.04470614, -0.0238936 , -0.0238936 , -0.0238936 , -0.0238936 ,
       -0.0238936 , -0.0238936 , -0.04470614, -0.04470614, -0.04470614,
       -0.0238936 , -0.0238936 , -0.0238936 , -0.0238936 , -0.0238936 ,
       -0.0238936 ]), 'split96_test_score': array([-0.03346071, -0.03346071, -0.03428112, -0.02162077, -0.03840
221,
       -0.03840221, -0.03346071, -0.04574184, -0.03346071, -0.03346071,
       -0.03346071, -0.03346071, -0.03346071, -0.03346071, -0.03346071,
       -0.03346071, -0.03346071, -0.03346071, -0.03346071, -0.03346071,
       -0.03346071, -0.03346071, -0.03346071, -0.03346071, -0.03346071,
       -0.03346071, -0.03346071, -0.03346071, -0.03346071, -0.03346071,
       -0.03840221, -0.03840221, -0.03346071, -0.03840221, -0.03840221,
       -0.03840221, -0.03346071, -0.03346071, -0.03346071, -0.03346071,
       -0.03346071, -0.03346071, -0.03346071, -0.03346071, -0.03346071,
       -0.03346071, -0.03346071, -0.03346071, -0.03346071, -0.03346071,
       -0.03346071, -0.03346071, -0.03346071, -0.03346071, -0.03346071,
       -0.03346071, -0.03346071, -0.03840221, -0.03840221, -0.03346071,
       -0.03840221, -0.03840221, -0.03840221, -0.03346071, -0.03346071,
       -0.03346071, -0.03346071, -0.03346071, -0.03346071, -0.03346071,
       -0.03346071, -0.03346071, -0.03346071, -0.03346071, -0.03346071,
       -0.03346071, -0.03346071, -0.03346071, -0.03346071, -0.03346071,
       -0.03346071]), 'split97_test_score': array([-0.03133363, -0.03133363, -0.03133363, -0.03461356, -0.03133
363,
       -0.03133363, -0.02955285, -0.03133363,  0.        , -0.03133363,
       -0.03133363, -0.03133363, -0.01765734, -0.01765734, -0.01765734,
        0.        , -0.01765734, -0.01765734, -0.03133363, -0.03133363,
       -0.03133363, -0.01765734, -0.01765734, -0.01765734,  0.        ,
       -0.01765734, -0.01765734, -0.03133363, -0.03133363, -0.03133363,
        0.        , -0.03133363, -0.03133363,  0.        ,  0.        ,
       -0.01765734, -0.03133363, -0.03133363, -0.03133363, -0.01765734,
       -0.01765734, -0.01765734, -0.01765734, -0.03133363, -0.01765734,
       -0.03133363, -0.03133363, -0.03133363, -0.01765734, -0.01765734,
       -0.01765734, -0.01765734, -0.03133363, -0.01765734, -0.03133363,
       -0.03133363, -0.03133363,  0.        , -0.03133363, -0.03133363,
        0.        ,  0.        , -0.01765734, -0.03133363, -0.03133363,
       -0.03133363, -0.01765734, -0.01765734, -0.01765734, -0.01765734,
       -0.03133363, -0.01765734, -0.03133363, -0.03133363, -0.03133363,
       -0.01765734, -0.01765734, -0.01765734, -0.01765734, -0.03133363,
       -0.01765734]), 'split98_test_score': array([-0.01571873, -0.01571873, -0.01571873, -0.03261793, -0.01798
508,
       -0.00973804, -0.01571873, -0.01571873, -0.01798508, -0.02481582,
       -0.02481582, -0.02481582, -0.02481582, -0.01798508, -0.01798508,
       -0.01798508, -0.01798508, -0.01798508, -0.02481582, -0.02481582,
       -0.02481582, -0.02481582, -0.01798508, -0.01798508, -0.01798508,
       -0.01798508, -0.01798508, -0.01798508, -0.02481582, -0.02481582,
       -0.02481582, -0.01798508, -0.01798508, -0.02481582, -0.02481582,
       -0.01798508, -0.02481582, -0.01798508,  0.        , -0.01798508,
       -0.01798508, -0.02481582, -0.02481582, -0.02481582, -0.01798508,
       -0.01798508, -0.01798508, -0.01798508, -0.01798508, -0.01798508,
       -0.02481582, -0.02481582, -0.02481582, -0.01798508, -0.01798508,
       -0.01798508, -0.01798508, -0.01798508, -0.01798508, -0.02481582,
       -0.02481582, -0.02481582, -0.01798508, -0.02481582, -0.01798508,
        0.        , -0.01798508, -0.01798508, -0.02481582, -0.02481582,
       -0.02481582, -0.01798508, -0.01798508, -0.01798508, -0.01798508,
       -0.01798508, -0.01798508, -0.02481582, -0.02481582, -0.02481582,
       -0.01798508, -0.01798508, -0.01798508, -0.01798508, -0.01798508,
       -0.01798508]), 'split99_test_score': array([-0.03687272, -0.011646  , -0.011646  , -0.0272024 , -0.03687
272,
       -0.03687272, -0.0272024 , -0.0272024 , -0.011646  , -0.011646  ,
       -0.011646  , -0.011646  , -0.011646  , -0.01669985, -0.01669985,
       -0.011646  , -0.01669985, -0.01669985, -0.01669985, -0.011646  ,
       -0.011646  , -0.011646  , -0.01669985, -0.01669985, -0.011646  ,
       -0.01669985, -0.01669985, -0.01669985, -0.0272024 , -0.0272024 ,
       -0.04147798, -0.011646  , -0.03687272, -0.01669985, -0.01669985,
       -0.01669985, -0.011646  , -0.011646  , -0.011646  , -0.01669985,
       -0.01669985, -0.01669985, -0.011646  , -0.01669985, -0.01669985,
       -0.011646  , -0.011646  , -0.011646  , -0.01669985, -0.01669985,
       -0.01669985, -0.011646  , -0.01669985, -0.01669985, -0.0272024 ,
       -0.0272024 , -0.0272024 , -0.04147798, -0.011646  , -0.03687272,
       -0.01669985, -0.01669985, -0.01669985, -0.011646  , -0.011646  ,
       -0.011646  , -0.01669985, -0.01669985, -0.01669985, -0.011646  ,
       -0.01669985, -0.01669985, -0.011646  , -0.011646  , -0.011646  ,
       -0.01669985, -0.01669985, -0.01669985, -0.011646  , -0.01669985,
       -0.01669985]), 'mean_test_score': array([-0.01977774, -0.02020183, -0.02017084, -0.024466  , -0.0236003
8,
       -0.02186589, -0.0238394 , -0.0206155 , -0.0209776 , -0.02328594,
       -0.02370153, -0.02309752, -0.02357   , -0.02316478, -0.02175101,
       -0.02209345, -0.02087788, -0.02279337, -0.02328594, -0.02370153,
       -0.02309752, -0.02357   , -0.02316478, -0.02175101, -0.02209345,
       -0.02087788, -0.02279337, -0.02013128, -0.02011141, -0.02076914,
       -0.02044412, -0.0217781 , -0.02106432, -0.02190585, -0.02264469,
       -0.02071127, -0.02112991, -0.02281806, -0.02242874, -0.021617  ,
       -0.02139396, -0.02120862, -0.02234542, -0.02104348, -0.02202819,
       -0.02112991, -0.02281806, -0.02242874, -0.021617  , -0.02139396,
       -0.02120862, -0.02234542, -0.02104348, -0.02202819, -0.02013128,
       -0.02011141, -0.02076914, -0.02044412, -0.0217781 , -0.02106432,
       -0.02190585, -0.02264469, -0.02071127, -0.02112991, -0.02281806,
       -0.02242874, -0.021617  , -0.02139396, -0.02120862, -0.02234542,
       -0.02104348, -0.02202819, -0.02112991, -0.02281806, -0.02242874,
       -0.021617  , -0.02139396, -0.02120862, -0.02234542, -0.02104348,
       -0.02202819]), 'std_test_score': array([0.01469599, 0.0148445 , 0.01536728, 0.01881805, 0.01941824,
       0.01753062, 0.01979168, 0.01569318, 0.01767224, 0.01620894,
       0.01938291, 0.01931037, 0.01696619, 0.01711736, 0.01615703,
       0.01612797, 0.01514793, 0.01632183, 0.01620894, 0.01938291,
       0.01931037, 0.01696619, 0.01711736, 0.01615703, 0.01612797,
       0.01514793, 0.01632183, 0.01665709, 0.0160734 , 0.01659189,
       0.01792903, 0.0174276 , 0.01721779, 0.01776393, 0.01637097,
       0.01668692, 0.01596723, 0.01718255, 0.01655693, 0.01680035,
       0.01619065, 0.01644446, 0.01624691, 0.01609927, 0.01689744,
       0.01596723, 0.01718255, 0.01655693, 0.01680035, 0.01619065,
       0.01644446, 0.01624691, 0.01609927, 0.01689744, 0.01665709,
       0.0160734 , 0.01659189, 0.01792903, 0.0174276 , 0.01721779,
       0.01776393, 0.01637097, 0.01668692, 0.01596723, 0.01718255,
       0.01655693, 0.01680035, 0.01619065, 0.01644446, 0.01624691,
       0.01609927, 0.01689744, 0.01596723, 0.01718255, 0.01655693,
       0.01680035, 0.01619065, 0.01644446, 0.01624691, 0.01609927,
       0.01689744]), 'rank_test_score': array([ 1,  7,  6, 81, 77, 44, 80, 10, 17, 73, 78, 69, 75, 71, 40, 51,
 15,
       63, 73, 78, 69, 75, 71, 40, 51, 15, 63,  4,  2, 13,  8, 42, 22, 45,
       61, 11, 24, 65, 57, 36, 32, 28, 53, 18, 47, 24, 65, 57, 36, 32, 28,
       53, 18, 47,  4,  2, 13,  8, 42, 22, 45, 61, 11, 24, 65, 57, 36, 32,
       28, 53, 18, 47, 24, 65, 57, 36, 32, 28, 53, 18, 47])}
```

In [266…
```
# Saving the trained grid search model to be used later without retraining the model again
joblib.dump(gs, 'grid_search_rf.pkl')
```
Out[266]:
```
['grid_search_rf.pkl']
```

In [149…
```
d = gs.cv_results_
model1_rank_test_score = d['rank_test_score']
dict_keys_list_rf= list(d.keys())
split_list_rf = dict_keys_list_rf[9:109]

error_per_group_for_best_model_rf = []
for i in split_list_rf:
    error_per_group_for_best_model_rf.append(d[i][model1_rank_test_score.argmin()])

error_per_group_for_best_model_rf = np.array(error_per_group_for_best_model_rf)
rmse_best_model_rf = rmse(error_per_group_for_best_model_rf)

# RMSE SCORE for the best model (from Grid Search) - Random Forest - TRAIN DATA
print('RMSE score from cross-validation scores:', rmse_best_model_rf)
```
```
RMSE score from cross-validation scores: 0.024640033321510747
```

In [268…
```
# RMSE SCORE for the best model (from Grid Search) - Random Forest - TRAIN DATA
rf_best_model = gs.best_estimator_
y_train_pred = np.array(rf_best_model.predict(X_train))

errors = np.array(error_func(y_train, y_train_pred))
rmse4 = rmse(errors)

rmse4
```
Out[268]:
```
0.014038304452422363
```

In [269…
```
# RMSE SCORE for the best model (from Grid Search) - Random Forest - TEST DATA
y_test_pred = np.array(rf_best_model.predict(X_test))

errors = np.array(error_func(y_test, y_test_pred))
rmse5 = rmse(errors)

rmse5
```
Out[269]:
```
0.030838583953280804
```

In [270…
```
rf_best_model
```
Out[270]:
```
RandomForestRegressor(max_depth=10, max_features='sqrt', n_estimators=50,
                      random_state=42)
```

ML Model Fitting and Scoring (Gradient Boosting Regressor)

3.3 Train Gradient Boosting Regressor

In [65]:
```
start = time.time()

gb = GradientBoostingRegressor(random_state=0)
gb.fit(X_train, y_train)

print("Time spent on training:", (time.time() - start))
print("R2 for train set:", gb.score(X_train, y_train))
print("R2 for test set:", gb.score(X_test, y_test))
```
```
Time spent on training: 1.6755609512329102
R2 for train set: 0.8856903753319958
R2 for test set: 0.7453764940160047
```

3.4 Error Calculation

In [66]:
```
# RMSE for GB on TRAIN DATA
y_train_pred = np.array(gb.predict(X_train))

errors = np.array(error_func(y_train, y_train_pred))
rmse6 = rmse(errors)

rmse6
```
Out[66]:
```
0.021465999188979543
```

In [67]:
```
# RMSE for GB on TEST DATA
y_test_pred = gb.predict(X_test)

errors = np.array(error_func(y_test, y_test_pred))
rmse7 = rmse(errors)

rmse7
```
Out[67]:
```
0.0211611879029742
```

4. Cross Validation

In [68]:
```
start = time.time()

scores = cross_val_score(gb, X_train, y_train, cv=logo, groups=groups_train, scoring=error_score, n_jobs=-1)

print('Mean cross validation score:', np.mean(scores))
print('Time spent on cross validation:', time.time() - start)
print('\nAll cross validation scores:', scores)
```
```
Mean cross validation score: -0.02167907333
Time spent on cross validation: 52.41505765914917

All cross validation scores: [-0.01124595 -0.03026619 -0.02419869  0.          0.         -0.04092369
 -0.00102065 -0.11404314 -0.00157561  0.         -0.01040851 -0.00535117
 -0.01289691 -0.00069682 -0.00463188 -0.07883552 -0.00990789 -0.02545382
 -0.01817669 -0.02259991 -0.01527124 -0.01204386 -0.02332473  0.
 -0.03297156 -0.02236745 -0.00616481 -0.0100213  -0.01986231 -0.01062467
 -0.01133016 -0.00858581  0.         -0.009203   -0.02399759 -0.01238402
 -0.0143158  -0.01886496 -0.0112903  -0.00885152 -0.02785339 -0.0109947
  0.         -0.01127753 -0.00983405 -0.01855489 -0.01660038 -0.04756948
 -0.04698652  0.         -0.0345263  -0.02702829 -0.03488718 -0.02877947
 -0.0140585  -0.02715555 -0.03358171 -0.05359752 -0.00630362 -0.02762686
  0.         -0.01836908  0.          0.         -0.00891985 -0.02389754
 -0.00689038 -0.0456611  -0.0309798  -0.05899711 -0.02848035 -0.01838918
 -0.01548873 -0.02352401 -0.04577496 -0.01406366 -0.03340309 -0.05542948
 -0.01392337 -0.10176827  0.         -0.00243656 -0.0204791  -0.02273548
 -0.01888756 -0.02737245 -0.0207432  -0.01589934 -0.0149047  -0.0040744
 -0.04928178 -0.04353365 -0.02293788 -0.00382145 -0.0214198  -0.02724896
 -0.03744146 -0.04126707 -0.02800841 -0.04053005]
```

In [69]:
```
# RMSE SCORE for Cross Validation - GB - TRAIN DATA
rmse8 = rmse(scores)
rmse8
```
Out[69]:
```
0.029393674750418958
```

5. Hyper-parameter optimization

In [276…
```
# Define the hyperparameter grid that we want to optimize for
param_grid2={'min_samples_leaf' : [1, 5, 10],
            "n_estimators":[100,150,200],
            "max_depth":[3,5,10]}
```

In [277…
```
# Fitting the GridSearchCV
gs2 = GridSearchCV(reg, param_grid=param_grid2, cv=logo, scoring=error_score, n_jobs=11)
gs2.fit(X_train, y_train, groups=groups_train)
```
Out[277]:
```
GridSearchCV(cv=LeaveOneGroupOut(),
             estimator=GradientBoostingRegressor(random_state=0), n_jobs=11,
             param_grid={'max_depth': [3, 5, 10],
                         'min_samples_leaf': [1, 5, 10],
                         'n_estimators': [100, 150, 200]},
             scoring=make_scorer(error_func))
```

In [278…
```
# Saving the trained grid search model to be used later without retraining the model again
joblib.dump(gs2, 'grid_search_gb.pkl')
```
Out[278]:
```
['grid_search_gb.pkl']
```

In [279…
```
try:
    print('Best parameters:', gs2.best_params_)
    print('Best score:', gs2.best_score_)
    print('Best estimator:', gs2.best_estimator_)
    print('CV results:', gs2.cv_results_)
except:
    print("-")
```

       -0.00157561, -0.00157561]), 'split9_test_score': array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]), 'split10_test_score': array([-0.01040851, -0.01040851, -0.0104
0851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851, -0.01040851, -0.01040851, -0.01040851,
       -0.01040851, -0.01040851]), 'split11_test_score': array([-0.00535117, -0.00535117, -0.00535117, -0.00535
117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117, -0.00535117, -0.00578242, -0.00578242,
       -0.00578242, -0.00535117, -0.00535117, -0.00535117, -0.00535117,
       -0.00535117, -0.00535117]), 'split12_test_score': array([-0.01289691, -0.01289691, -0.01289691, -0.01289
691, -0.01289691,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691, -0.01289691,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691, -0.01289691,
       -0.01289691, -0.01289691, -0.01289691, -0.01289691, -0.01289691,
       -0.01289691, 0.   , 0.   , 0.   , -0.01289691,
        0.   , 0.   ]), 'split13_test_score': array([-0.00069682, -0.00069682, -0.00069682, -0.00069
682, -0.00069682,
       -0.00069682, -0.00069682, -0.01591847, -0.01591847, -0.00069682,
       -0.00069682, -0.00069682, -0.00069682, -0.00069682, -0.01591847,
       -0.00069682, -0.00069682, -0.01591847, -0.00069682, -0.01674754,
       -0.01674754, -0.00069682, -0.00069682, -0.00069682, -0.01591847,
       -0.01591847, -0.01591847]), 'split14_test_score': array([-0.00463188, -0.00463188, -0.00463188, -0.00463
188, -0.00463188,
       -0.00463188, -0.00463188, -0.00463188, -0.00463188, -0.00463188,
       -0.00463188, -0.00463188, -0.00463188, -0.00463188, -0.00463188,
       -0.00463188, -0.00463188, -0.00463188, -0.00561272, -0.00561272,
       -0.00561272, 0.   , 0.   , 0.   , -0.02735766,
       -0.06160639, -0.06160639]), 'split15_test_score': array([-0.07883552, -0.03667882, -0.03667882, -0.07883
552, -0.07883552,
       -0.03667882, -0.03667882, -0.03667882, -0.03667882, -0.03660663,
       -0.03660663, -0.03660663, -0.03660663, -0.03660663, -0.03660663,
       -0.03411256, -0.03411256, -0.03411256, 0.   , 0.   ,
        0.   , -0.03660663, -0.03660663, -0.03660663, -0.03660663,
       -0.03660663, -0.03660663]), 'split16_test_score': array([-0.00990789, -0.00990789, -0.00990789, -0.00990
789, -0.00990789,
       -0.0129736 , -0.00990789, -0.00990789, -0.00990789, -0.0129736 ,
       -0.0129736 , -0.0129736 , -0.00990789, -0.0129736 , -0.0129736 ,
       -0.0129736 , -0.0129736 , -0.0129736 , -0.0129736 , -0.0129736 ,
       -0.0129736 , -0.0129736 , -0.0129736 , -0.0129736 , -0.0129736 ,
       -0.0129736 , -0.0129736]), 'split17_test_score': array([-0.02545382, -0.02545382, -0.03559386, -0.02545
382, -0.03559386,
       -0.03559386, -0.02117085, -0.02117085, -0.02117085, -0.03559386,
       -0.03559386, -0.03559386, -0.03559386, -0.03559386, -0.03870327,
       -0.03559386, -0.03559386, -0.03870327, -0.03559386, -0.03559386,
       -0.03559386, -0.03559386, -0.03559386, -0.03559386, -0.03870327,
       -0.03559386, -0.03559386]), 'split18_test_score': array([-0.01817669, -0.01817669, -0.00628353, -0.01817
669, -0.00628353,
       -0.00628353, -0.01817669, -0.01817669, -0.01485493, 0.   ,
        0.   , 0.   , 0.   , -0.00628353, -0.00628353,
       -0.00628353, 0.   , 0.   , 0.   , 0.   ,
        0.   , -0.02652747, -0.02652747, -0.02652747, -0.00628353,
       -0.00628353, -0.00628353]), 'split19_test_score': array([-0.02259991, -0.02259991, -0.02259991, -0.01146
613, -0.01146613,
       -0.01146613, -0.02259991, -0.02259991, -0.02259991, -0.02259991,
       -0.00645857, -0.00645857, -0.00645857, -0.00645857, -0.00645857,
       -0.01146613, -0.00645857, -0.00645857, -0.00645857, -0.00645857,
       -0.00645857, -0.00645857, -0.00645857, -0.00645857, -0.00645857,
       -0.00645857, -0.00645857]), 'split20_test_score': array([-0.01527124, -0.00918663, -0.00918663, -0.01527
124, -0.01527124,
       -0.01527124, -0.01527124, -0.01527124, -0.02068908, -0.02068908,
       -0.00918663, -0.00918663, 0.   , -0.01527124, -0.01527124,
        0.   , -0.02068908, -0.02068908, -0.00918663, -0.00918663,
       -0.00918663, -0.00918663, -0.00918663, -0.00918663, -0.00918663,
       -0.00918663, -0.00918663]), 'split21_test_score': array([-0.01204386, -0.01204386, -0.01204386, -0.03918
006, -0.01204386,
       -0.01204386, -0.01204386, -0.01204386, -0.01204386, -0.01204386,
       -0.01204386, -0.01204386, -0.01204386, -0.01204386, -0.01204386,
       -0.01204386, -0.01204386, -0.01204386, -0.01204386, -0.01204386,
       -0.01204386, -0.01204386, -0.01204386, -0.01204386, -0.01204386,
       -0.01204386, -0.01204386]), 'split22_test_score': array([-0.02332473, -0.02332473, -0.02332473, -0.04126
113, -0.04126113,
       -0.04126113, -0.02332473, -0.04126113, -0.04126113, -0.01450328,
       -0.01450328, -0.01450328, -0.01450328, -0.01450328, -0.01450328,
       -0.01450328, -0.01450328, -0.01450328, -0.03649216, -0.03649216,
       -0.040786 , -0.04126113, -0.04126113, -0.03649216, -0.03649216,
       -0.03649216, -0.03649216, -0.03649216, -0.03649216, -0.03649216,
       -0.03649216, -0.03649216]), 'split23_test_score': array([0.   , 0.   , 0.   , -0.00696
724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00435273, -0.00435273,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724, -0.00696724, -0.00696724, -0.00696724,
       -0.00696724, -0.00696724]), 'split24_test_score': array([-0.03297156, -0.03297156, -0.03297156, 0.
   , 0.   ,
        0.   , -0.03297156, -0.03297156, -0.03297156, 0.   ,
       -0.00638887, -0.00638887, 0.   , -0.00638887, -0.00638887,
        0.   , 0.   , 0.   , -0.00638887, -0.00638887,
       -0.00638887, -0.00638887, -0.00638887, -0.00638887, -0.03107268,
       -0.03107268, -0.03107268]), 'split25_test_score': array([-0.02236745, -0.00754795, -0.00754795, -0.00754
795, -0.00754795,
       -0.00754795, -0.02236745, -0.00754795, -0.00754795, -0.02236745,
       -0.02236745, -0.00762855, -0.00762855, -0.00762855, -0.00762855,
       -0.02236745, -0.02236745, -0.02236745, -0.02236745, -0.02236745,
       -0.02236745, -0.02236745, -0.02236745, -0.02236745, -0.02236745,
       -0.02236745, -0.02236745]), 'split26_test_score': array([-0.00616481, -0.00616481, -0.01936586, -0.00616
481, 0.   ,
        0.   , 0.   , 0.   , 0.   , 0.   ,
        0.   , 0.   , 0.   , 0.   , 0.   ,
        0.   , 0.   , 0.   , 0.   , -0.0079833 ,
       -0.0079833 , -0.0079833 , -0.0079833 , -0.0079833 , -0.0079833 ,
       -0.0079833 , -0.0079833]), 'split27_test_score': array([-0.0100213 , -0.0100213 , -0.0100213 , -0.01002
13 , -0.0100213 ,
       -0.00983974, -0.0100213 , -0.00983974, -0.00983974, -0.00983974,
        0.   , 0.   , 0.   , 0.   , 0.   ,
       -0.00983974, 0.   , -0.0100213 , 0.   , 0.   ,
        0.   , 0.   , 0.   , 0.   , -0.0100213 ,
        0.   , 0.   ]), 'split28_test_score': array([-0.01986231, -0.01986231, -0.01503885, -0.01986
231, -0.01986231,
       -0.01986231, -0.01986231, -0.01986231, -0.01986231, 0.   ,
        0.   , 0.   , -0.01986231, -0.01986231, -0.01986231,
       -0.01986231, -0.01986231, -0.01986231, 0.   , 0.   ,
        0.   , 0.   , 0.   , 0.   , 0.   ,
        0.   , 0.   ]), 'split29_test_score': array([-0.01062467, -0.01062467, -0.01062467, -0.01062
467, -0.01062467,
       -0.01062467, -0.01062467, -0.01062467, -0.01062467, 0.   ,
        0.   , 0.   , 0.   , 0.   , 0.   ,
       -0.01062467, -0.01062467, 0.   , -0.01062467, -0.01062467,
       -0.01062467, -0.01062467, -0.01062467, -0.01062467, -0.01062467,
       -0.01062467, -0.01062467]), 'split30_test_score': array([-0.01133016, -0.02565542, -0.02565542, -0.01133
016, -0.01133016,
       -0.01133016, -0.01133016, -0.01133016, -0.01133016, -0.01133016,
       -0.01133016, -0.01133016, -0.02565542, -0.02565542, -0.02565542,
       -0.01133016, -0.01133016, -0.01133016, -0.01019116, -0.01019116,
       -0.01019116, -0.01133016, -0.01133016, -0.01133016, -0.01133016,
       -0.01133016, -0.01133016]), 'split31_test_score': array([-0.00858581, -0.00858581, -0.00785826, -0.02259
789, -0.00785826,
       -0.00785826, -0.00858581, -0.00858581, -0.00858581, -0.00785826,
       -0.00785826, -0.00785826, -0.00785826, -0.00785826, -0.00785826,
       -0.00785826, -0.00785826, -0.00785826, -0.00785826, -0.00858581,
       -0.00858581, -0.00858581, -0.00858581, -0.00858581, -0.00858581,
       -0.00858581, -0.03291235]), 'split32_test_score': array([0.   , 0.   , 0.   , 0.
   , 0.   ,
        0.   , 0.   , 0.   , 0.   , -0.01545025,
       -0.01545025, -0.01545025, -0.01545025, -0.01545025, -0.01545025,
       -0.01545025, -0.01545025, -0.01545025, -0.00526222, -0.00526222,
       -0.00526222, -0.01545025, -0.01545025, -0.01545025, -0.01545025,
       -0.01545025, -0.01545025]), 'split33_test_score': array([-0.009203 , -0.009203 , -0.009203 , -0.00644
5 , -0.006445 ,
       -0.009203 , -0.006445 , -0.009203 , -0.009203 , -0.006445 ,
       -0.006445 , -0.006445 , -0.006445 , -0.006445 , -0.006445 ,
       -0.006445 , -0.006445 , -0.006445 , -0.04185789, -0.04185789,
       -0.04185789, -0.01969709, -0.01969709, -0.01969709, -0.03901925,
       -0.03901925, -0.01969709]), 'split34_test_score': array([-0.02399759, -0.02399759, -0.02589468, -0.02399
759, -0.02399759,
       -0.02589468, -0.02399759, -0.02589468, -0.02589468, -0.02399759,
       -0.02399759, -0.02399759, -0.02589468, -0.02589468, -0.02589468,
       -0.02589468, -0.02399759, -0.02399759, -0.01743191, -0.01743191,
       -0.01743191, -0.02589468, -0.02589468, -0.02589468, -0.02589468,
       -0.02589468, -0.02589468]), 'split35_test_score': array([-0.01238402, -0.01238402, -0.01238402, -0.02680
558, -0.01238402,
       -0.01238402, -0.01238402, -0.01238402, -0.01238402, -0.02680558,
       -0.01238402, -0.01238402, -0.01238402, -0.01238402, -0.01238402,
       -0.01238402, -0.01238402, -0.01238402, -0.02333597, -0.02333597,
       -0.02333597, -0.0261353 , -0.0261353 , -0.0261353 , -0.01238402,
       -0.01238402, -0.01238402]), 'split36_test_score': array([-0.0143158 , -0.0143158 , -0.0143158 , -0.02956
661, -0.02956661,
       -0.02956661, -0.0143158 , -0.02563855, -0.02563855, -0.01249155,
       -0.01249155, -0.01249155, -0.01249155, -0.01249155, -0.01249155,
       -0.01249155, -0.01249155, -0.01249155, -0.01249155, -0.01249155,
       -0.01249155, -0.01249155, -0.01249155, -0.01249155, -0.01249155,
       -0.01249155, 0.   ]), 'split37_test_score': array([-0.01886496, -0.01886496, -0.01886496, -0.01886
496, -0.01916341,
       -0.01916341, -0.01886496, -0.01886496, -0.01886496, -0.00853785,
       -0.00853785, -0.00853785, -0.00853785, -0.00853785, -0.00853785,
       -0.00853785, -0.00853785, -0.00853785, -0.0173441 , -0.0173441 ,
       -0.0173441 , -0.0173441 , -0.0173441 , -0.0173441 , -0.00853785,
       -0.00853785, -0.00853785]), 'split38_test_score': array([-0.0112903 , -0.01127459, -0.01127459, -0.00286
843, -0.0112903 ,
       -0.0112903 , -0.01127459, -0.03057287, -0.03057287, -0.01127459,
       -0.01127459, -0.01127459, -0.01127459, -0.0112903 , -0.0112903 ,
       -0.01127459, -0.01127459, -0.01127459, -0.01862091, -0.01862091,
       -0.01862091, -0.01127459, -0.01127459, -0.01127459, -0.01127459,
       -0.01127459, -0.01127459]), 'split39_test_score': array([-0.00885152, -0.00885152, -0.00885152, -0.01644
333, -0.00885152,
       -0.00885152, -0.01644333, -0.00885152, -0.00885152, -0.01644333,
       -0.00885152, -0.00885152, -0.01644333, -0.00885152, -0.00885152,
       -0.01713086, -0.00917298, -0.00917298, -0.00917298, -0.00917298,
       -0.00917298, -0.02111023, -0.02111023, -0.02111023, -0.00917298,
       -0.00917298, -0.01644333]), 'split40_test_score': array([-0.02785339, -0.02785339, -0.00093479, -0.02785
339, -0.02785339,
       -0.02785339, -0.02785339, -0.02785339, -0.02785339, -0.0248005 ,
       -0.00093479, -0.00093479, -0.0248005 , -0.0248005 , -0.0248005 ,
       -0.0248005 , -0.0248005 , -0.00093479, 0.   , 0.   ,
        0.   , -0.0248005 , -0.0248005 , -0.0248005 , -0.00093479,
       -0.00093479, -0.00093479]), 'split41_test_score': array([-0.0109947 , -0.0109947 , -0.0109947 , -0.01099
47 , -0.0109947 ,
       -0.0109947 , -0.0109947 , -0.0109947 , -0.01197867, -0.01197867,
       -0.01197867, -0.01197867, -0.0109947 , -0.0109947 , -0.0109947 ,
       -0.0109947 , -0.0109947 , -0.0109947 , 0.   , 0.   ,
        0.   , -0.0109947 , -0.0109947 , -0.0109947 , 0.   ,
        0.   , 0.   ]), 'split42_test_score': array([0.   , 0.   , 0.   , -0.02047
689, 0.   ,
        0.   , -0.02047689, -0.02047689, -0.02047689, -0.02047689,
       -0.02047689, -0.02047689, 0.   , 0.   , 0.   ,
        0.   , 0.   , 0.   , -0.01799516, -0.01799516,
       -0.01799516, -0.01585338, -0.01585338, -0.01585338, -0.01585338,
       -0.01585338, -0.01585338]), 'split43_test_score': array([-0.01127753, -0.01841869, -0.01841869, -0.01127
753, -0.01841869,
       -0.01841869, -0.02216972, -0.01841869, -0.01841869, -0.01127753,
       -0.01841869, -0.01841869, -0.01841869, -0.01841869, -0.01841869,
       -0.01841869, -0.01841869, -0.01841869, -0.00954111, -0.00954111,
       -0.00954111, -0.00954111, -0.00954111, -0.00954111, -0.02216972,
       -0.03269157, -0.03269157]), 'split44_test_score': array([-0.00983405, -0.00763493, -0.00763493, -0.00983
405, -0.00983405,
       -0.00763493, -0.00983405, -0.00763493, -0.00763493, -0.00763493,
       -0.00763493, -0.00763493, -0.00763493, -0.00763493, -0.0188117 ,
       -0.00763493, -0.0188117 , -0.00763493, -0.03803256, -0.03803256,
       -0.03803256, -0.03803256, -0.03803256, -0.03803256, -0.00763493,
       -0.00763493, -0.00763493]), 'split45_test_score': array([-0.01855489, -0.02306956, -0.02306956, -0.01855
489, -0.02306956,
       -0.02306956, -0.01855489, -0.01855489, -0.02306956, -0.02306956,
       -0.01855489, -0.01855489, -0.01855489, -0.01855489, -0.01855489,
       -0.02306956, -0.02306956, -0.01855489, 0.   , 0.   ,
        0.   , -0.02306956, 0.   , 0.   , -0.02306956,
        0.   , 0.   ]), 'split46_test_score': array([-0.01660038, -0.01660038, -0.01717989, -0.00408
344, -0.01660038,
       -0.01660038, -0.01717989, -0.01717989, -0.01717989, -0.01660038,
       -0.01660038, -0.01660038, -0.01660038, -0.00408344, -0.00408344,
       -0.01660038, -0.01660038, -0.01717989, -0.01660038, -0.01660038,
       -0.01660038, -0.01660038, -0.01660038, -0.01660038, -0.01660038,
       -0.01660038, -0.01660038]), 'split47_test_score': array([-0.04756948, -0.04295324, -0.00896535, -0.04756
948, -0.04295324,
       -0.00896535, -0.04295324, -0.00896535, -0.00896535, -0.04693107,
       -0.00896535, -0.00896535, -0.00896535, -0.00896535, -0.00896535,
       -0.01667366, -0.00896535, -0.00896535, -0.00896535, -0.00896535,
       -0.00896535, -0.00896535, -0.00258154, -0.00258154, -0.00896535,
       -0.00896535, -0.00896535]), 'split48_test_score': array([-0.04698652, 0.   , 0.   , -0.04698
652, -0.04698652,
        0.   , -0.04320603, -0.04707487, -0.04707487, -0.04707487,
       -0.02276021, -0.02276021, -0.02377545, -0.02377545, -0.02377545,
       -0.04707487, -0.04707487, -0.04707487, -0.0364566 , -0.0364566 ,
       -0.0364566 , 0.   , -0.0364566 , -0.0364566 , -0.04698652,
       -0.04698652, -0.04698652]), 'split49_test_score': array([0.   , 0.   , 0.   , 0.
   , 0.   ,
        0.   , 0.   , 0.   , 0.   , 0.   ,
        0.   , 0.   , 0.   , 0.   , -0.0450196 ,
        0.   , -0.01702137, -0.01702137, -0.05510352, -0.05510352,
       -0.03827498, 0.   , 0.   , 0.   , -0.05864883,
       -0.01702137, -0.01702137]), 'split50_test_score': array([-0.0345263 , -0.0345263 , -0.04576453, -0.07097
895, -0.04576453,
       -0.07097895, -0.04576453, -0.04576453, -0.04576453, -0.0345263 ,
       -0.0345263 , -0.0345263 , -0.0345263 , -0.06886675, -0.02430882,
       -0.0345263 , -0.0345263 , -0.02430882, -0.0345263 , -0.0345263 ,
       -0.0345263 , -0.0345263 , -0.0345263 , -0.0345263 , -0.0345263 ,
       -0.0345263 , -0.0345263]), 'split51_test_score': array([-0.02702829, -0.04875408, -0.04875408, -0.03988
944, -0.03406946,
       -0.03406946, -0.03988944, -0.03406946, -0.02720274, -0.02720274,
       -0.00871223, -0.00871223, -0.03406946, -0.00871223, -0.03406946,
       -0.03406946, -0.03252539, -0.03252539, -0.02702829, -0.02702829,
       -0.02702829, -0.03252539, -0.03252539, -0.03252539, -0.04089422,
       -0.04233855, -0.04233855]), 'split52_test_score': array([-0.03488718, -0.01047262, -0.01047262, -0.01047
262, -0.01047262,
        0.   , -0.03055939, -0.01047262, -0.01047262, -0.01047262,
       -0.01047262, -0.01047262, -0.03055939, -0.03055939, -0.03055939,
       -0.03055939, -0.03055939, -0.03055939, -0.0316382 , -0.0316382 ,
       -0.03055939, -0.03055939, -0.03055939, -0.03055939, -0.03055939,
       -0.01834451, -0.01834451]), 'split53_test_score': array([-0.02877947, -0.02877947, -0.02553667, -0.02036
417, -0.02036417,
       -0.00836874, -0.00836874, -0.00836874, -0.00836874, -0.00836874,
       -0.00836874, -0.00836874, -0.04470173, -0.04470173, -0.04470173,
       -0.02036417, -0.02036417, -0.02036417, -0.02877947, -0.02877947,
       -0.02877947, -0.02736611, -0.02877947, -0.02877947, -0.02877947,
       -0.00836874, -0.00836874]), 'split54_test_score': array([-0.0140585 , -0.01353868, -0.03804492, -0.01405
85 , -0.01353868,
       -0.03804492, -0.0140585 , -0.03804492, -0.03804492, -0.0140585 ,
       -0.0140585 , -0.0140585 , -0.0140585 , -0.0140585 , -0.0140585 ,
       -0.01353868, -0.01353868, -0.0140585 , -0.0196231 , -0.0196231 ,
       -0.0196231 , -0.0196231 , -0.0196231 , -0.0196231 , -0.01353868,
       -0.0196231 , -0.0196231]), 'split55_test_score': array([-0.02715555, 0.   , 0.   , -0.00989
881, -0.00989881,
       -0.00989881, -0.02715555, 0.   , 0.   , -0.02715555,
       -0.00989881, -0.0186622 , -0.00989881, -0.02715555, -0.02715555,
       -0.00989881, -0.02715555, -0.02505411, -0.0186622 , -0.01106246,
       -0.01106246, -0.0186622 , -0.0186622 , -0.0186622 , -0.01106246,
       -0.01106246, -0.01106246]), 'split56_test_score': array([-0.03358171, -0.01772468, -0.01772468, -0.03358
171, -0.03358171,
       -0.03358171, -0.01772468, -0.01772468, -0.01772468, -0.01772468,
       -0.01772468, -0.01772468, -0.01772468, -0.01772468, -0.01772468,
       -0.01772468, -0.01772468, -0.01772468, -0.01772468, -0.01772468,
       -0.01772468, -0.01772468, -0.05199849, -0.05199849, -0.01772468,
       -0.01772468, -0.01772468]), 'split57_test_score': array([-0.05359752, -0.05359752, -0.05359752, -0.04621
437, -0.04621437,
       -0.04621437, -0.05359752, -0.05359752, -0.05359752, -0.05359752,
       -0.05359752, -0.05359752, -0.05359752, -0.05359752, -0.05413964,
       -0.05359752, -0.05359752, -0.05359752, -0.05359752, -0.05359752,
       -0.05359752, -0.05359752, -0.05359752, -0.05359752, -0.05359752,
       -0.05359752, -0.05359752]), 'split58_test_score': array([-0.00630362, -0.00630362, -0.00630362, -0.03668
929, -0.03668929,